

Documentation
of
FullSWOF_2D

v1.08.00 (2018-01-31)

Generated by Doxygen 1.8.10
on Wed Jan 31 2018 10:37:45

Chapter 1

Todo List

Class `Boundary_condition`

Add time and space dependency in the boundary conditions.

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

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Boundary_condition	25
Bc_imp_discharge	15
Bc_imp_height	19
Bc_Neumann	20
Bc_periodic	22
Bc_wall	24
Choice_condition	28
Choice_flux	32
Choice_friction	34
Choice_infiltration	37
Choice_init_huv	38
Choice_init_topo	39
Choice_limiter	40
Choice_output	41
Choice_rain	45
Choice_reconstruction	45
Choice_save_specific_points	47
Choice_scheme	48
Flux	61
F_HLL	53
F_HLL2	55
F_HLLC	56
F_HLLC2	58
F_Rusanov	60
Friction	73
Fr_Darcy_Weisbach	64
Fr_Laminar	67
Fr_Manning	70
No_Friction	101
Hydrostatic	87
Infiltration	89
GreenAmpt	78
No_Infiltration	103
Initialization_huv	92
Huv_generated	81

Huv_generated_Radial_Dam_dry	82
Huv_generated_Radial_Dam_wet	83
Huv_generated_Thacker	84
Huv_read	85
Initialization_topo	93
Topo_generated_flat	175
Topo_generated_Thacker	176
Topo_read	177
Limiter	95
Minmod	96
VanAlbada	178
VanLeer	180
Output	111
Gnuplot	76
No_Evolution_File	99
Vtk_Out	181
Parameters	117
Parser	149
Rain	150
No_Rain	104
Rain_generated	152
Rain_read	153
Reconstruction	154
ENO	49
ENO_mod	51
MUSCL	97
Save_specific_points	156
No_save	105
One_point	107
Several_points	173
Scheme	158
Order1	108
Order2	110

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	19
Bc_Neumann	Neumann condition	20
Bc_periodic	Periodic condition	22
Bc_wall	Wall condition	24
Boundary_condition	Boundary condition	25
Choice_condition	Choice of boundary condition	28
Choice_flux	Choice of numerical flux	32
Choice_friction	Choice of friction law	34
Choice_infiltration	Choice of infiltration law	37
Choice_init_huv	Choice of initialization for h and $U=(u,v)$	38
Choice_init_topo	Choice of initialization for the topography	39
Choice_limiter	Choice of slope limiter	40
Choice_output	Choice of output format	41
Choice_rain	Choice of initialization for the rain	45
Choice_reconstruction	Choice of reconstruction	45
Choice_save_specific_points	Choice of the output of the specific points	47
Choice_scheme	Choice of numerical scheme	48

ENO	ENO reconstruction	49
ENO_mod	Modified ENO reconstruction	51
F_HLL	HLL flux	53
F_HLL2	HLL flux	55
F_HLLC	HLLC flux	56
F_HLLC2		58
F_Rusanov	Rusanov flux	60
Flux	Numerical flux	61
Fr_Darcy_Weisbach	Darcy-Weisbach law	64
Fr_Laminar	Laminar law	67
Fr_Manning	Manning law	70
Friction	Friction law	73
Gnuplot	Gnuplot output	76
GreenAmpt	Green-Ampt law	78
Huv_generated	No water configuration	81
Huv_generated_Radial_Dam_dry	Dry radial dam break configuration	82
Huv_generated_Radial_Dam_wet	Wet radial dam break configuration	83
Huv_generated_Thacker	Thacker configuration	84
Huv_read	File configuration	85
Hydrostatic	Hydrostatic reconstruction	87
Infiltration	Definition of infiltration law	89
Initialization_huv	Initialization of h, u and v	92
Initialization_topo	Initialization of z	93
Limiter	Slope limiter	95
Minmod	Minmod slope limiter	96
MUSCL	MUSCL reconstruction	97
No_Evolution_File	No output	99

No_Friction	No friction	101
No_Infiltration	No infiltration	103
No_Rain	No rain	104
No_save	No output	105
One_point	One point output	107
Order1	Order 1 scheme	108
Order2	Order 2 scheme	110
Output	Output format	111
Parameters	Gets parameters	117
Parser	Parser to read the entries	149
Rain	Initialization of the rain	150
Rain_generated	Constant rain configuration	152
Rain_read	File configuration	153
Reconstruction	Reconstruction of the variables	154
Save_specific_points	Specific points to save	156
Scheme	Numerical scheme	158
Several_points	Several points output	173
Topo_generated_flat	Flat configuration	175
Topo_generated_Thacker	Thacker configuration	176
Topo_read	File configuration	177
VanAlbada	Van Albada slope limiter	178
VanLeer	Van Leer slope limiter	180
Vtk_Out	VTK output	181

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	183
Headers/libboundaryconditions/ bc_imp_height.hpp	
Imposed water height	184
Headers/libboundaryconditions/ bc_neumann.hpp	
Neumann condition	184
Headers/libboundaryconditions/ bc_periodic.hpp	
Periodic condition	185
Headers/libboundaryconditions/ bc_wall.hpp	
Wall condition	186
Headers/libboundaryconditions/ boundary_condition.hpp	
Boundary condition	186
Headers/libboundaryconditions/ choice_condition.hpp	
Choice of boundary condition	187
Headers/libflux/ choice_flux.hpp	
Choice of numerical flux	188
Headers/libflux/ f_hll.hpp	
HLL flux	189
Headers/libflux/ f_hll2.hpp	
HLL flux	190
Headers/libflux/ f_hllc.hpp	
HLLC flux	190
Headers/libflux/ f_hllc2.hpp	
HLLC flux	191
Headers/libflux/ f_rusanov.hpp	
Rusanov flux	192
Headers/libflux/ flux.hpp	
Numerical flux	192
Headers/libfrictions/ choice_friction.hpp	
Choice of friction law	193
Headers/libfrictions/ fr_darcy_weisbach.hpp	
Darcy-Weisbach law	194
Headers/libfrictions/ fr_laminar.hpp	
Laminar law	195
Headers/libfrictions/ fr_manning.hpp	
Manning law	195

Headers/libfrictions/ friction.hpp	
Friction law	196
Headers/libfrictions/ no_friction.hpp	
No friction	196
Headers/libinitializations/ choice_init_huv.hpp	
Choice of initialization for h, u and v	197
Headers/libinitializations/ choice_init_topo.hpp	
Choice of initialization for the topography	198
Headers/libinitializations/ huv_generated.hpp	
No water configuration	199
Headers/libinitializations/ huv_generated_radial_dam_dry.hpp	
Dry radial dam break configuration	200
Headers/libinitializations/ huv_generated_radial_dam_wet.hpp	
Wet radial dam break configuration	200
Headers/libinitializations/ huv_generated_thacker.hpp	
Thacker configuration	201
Headers/libinitializations/ huv_read.hpp	
File configuration	202
Headers/libinitializations/ initialization_huv.hpp	
Initialization of h, u and v	202
Headers/libinitializations/ initialization_topo.hpp	
Initialization of z	203
Headers/libinitializations/ topo_generated_flat.hpp	
Flat configuration	204
Headers/libinitializations/ topo_generated_thacker.hpp	
Thacker configuration	204
Headers/libinitializations/ topo_read.hpp	
File configuration	205
Headers/liblimitations/ choice_limiter.hpp	
Choice of slope limiter	206
Headers/liblimitations/ limiter.hpp	
Slope limiter	207
Headers/liblimitations/ minmod.hpp	
Minmod limiter	207
Headers/liblimitations/ vanalbada.hpp	
Van Albada limiter	208
Headers/liblimitations/ vanleer.hpp	
Van Leer limiter	209
Headers/libparameters/ misc.hpp	
Definitions	209
Headers/libparameters/ parameters.hpp	
Gets parameters	212
Headers/libparser/ parser.hpp	
Parser	213
Headers/librain_infiltration/ choice_infiltration.hpp	
Choice of infiltration law	213
Headers/librain_infiltration/ choice_rain.hpp	
Choice of initialization for the rain	214
Headers/librain_infiltration/ greenampt.hpp	
Green-Ampt law	215
Headers/librain_infiltration/ infiltration.hpp	
Infiltration law	216

Headers/librain_infiltration/no_infiltration.hpp	
No infiltration	216
Headers/librain_infiltration/no_rain.hpp	
No rain	217
Headers/librain_infiltration/rain.hpp	
Rain	218
Headers/librain_infiltration/rain_generated.hpp	
Constant rain configuration	218
Headers/librain_infiltration/rain_read.hpp	
File configuration	219
Headers/libreconstructions/choice_reconstruction.hpp	
Choice of reconstruction	219
Headers/libreconstructions/eno.hpp	
ENO reconstruction	220
Headers/libreconstructions/eno_mod.hpp	
Modified ENO reconstruction	221
Headers/libreconstructions/hydrostatic.hpp	
Hydrostatic reconstruction	222
Headers/libreconstructions/muscl.hpp	
MUSCL reconstruction	222
Headers/libreconstructions/reconstruction.hpp	
Reconstruction	223
Headers/libsave/choice_output.hpp	
Choice of output format	224
Headers/libsave/choice_save_specific_points.hpp	
Choice of the output of the specific points	224
Headers/libsave/gnuplot.hpp	
Gnuplot output	225
Headers/libsave/no_evolution_file.hpp	
No output	226
Headers/libsave/no_save.hpp	
No save	227
Headers/libsave/one_point.hpp	
One point output	227
Headers/libsave/output.hpp	
Output format	228
Headers/libsave/save_specific_points.hpp	
Specific points to save	228
Headers/libsave/several_points.hpp	
Saving several specific points	229
Headers/libsave/vtk_out.hpp	
VTK output	230
Headers/libschemes/choice_scheme.hpp	
Choice of numerical scheme	230
Headers/libschemes/order1.hpp	
Order 1 scheme	231
Headers/libschemes/order2.hpp	
Order 2 scheme	232
Headers/libschemes/scheme.hpp	
Numerical scheme	233
Sources/FullSWOF_2D.cpp	
Main function	233

Sources/libboundaryconditions/bc_imp_discharge.cpp	
Imposed discharge	234
Sources/libboundaryconditions/bc_imp_height.cpp	
Imposed water height	235
Sources/libboundaryconditions/bc_neumann.cpp	
Neumann condition	235
Sources/libboundaryconditions/bc_periodic.cpp	
Periodic condition	236
Sources/libboundaryconditions/bc_wall.cpp	
Wall condition	237
Sources/libboundaryconditions/boundary_condition.cpp	
Boundary condition	237
Sources/libboundaryconditions/choice_condition.cpp	
Choice of boundary condition	238
Sources/libflux/choice_flux.cpp	
Choice of numerical flux	238
Sources/libflux/f_hll.cpp	
HLL flux	239
Sources/libflux/f_hll2.cpp	
HLL flux	239
Sources/libflux/f_hllc.cpp	
HLLC flux	240
Sources/libflux/f_hllc2.cpp	
HLLC flux	240
Sources/libflux/f_rusanov.cpp	
Rusanov flux	240
Sources/libflux/flux.cpp	
Numerical flux	241
Sources/libfrictions/choice_friction.cpp	
Choice of friction law	241
Sources/libfrictions/fr_darcy_weisbach.cpp	
Darcy-Weisbach law	242
Sources/libfrictions/fr_laminar.cpp	
Laminar law	243
Sources/libfrictions/fr_manning.cpp	
Manning law	243
Sources/libfrictions/friction.cpp	
Friction law	244
Sources/libfrictions/no_friction.cpp	
No friction	244
Sources/libinitializations/choice_init_huv.cpp	
Choice of initialization for h, u and v	245
Sources/libinitializations/choice_init_topo.cpp	
Choice of initialization for the topography	245
Sources/libinitializations/huv_generated.cpp	
No water configuration	246
Sources/libinitializations/huv_generated_radial_dam_dry.cpp	
Dry radial dam break configuration	246
Sources/libinitializations/huv_generated_radial_dam_wet.cpp	
Wet radial dam break configuration	247
Sources/libinitializations/huv_generated_thacker.cpp	
Thacker configuration	247
Sources/libinitializations/huv_read.cpp	
File configuration	248

Sources/libinitializations/ initialization_huv.cpp	
Initialization of h, u and v	248
Sources/libinitializations/ initialization_topo.cpp	
Initialization of z	249
Sources/libinitializations/ topo_generated_flat.cpp	
Flat configuration	249
Sources/libinitializations/ topo_generated_thacker.cpp	
Thacker configuration	250
Sources/libinitializations/ topo_read.cpp	
File configuration	250
Sources/liblimitations/ choice_limiter.cpp	
Choice of slope limiter	251
Sources/liblimitations/ limiter.cpp	
Slope limiter	251
Sources/liblimitations/ minmod.cpp	
Minmod limiter	252
Sources/liblimitations/ vanalbada.cpp	
Van Albada limiter	252
Sources/liblimitations/ vanleer.cpp	
Van Leer limiter	253
Sources/libparameters/ parameters.cpp	
Gets parameters	253
Sources/libparser/ parser.cpp	
Parser	254
Sources/librain_infiltration/ choice_infiltration.cpp	
Choice of infiltration law	254
Sources/librain_infiltration/ choice_rain.cpp	
Choice of initialization for the rain	255
Sources/librain_infiltration/ greenampt.cpp	
Green-Ampt law	255
Sources/librain_infiltration/ infiltration.cpp	
Infiltration law	256
Sources/librain_infiltration/ no_infiltration.cpp	
No infiltration	256
Sources/librain_infiltration/ no_rain.cpp	
No rain	257
Sources/librain_infiltration/ rain.cpp	
Rain	257
Sources/librain_infiltration/ rain_generated.cpp	
Constant rain configuration	258
Sources/librain_infiltration/ rain_read.cpp	
File configuration	258
Sources/libreconstructions/ choice_reconstruction.cpp	
Choice of reconstruction	259
Sources/libreconstructions/ eno.cpp	
ENO reconstruction	259
Sources/libreconstructions/ eno_mod.cpp	
Modified ENO reconstruction	260
Sources/libreconstructions/ hydrostatic.cpp	
Hydrostatic reconstruction	260
Sources/libreconstructions/ muscl.cpp	
MUSCL reconstruction	261

Sources/libreconstructions/ reconstruction.cpp	
Reconstruction	261
Sources/libsave/ choice_output.cpp	
Choice of output format	262
Sources/libsave/ choice_save_specific_points.cpp	
Choice of the output of the specific points	262
Sources/libsave/ gnuplot.cpp	
Gnuplot output	263
Sources/libsave/ no_evolution_file.cpp	
No output	263
Sources/libsave/ no_save.cpp	
No output	264
Sources/libsave/ one_point.cpp	
One point output	264
Sources/libsave/ output.cpp	
Output format	265
Sources/libsave/ save_specific_points.cpp	
Specific points to save	265
Sources/libsave/ several_points.cpp	
Saving several specific points	266
Sources/libsave/ vtk_out.cpp	
VTK output	266
Sources/libschemas/ choice_scheme.cpp	
Choice of numerical scheme	267
Sources/libschemas/ order1.cpp	
Order 1 scheme	267
Sources/libschemas/ order2.cpp	
Order 2 scheme	268
Sources/libschemas/ scheme.cpp	
Numerical scheme	268
Tools/ConvertFormat/ asc2xyz.c	
ArcGIS ASCII -> FullSWOF_2D XYZ	269
Tools/ConvertFormat/ xyz2asc.c	
FullSWOF_2D XYZ -> ArcGIS ASCII	270
Tools/ExtractWindow/ cropxyz.c	
Crop a FullSWOF_2D XYZ file	272

Chapter 5

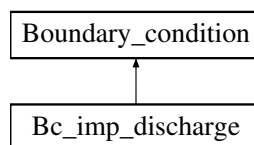
Class Documentation

5.1 Bc_imp_discharge Class Reference

Imposed discharge.

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

Inheritance diagram for Bc_imp_discharge:



Public Member Functions

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

Additional Inherited Members

5.1.1 Detailed Description

Imposed discharge.

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

Definition at line 73 of file bc_imp_discharge.hpp.

5.1.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 60 of file bc_imp_discharge.cpp.

Bc_imp_discharge::~Bc_imp_discharge () [virtual]

Destructor.

Definition at line 255 of file bc_imp_discharge.cpp.

5.1.3 Member Function Documentation

void Bc_imp_discharge::calcul (SCALAR *hin*, SCALAR *unorm_in*, SCALAR *utan_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *unorm_in_oppbound*, SCALAR *utan_in_oppbound*, SCALAR *time*, int *n1*, int *n2*) [virtual]

Calculates the boundary condition.

Two cases are considered: subcritical and supercritical flows.

Parameters

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

Warning

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

Modifies

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

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

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

Implements [Boundary_condition](#).

Definition at line 190 of file bc_imp_discharge.cpp.

SCALAR Bc_imp_discharge::getValueofDerivativeOfPolynomial (const SCALAR *HIN*, const SCALAR *UNORM_IN*, const SCALAR *UTAN_IN*, const SCALAR *H*, int *n1*, int *n2*) const

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

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

Parameters

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

Returns

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

Definition at line 132 of file `bc_imp_discharge.cpp`.

SCALAR `Bc_imp_discharge::getValueOfPolynomial` (const SCALAR *HIN*, const SCALAR *UNORM_IN*, const SCALAR *UTAN_IN*, const SCALAR *QFIX*, const SCALAR *H*, int *n1*, int *n2*) const

Gives the value of the function that must vanish.

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

Parameters

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

Returns

The value of the polynomial function.

Definition at line 112 of file `bc_imp_discharge.cpp`.

SCALAR `Bc_imp_discharge::newtonSolver` (const SCALAR *HIN*, const SCALAR *UNORM_IN*, const SCALAR *UTAN_IN*, const SCALAR *QFIX*, const SCALAR *H_INIT*, int *n1*, int *n2*) const

Solves the equation with Newton iterative method.

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

Parameters

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

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

Warning

Warning: Newton bc did not converge.

Returns

h: water height that satisfies Riemann invariants.

Definition at line 151 of file bc_imp_discharge.cpp.

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

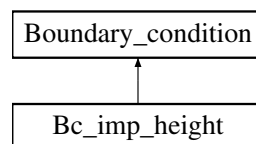
- Headers/libboundaryconditions/bc_imp_discharge.hpp
- Sources/libboundaryconditions/bc_imp_discharge.cpp

5.2 Bc_imp_height Class Reference

Imposed water height.

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

Inheritance diagram for Bc_imp_height:



Public Member Functions

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

Additional Inherited Members

5.2.1 Detailed Description

Imposed water height.

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

Definition at line 76 of file bc_imp_height.hpp.

5.2.2 Constructor & Destructor Documentation

Bc_imp_height::Bc_imp_height ([Parameters](#) & *par*, [TAB](#) & *z*, int *n1*, int *n2*)

Constructor.

Parameters

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

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

Bc_imp_height::~Bc_imp_height () [virtual]

Destructor.

Definition at line 179 of file `bc_imp_height.cpp`.

5.2.3 Member Function Documentation

void Bc_imp_height::calcul (SCALAR *hin*, SCALAR *unorm_in*, SCALAR *utan_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *unorm_in_oppbound*, SCALAR *utan_in_oppbound*, SCALAR *time*, int *n1*, int *n2*) [virtual]

Calculates the boundary condition.

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

Parameters

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

Modifies

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

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

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

Implements [Boundary_condition](#).

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

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

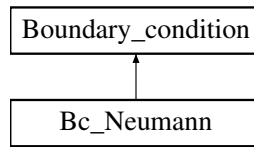
- [Headers/libboundaryconditions/bc_imp_height.hpp](#)
- [Sources/libboundaryconditions/bc_imp_height.cpp](#)

5.3 Bc_Neumann Class Reference

Neumann condition.

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

Inheritance diagram for Bc_Neumann:



Public Member Functions

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

Additional Inherited Members

5.3.1 Detailed Description

Neumann condition.

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

Definition at line 73 of file bc_neumann.hpp.

5.3.2 Constructor & Destructor Documentation

Bc_Neumann::Bc_Neumann (Parameters & par, TAB & z, int n1, int n2)

Constructor.

Parameters

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

Definition at line 61 of file bc_neumann.cpp.

Bc_Neumann::~~Bc_Neumann () [virtual]

Destructor.

Definition at line 141 of file bc_neumann.cpp.

5.3.3 Member Function Documentation

void Bc_Neumann::calcul (SCALAR hin, SCALAR unorm_in, SCALAR utan_in, SCALAR hfix, SCALAR qfix, SCALAR hin_oppbound, SCALAR unorm_in_oppbound, SCALAR utan_in_oppbound, SCALAR time, int n1, int n2) [virtual]

Calculates the boundary condition.

Parameters

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

Modifies

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

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

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

Implements [Boundary_condition](#).

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

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

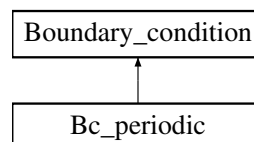
- `Headers/libboundaryconditions/bc_neumann.hpp`
- `Sources/libboundaryconditions/bc_neumann.cpp`

5.4 Bc_periodic Class Reference

Periodic condition.

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

Inheritance diagram for `Bc_periodic`:

**Public Member Functions**

- `Bc_periodic` ([Parameters](#) &, [TAB](#) &, int, int)

Constructor.

- void `calcul` ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), int, int)

Calculates boundary condition.

- virtual `~Bc_periodic` ()

Destructor.

Additional Inherited Members

5.4.1 Detailed Description

Periodic condition.

Class that computes the periodic boundary condition

Definition at line 74 of file bc_periodic.hpp.

5.4.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 61 of file bc_periodic.cpp.

Bc_periodic::~~Bc_periodic () [virtual]

Destructor.

Definition at line 144 of file bc_periodic.cpp.

5.4.3 Member Function Documentation

void Bc_periodic::calcul (SCALAR *hin*, SCALAR *unorm_in*, SCALAR *utan_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *unorm_in_oppbound*, SCALAR *utan_in_oppbound*, SCALAR *time*, int *n1*, int *n2*) [virtual]

Calculates boundary condition.

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

Parameters

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

Modifies

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

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

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

Implements [Boundary_condition](#).

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

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

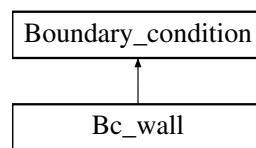
- [Headers/libboundaryconditions/bc_periodic.hpp](#)
- [Sources/libboundaryconditions/bc_periodic.cpp](#)

5.5 Bc_wall Class Reference

Wall condition.

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

Inheritance diagram for `Bc_wall`:



Public Member Functions

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

Additional Inherited Members

5.5.1 Detailed Description

Wall condition.

Class that computes the wall boundary condition.

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

5.5.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

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

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

Bc_wall::~Bc_wall() [virtual]

Destructor.

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

5.5.3 Member Function Documentation

void Bc_wall::calcul (SCALAR *hin*, SCALAR *unorm_in*, SCALAR *utan_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *unorm_in_oppbound*, SCALAR *utan_in_oppbound*, SCALAR *time*, int *n1*, int *n2*) [virtual]

Calculates the boundary condition.

Parameters

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

Modifies

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

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

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

Implements [Boundary_condition](#).

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

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

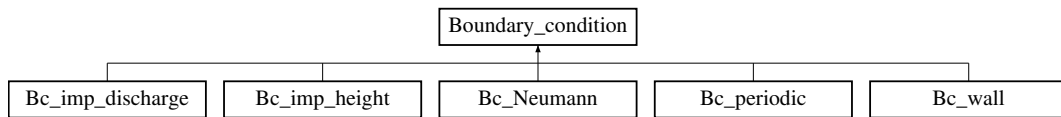
- [Headers/libboundaryconditions/bc_wall.hpp](#)
- [Sources/libboundaryconditions/bc_wall.cpp](#)

5.6 Boundary_condition Class Reference

Boundary condition.

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

Inheritance diagram for `Boundary_condition`:



Public Member Functions

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

Protected Attributes

- const int `NXCELL`
- const int `NYCELL`
- `SCALAR hbound`
- `SCALAR unormbound`
- `SCALAR utanbound`
- `SCALAR unormfix`
- `map< int, int > Choice_Lbound`
- `map< int, int > Choice_Rbound`
- `map< int, int > Choice_Bbound`
- `map< int, int > Choice_Tbound`

5.6.1 Detailed Description

Boundary condition.

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

Todo Add time and space dependency in the boundary conditions.

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

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

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

5.6.2 Constructor & Destructor Documentation

Boundary_condition::Boundary_condition (Parameters & *par*)

Constructor.

Defines the number of cells.

Parameters

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

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

Boundary_condition::~Boundary_condition () [virtual]

Destructor.

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

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

Function to be specified in each boundary condition.

Implemented in [Bc_imp_discharge](#), [Bc_imp_height](#), [Bc_periodic](#), [Bc_Neumann](#), and [Bc_wall](#).

SCALAR Boundary_condition::get_hbound () const

Gives the water height on the fictive cell.

Returns

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

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

SCALAR Boundary_condition::get_unormbound () const

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

Returns

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

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

SCALAR Boundary_condition::get_utanbound () const

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

Returns

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

Definition at line 95 of file `boundary_condition.cpp`.

5.6.4 Member Data Documentation**map<int,int> Boundary_condition::Choice_Bbound [protected]**

Definition at line 112 of file `boundary_condition.hpp`.

map<int,int> Boundary_condition::Choice_Lbound [protected]

Definition at line 110 of file `boundary_condition.hpp`.

map<int,int> Boundary_condition::Choice_Rbound [protected]

Definition at line 111 of file boundary_condition.hpp.

map<int,int> Boundary_condition::Choice_Tbound [protected]

Definition at line 113 of file boundary_condition.hpp.

SCALAR Boundary_condition::hbound [protected]

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

Definition at line 102 of file boundary_condition.hpp.

const int Boundary_condition::NXCELL [protected]

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

Definition at line 98 of file boundary_condition.hpp.

const int Boundary_condition::NYCELL [protected]

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

Definition at line 100 of file boundary_condition.hpp.

SCALAR Boundary_condition::unormbound [protected]

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

Definition at line 104 of file boundary_condition.hpp.

SCALAR Boundary_condition::unormfix [protected]

Imposed value of the velocity from [Parameters::left_imp_discharge](#) (or [Parameters::right_imp_discharge](#), [Parameters::bottom_imp_discharge](#), [Parameters::top_imp_discharge](#)) and [Parameters::left_imp_h](#) (or [Parameters::right_imp_h](#), [Parameters::bottom_imp_h](#), [Parameters::top_imp_h](#)).

Definition at line 108 of file boundary_condition.hpp.

SCALAR Boundary_condition::utanbound [protected]

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

Definition at line 106 of file boundary_condition.hpp.

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

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

5.7 Choice_condition Class Reference

Choice of boundary condition.

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

Public Member Functions

- `Choice_condition` (map< int, int > &, Parameters &, TAB &, int, int)
Constructor.
- void `calcul` (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int, int)
Calculates the boundary condition.
- `SCALAR get_hbound` ()
Gives the water height on the fictive cell.
- `SCALAR get_unormbound` ()
Gives the normal velocity of the flow on the fictive cell.
- `SCALAR get_utanbound` ()
Gives the tangential velocity of the flow on the fictive cell.
- virtual `~Choice_condition` ()
Destructor.
- void `setXY` (const int, const int)
set index (i,j) of mesh to compute boundary condition
- void `setChoice` (map< int, int > &)
set the type of boundary condition

5.7.1 Detailed Description

Choice of boundary condition.

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

Definition at line 94 of file choice_condition.hpp.

5.7.2 Constructor & Destructor Documentation

Choice_condition::Choice_condition (map< int, int > & choice, Parameters & par, TAB & z, int n1, int n2)

Constructor.

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

Parameters

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

Definition at line 61 of file choice_condition.cpp.

Choice_condition::~~Choice_condition () [virtual]

Destructor.

Definition at line 241 of file choice_condition.cpp.

5.7.3 Member Function Documentation

void Choice_condition::calcul (SCALAR hin, SCALAR unorm_in, SCALAR utan_in, SCALAR hfix, SCALAR qfix, SCALAR hin_oppbound, SCALAR unorm_in_oppbound, SCALAR utan_in_oppbound, SCALAR time, int n1, int n2)

Calculates the boundary condition.

Calls the calculation of the boundary condition.

Parameters

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

Definition at line 85 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 165 of file `choice_condition.cpp`.

SCALAR Choice_condition::get_unormbound ()

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

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

Returns

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

Definition at line 176 of file `choice_condition.cpp`.

SCALAR Choice_condition::get_utanbound ()

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

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

Returns

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

Definition at line 190 of file `choice_condition.cpp`.

void Choice_condition::setChoice (map< int, int > & vChoice_bound)

set the type of boundary condition

Calls the function to define the container containing the set of choices

Parameters

<code>in, out</code>	<i>container</i>	containing the choice of boundary conditions
----------------------	------------------	--

Returns

none.

Definition at line 229 of file `choice_condition.cpp`.

void Choice_condition::setXY (const int i, const int j)

set index (i,j) of mesh to compute boundary condition

Calls the function to define the point indices where the boundary condition is evaluated

Parameters

<code>in</code>	<i>index</i>	of the point in the x direction
<code>in</code>	<i>index</i>	of the point in the y direction

Warning

***: ERROR: the indexes i and j are too big/small.

Returns

none.

Definition at line 202 of file `choice_condition.cpp`.

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

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

5.8 Choice_flux Class Reference

Choice of numerical flux.

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

Public Member Functions

- [Choice_flux](#) (int)
Constructor.
- void [calcul](#) (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)
Calculates the numerical flux.
- void [set_tx](#) (SCALAR)
Sets the variable `Flux::tx`.
- [SCALAR get_f1](#) ()
Gives the first component of the numerical flux.
- [SCALAR get_f2](#) ()
Gives the second component of the numerical flux.
- [SCALAR get_f3](#) ()
Gives the third component of the numerical flux.
- [SCALAR get_cfl](#) ()
Gives the CFL value.
- virtual [~Choice_flux](#) ()
Destructor.

5.8.1 Detailed Description

Choice of numerical flux.

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

Definition at line 93 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 61 of file choice_flux.cpp.

Choice_flux::~~Choice_flux () [virtual]

Destructor.

Definition at line 161 of file choice_flux.cpp.

5.8.3 Member Function Documentation

void Choice_flux::calcul (SCALAR *h_L*, SCALAR *u_L*, SCALAR *v_L*, SCALAR *h_R*, SCALAR *u_R*, SCALAR *v_R*)

Calculates the numerical flux.

Calls the calculation of the numerical flux.

Parameters

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

Definition at line 90 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 150 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 117 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 128 of file choice_flux.cpp.

SCALAR Choice_flux::get_f3 ()

Gives the third component of the numerical flux.

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

Returns

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

Definition at line 139 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

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

Definition at line 106 of file choice_flux.cpp.

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

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

5.9 Choice_friction Class Reference

Choice of friction law.

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

Public Member Functions

- [Choice_friction](#) ([Parameters](#) &)
Constructor.
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, [SCALAR](#))
Calculates the friction term.
- [TAB get_q1mod](#) ()

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

- `TAB get_q2mod ()`

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

- `void calculSf (const TAB &, const TAB &, const TAB &)`

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

- `TAB get_Sf1 ()`

Gives the explicit friction term in the first direction.

- `TAB get_Sf2 ()`

Gives the explicit friction term in the second direction.

- `virtual ~Choice_friction ()`

Destructor.

5.9.1 Detailed Description

Choice of friction law.

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

Definition at line 89 of file choice_friction.hpp.

5.9.2 Constructor & Destructor Documentation

`Choice_friction::Choice_friction (Parameters & par)`

Constructor.

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

Parameters

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

Definition at line 60 of file choice_friction.cpp.

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

Destructor.

Definition at line 161 of file choice_friction.cpp.

5.9.3 Member Function Documentation

`void Choice_friction::calcul (const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt)`

Calculates the friction term.

Calls the calculation of the friction law.

Parameters

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

<code>in</code>	<code>q2new</code>	discharge in the second direction after the Shallow-Water computation (without friction).
<code>in</code>	<code>dt</code>	time step.

Note

The friction only affects the discharge.

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

void Choice_friction::calculSf (const TAB & h, const TAB & u, const TAB & v)

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

Calls the calculation of the explicit friction law.

Parameters

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

Note

This term will be used to compute erosion.

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

TAB Choice_friction::get_q1mod ()

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

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

Returns

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

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

TAB Choice_friction::get_q2mod ()

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

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

Returns

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

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

TAB Choice_friction::get_Sf1 ()

Gives the explicit friction term in the first direction.

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

Returns

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

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

TAB Choice_friction::get_Sf2 ()

Gives the explicit friction term in the second direction.

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

Returns

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

Definition at line 150 of file choice_friction.cpp.

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

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

5.10 Choice_infiltration Class Reference

Choice of infiltration law.

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

Public Member Functions

- [Choice_infiltration](#) ([Parameters](#) &)
Constructor.
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))
Performs the computation of the modified water height and the infiltrated volume.
- virtual [~Choice_infiltration](#) ()
Destructor.
- [TAB get_hmod](#) ()
Gives the value of the modified water height.
- [TAB get_Vin](#) ()
Gives the value of the infiltrated volume.

5.10.1 Detailed Description

Choice of infiltration law.

Class that calls the infiltration chosen in the parameters file.

Definition at line 82 of file choice_infiltration.hpp.

5.10.2 Constructor & Destructor Documentation**Choice_infiltration::Choice_infiltration ([Parameters](#) & *par*)**

Constructor.

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

Parameters

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

Definition at line 61 of file choice_infiltration.cpp.

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

Destructor.

Definition at line 112 of file choice_infiltration.cpp.

5.10.3 Member Function Documentation

void Choice_infiltration::calcul (const TAB & h, const TAB & Vin, const SCALAR dt)

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

Calls the computation of infiltration.

Parameters

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

Definition at line 80 of file choice_infiltration.cpp.

TAB Choice_infiltration::get_hmod ()

Gives the value of the modified water height.

Returns

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

Definition at line 92 of file choice_infiltration.cpp.

TAB Choice_infiltration::get_Vin ()

Gives the value of the infiltrated volume.

Returns

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

Definition at line 102 of file choice_infiltration.cpp.

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

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

5.11 Choice_init_huv Class Reference

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

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

Public Member Functions

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

5.11.1 Detailed Description

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

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

Definition at line 93 of file choice_init_huv.hpp.

5.11.2 Constructor & Destructor Documentation

Choice_init_huv::Choice_init_huv (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 60 of file choice_init_huv.cpp.

Choice_init_huv::~~Choice_init_huv () [virtual]

Destructor.

Definition at line 101 of file choice_init_huv.cpp.

5.11.3 Member Function Documentation

void Choice_init_huv::initialization (TAB & *h*, TAB & *u*, TAB & *v*)

Performs the initialization.

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

Parameters

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

Definition at line 88 of file choice_init_huv.cpp.

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

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

5.12 Choice_init_topo Class Reference

Choice of initialization for the topography.

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

Public Member Functions

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

5.12.1 Detailed Description

Choice of initialization for the topography.

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

Definition at line 84 of file choice_init_topo.hpp.

5.12.2 Constructor & Destructor Documentation

Choice_init_topo::Choice_init_topo (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 60 of file choice_init_topo.cpp.

Choice_init_topo::~~Choice_init_topo () [virtual]

Destructor.

Definition at line 93 of file choice_init_topo.cpp.

5.12.3 Member Function Documentation

void Choice_init_topo::initialization (TAB & *topo*)

Performs the initialization.

Calls the initialization of the topography.

Parameters

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

Definition at line 82 of file choice_init_topo.cpp.

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

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

5.13 Choice_limiter Class Reference

Choice of slope limiter.

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

Public Member Functions

- [Choice_limiter](#) (int)
Constructor.
- void [calcul](#) (SCALAR, SCALAR)
Calculates the slope limiter.
- [SCALAR get_rec](#) () const
Gives the reconstructed value.
- virtual [~Choice_limiter](#) ()
Destructor.

5.13.1 Detailed Description

Choice of slope limiter.

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

Definition at line 84 of file choice_limiter.hpp.

Saves the infiltrated and rain volumes.

- void **result** (const **SCALAR** &, const clock_t &, const **SCALAR** &, const **SCALAR** &, const **SCALAR** &, const **SCALAR** &, const int &, const **SCALAR** &) const

Saves global values.

- void **initial** (const **TAB** &, const **TAB** &, const **TAB** &, const **TAB** &) const

Saves the initial time.

- void **final** (const **TAB** &, const **TAB** &, const **TAB** &, const **TAB** &) const

Saves the final time.

- **SCALAR** **boundaries_flux** (const **SCALAR** &, const **TAB** &, const **TAB** &, const **SCALAR** &, const **SCALAR** &, const int &, const int &) const

Saves the cumulated fluxes on the boundaries.

- void **boundaries_flux_LR** (const **SCALAR** &, const **TAB** &) const

Saves the fluxes on the left and right boundaries.

- void **boundaries_flux_BT** (const **SCALAR** &, const **TAB** &) const

Saves the fluxes on the bottom and top boundaries.

- virtual **~Choice_output** ()

Destructor.

5.14.1 Detailed Description

Choice of output format.

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

Definition at line 84 of file choice_output.hpp.

5.14.2 Constructor & Destructor Documentation

Choice_output::Choice_output (Parameters & par)

Constructor.

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

Parameters

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

Definition at line 59 of file choice_output.cpp.

Choice_output::~~Choice_output () [virtual]

Destructor.

Definition at line 202 of file choice_output.cpp.

5.14.3 Member Function Documentation

SCALAR Choice_output::boundaries_flux (const SCALAR & time, const TAB & flux_u, const TAB & flux_v, const SCALAR & dt, const SCALAR & dt_first, const int & ORDER, const int & verif) const

Saves the cumulated fluxes on the boundaries.

Calls the saving of the cumulative flux on the boundaries.

Parameters

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

Definition at line 161 of file choice_output.cpp.

void Choice_output::boundaries_flux_BT (const SCALAR & *time*, const TAB & *BT_flux*) const

Saves the fluxes on the bottom and top boundaries.

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

Parameters

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

Definition at line 190 of file choice_output.cpp.

void Choice_output::boundaries_flux_LR (const SCALAR & *time*, const TAB & *LR_flux*) const

Saves the fluxes on the left and right boundaries.

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

Parameters

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

Definition at line 178 of file choice_output.cpp.

void Choice_output::check_vol (const SCALAR & *time*, const SCALAR & *dt*, const SCALAR & *Vol_rain_tot*, const SCALAR & *Vol_inf*, const SCALAR & *Vol_of*, const SCALAR & *Vol_bound_tot*) const

Saves the infiltrated and rain volumes.

Calls the saving of the infiltrated and rain volumes.

Parameters

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

Definition at line 97 of file choice_output.cpp.

void Choice_output::final (const TAB & *z*, const TAB & *h*, const TAB & *u*, const TAB & *v*) const

Saves the final time.

Calls the saving of the final time.

Parameters

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

Definition at line 146 of file choice_output.cpp.

void Choice_output::initial (const TAB & z, const TAB & h, const TAB & u, const TAB & v) const

Saves the initial time.

Calls the saving of the initial time.

Parameters

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

Definition at line 131 of file choice_output.cpp.

void Choice_output::result (const SCALAR & time, const clock_t & cpu, const SCALAR & Vol_rain, const SCALAR & Vol_inf, const SCALAR & Vol_of, const SCALAR & FROUDE, const int & NBITER, const SCALAR & vol_output) const

Saves global values.

Calls the saving of the global values.

Parameters

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

Definition at line 113 of file choice_output.cpp.

void Choice_output::write (const TAB & h, const TAB & u, const TAB & v, const TAB & z, const SCALAR & time)

Saves the current time.

Calls the saving of the current time.

Parameters

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

Definition at line 82 of file choice_output.cpp.

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

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

5.15 Choice_rain Class Reference

Choice of initialization for the rain.

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

Public Member Functions

- [Choice_rain](#) ([Parameters](#) &)
Constructor.
- void [rain_func](#) ([SCALAR](#), [TAB](#) &)
Performs the initialization filling up the table of the rain intensity.
- virtual [~Choice_rain](#) ()
Destructor.

5.15.1 Detailed Description

Choice of initialization for the rain.

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

Definition at line 84 of file choice_rain.hpp.

5.15.2 Constructor & Destructor Documentation

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

Constructor.

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

Parameters

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

Definition at line 59 of file choice_rain.cpp.

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

Destructor.

Definition at line 94 of file choice_rain.cpp.

5.15.3 Member Function Documentation

void Choice_rain::rain_func ([SCALAR](#) *time*, [TAB](#) & *Tab_rain*)

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

Calls the initialization of the rain.

Parameters

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

Definition at line 83 of file choice_rain.cpp.

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

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

5.16 Choice_reconstruction Class Reference

Choice of reconstruction.

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

Public Member Functions

- [Choice_reconstruction](#) (Parameters &, TAB &)
Constructor.
- void [calcul](#) (TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &)
Calculates the second order reconstruction in space.
- virtual [~Choice_reconstruction](#) ()
Destructor.

5.16.1 Detailed Description

Choice of reconstruction.

Class that calls the reconstruction chosen in the parameters file.

Definition at line 86 of file choice_reconstruction.hpp.

5.16.2 Constructor & Destructor Documentation

Choice_reconstruction::Choice_reconstruction (Parameters & par, TAB & z)

Constructor.

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

Parameters

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

Definition at line 60 of file choice_reconstruction.cpp.

Choice_reconstruction::~~Choice_reconstruction () [virtual]

Destructor.

Definition at line 112 of file choice_reconstruction.cpp.

5.16.3 Member Function Documentation

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

Calculates the second order reconstruction in space.

Calls the calculation of the second order reconstruction in space.

Parameters

in	h	water height.
in	u	velocity of the flow in the first direction.
in	v	velocity of the flow in the second direction.
in	z	topography.
out	delzc1	difference between the reconstructed topographies on the left and on the right boundary of a cell in the first direction.

out	<i>delzc2</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the second direction.
out	<i>delz1</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the first direction.
out	<i>delz2</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the second direction.
out	<i>h1r</i>	reconstructed water height on the right of the cell in the first direction.
out	<i>u1r</i>	first component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>h2l</i>	reconstructed water height on the left of the cell in the second direction.
out	<i>u2l</i>	first component of the reconstructed velocity on the left of the cell in the second direction.
out	<i>v2l</i>	second component of the reconstructed velocity on the left of the cell in the second direction.

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

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

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

5.17 Choice_save_specific_points Class Reference

Choice of the output of the specific points.

```
#include <choice_save_specific_points.hpp>
```

Public Member Functions

- [Choice_save_specific_points](#) (Parameters &)
Constructor.
- void [save](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))
Save the current time.
- virtual [~Choice_save_specific_points](#) ()
Destructor.

5.17.1 Detailed Description

Choice of the output of the specific points.

From the value of the corresponding parameter, calls the savings of the corresponding specific points.

Definition at line 84 of file `choice_save_specific_points.hpp`.

5.17.2 Constructor & Destructor Documentation

Choice_save_specific_points::Choice_save_specific_points (Parameters & par)

Constructor.

Defines the number (0,1,>1) of points to save from the value given in the parameters file.

Parameters

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

Definition at line 59 of file choice_save_specific_points.cpp.

Choice_save_specific_points::~~Choice_save_specific_points () [virtual]

Destructor.

Definition at line 96 of file choice_save_specific_points.cpp.

5.17.3 Member Function Documentation

void Choice_save_specific_points::save (const TAB & h, const TAB & u, const TAB & v, const SCALAR time)

Save the current time.

Calls the saving of the current time.

Parameters

in	h	water height.
in	u	first component of the velocity.
in	v	second component of the velocity.
in	time	value of the current time.

Definition at line 82 of file choice_save_specific_points.cpp.

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

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

5.18 Choice_scheme Class Reference

Choice of numerical scheme.

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

Public Member Functions

- [Choice_scheme](#) (Parameters &)
Constructor.
- void [calcul](#) ()
Performs the scheme.
- virtual [~Choice_scheme](#) ()
Destructor.

5.18.1 Detailed Description

Choice of numerical scheme.

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

Definition at line 81 of file choice_scheme.hpp.

5.18.2 Constructor & Destructor Documentation

Choice_scheme::Choice_scheme (Parameters & par)

Constructor.

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

Parameters

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

Definition at line 60 of file choice_scheme.cpp.

Choice_scheme::~~Choice_scheme () [virtual]

Destructor.

Definition at line 88 of file choice_scheme.cpp.

5.18.3 Member Function Documentation

void Choice_scheme::calcul ()

Performs the scheme.

Calls the computation of the solution.

Definition at line 78 of file choice_scheme.cpp.

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

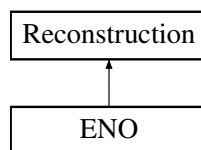
- Headers/lib schemes/[choice_scheme.hpp](#)
- Sources/lib schemes/[choice_scheme.cpp](#)

5.19 ENO Class Reference

ENO reconstruction

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

Inheritance diagram for ENO:



Public Member Functions

- [ENO \(Parameters &, TAB &\)](#)

Constructor.

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

Calculates the reconstruction in space.

- [~ENO \(\)](#)

Destructor.

Additional Inherited Members

5.19.1 Detailed Description

ENO reconstruction

Class that computes ENO reconstruction in space.

Definition at line 72 of file eno.hpp.

5.19.2 Constructor & Destructor Documentation

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

Constructor.

Initializations.

Parameters

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

Definition at line 59 of file eno.cpp.

ENO::~~ENO ()

Destructor.

Definition at line 370 of file eno.cpp.

5.19.3 Member Function Documentation

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

Calculates the reconstruction in space.

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

Parameters

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

out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>h2l</i>	reconstructed water height on the left of the cell in the second direction.
out	<i>u2l</i>	first component of the reconstructed velocity on the left of the cell in the second direction.
out	<i>v2l</i>	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 88 of file eno.cpp.

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

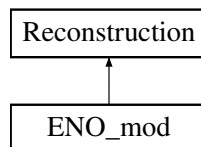
- Headers/libreconstructions/[eno.hpp](#)
- Sources/libreconstructions/[eno.cpp](#)

5.20 ENO_mod Class Reference

Modified ENO reconstruction.

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

Inheritance diagram for ENO_mod:



Public Member Functions

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

Additional Inherited Members

5.20.1 Detailed Description

Modified ENO reconstruction.

Class that computes the modified ENO reconstruction in space.

Definition at line 73 of file eno_mod.hpp.

5.20.2 Constructor & Destructor Documentation

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

Constructor.

Initializations.

Parameters

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

Definition at line 60 of file eno_mod.cpp.

ENO_mod::~~ENO_mod ()

Destructor.

Definition at line 398 of file eno_mod.cpp.

5.20.3 Member Function Documentation

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

Calculates the reconstruction in space.

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

Parameters

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

out	$h2r$	reconstructed water height on the right of the cell in the second direction.
out	$u2r$	first component of the reconstructed velocity on the right of the cell in the second direction.
out	$v2r$	second component of the reconstructed velocity on the right of the cell in the second direction.
out	$h2l$	reconstructed water height on the left of the cell in the second direction.
out	$u2l$	first component of the reconstructed velocity on the left of the cell in the second direction.
out	$v2l$	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 85 of file eno_mod.cpp.

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

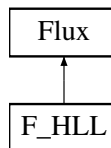
- Headers/libreconstructions/[eno_mod.hpp](#)
- Sources/libreconstructions/[eno_mod.cpp](#)

5.21 F_HLL Class Reference

HLL flux.

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

Inheritance diagram for F_HLL:



Public Member Functions

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

Additional Inherited Members

5.21.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux.

Definition at line 72 of file f_hll.hpp.

5.21.2 Constructor & Destructor Documentation

F_HLL::F_HLL ()

Constructor.

Definition at line 59 of file f_hll.cpp.

F_HLL::~~F_HLL() [virtual]

Destructor.

Definition at line 134 of file f_hll.cpp.

5.21.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

with

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

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

Parameters

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

Modifies

- Flux::f1** first component of the numerical flux.
- Flux::f2** second component of the numerical flux.
- Flux::f3** third component of the numerical flux.
- Flux::cfl** value of the CFL.

Note

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

Implements [Flux](#).

Definition at line 62 of file f_hll.cpp.

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

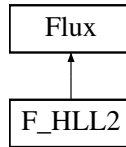
- [Headers/libflux/f_hll.hpp](#)
- [Sources/libflux/f_hll.cpp](#)

5.22 F_HLL2 Class Reference

HLL flux.

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

Inheritance diagram for F_HLL2:



Public Member Functions

- [F_HLL2 \(\)](#)
Constructor.
- void [calcul \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\)](#)
Calculates the numerical flux.
- virtual [~F_HLL2 \(\)](#)
Destructor.

Additional Inherited Members

5.22.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux with a reduced formulation.

Definition at line 71 of file f_hll2.hpp.

5.22.2 Constructor & Destructor Documentation

F_HLL2::F_HLL2 ()

Constructor.

Definition at line 60 of file f_hll2.cpp.

F_HLL2::~~F_HLL2 () [virtual]

Destructor.

Definition at line 118 of file f_hll2.cpp.

5.22.3 Member Function Documentation

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

Calculates the numerical flux.

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

If the water heights on the two sides are small or $c_1 \approx c_2 \approx 0$, there is no water. Else, HLL reduced formulation is used (see [Batten et al. \[1997\]](#)):

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

with

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

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

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

Parameters

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

Modifies

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

Note

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

Implements [Flux](#).

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

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

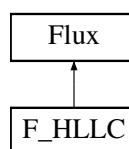
- [Headers/libflux/f_hll2.hpp](#)
- [Sources/libflux/f_hll2.cpp](#)

5.23 F_HLLC Class Reference

HLLC flux.

```
#include <f_hllc.hpp>
```

Inheritance diagram for `F_HLLC`:



Public Member Functions

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

Additional Inherited Members

5.23.1 Detailed Description

HLLC flux.

Class that computes HLLC numerical flux.

Definition at line 73 of file f_hllc.hpp.

5.23.2 Constructor & Destructor Documentation

F_HLLC::F_HLLC ()

Constructor.

Definition at line 60 of file f_hllc.cpp.

F_HLLC::~~F_HLLC () [virtual]

Destructor.

Definition at line 159 of file f_hllc.cpp.

5.23.3 Member Function Documentation

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

Calculates the numerical flux.

The HLLC approximate Riemann solver is a modification of the basic HLL scheme to account for the contact and shear waves (see [Toro \[2001\]](#)).

If the water heights on the two sides are small or $c_1 \approx c_2 \approx 0$, there is no water. Else, HLL formulation is used (see [Bouchut \[2004\]](#)):

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

with

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

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

If we consider the approximate flux HLL

$$F_{i+\frac{1}{2}} = \begin{pmatrix} F_{i+\frac{1}{2}}^1 \\ F_{i+\frac{1}{2}}^2 \\ F_{i+\frac{1}{2}}^3 \end{pmatrix}$$

then to obtain the HLLC solver just add the following expression for the third component

$$F_{i+\frac{1}{2}}^3 = \begin{cases} F_{i+\frac{1}{2}}^1 * V_L & \text{if } 0 \leq u_*, \\ F_{i+\frac{1}{2}}^1 * V_R & \text{if } u_* < 0, \end{cases}$$

Where

$$u_* = \frac{c_1 h_R (u_R - c_2) - c_2 h_L (u_L - c_1)}{h_R (u_R - c_2) - h_L (u_L - c_1)}$$

Parameters

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

Modifies

- `Flux::f1` first component of the numerical flux.
- `Flux::f2` second component of the numerical flux.
- `Flux::f3` third component of the numerical flux.
- `Flux::cfl` value of the CFL.

Note

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

Implements `Flux`.

Definition at line 63 of file `f_hllc.cpp`.

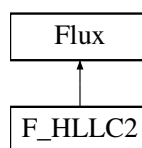
The documentation for this class was generated from the following files:

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

5.24 F_HLLC2 Class Reference

```
#include <f_hllc2.hpp>
```

Inheritance diagram for `F_HLLC2`:

**Public Member Functions**

- `F_HLLC2 ()`
Constructor.
- void `calcul (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)`
Calculates the numerical flux.
- virtual `~F_HLLC2 ()`
Destructor.

Additional Inherited Members**5.24.1 Detailed Description**

Definition at line 71 of file `f_hllc2.hpp`.

5.24.2 Constructor & Destructor Documentation

F_HLLC2::F_HLLC2 ()

Constructor.

Definition at line 60 of file f_hllc2.cpp.

F_HLLC2::~~F_HLLC2 () [virtual]

Destructor.

Definition at line 142 of file f_hllc2.cpp.

5.24.3 Member Function Documentation

void F_HLLC2::calcul (SCALAR h_L, SCALAR u_L, SCALAR v_L, SCALAR h_R, SCALAR u_R, SCALAR v_R) [virtual]

Calculates the numerical flux.

The HLLC approximate Riemann solver is a modification of the basic HLL scheme to account for the contact and shear waves (see [Toro \[2001\]](#)).

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

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

with

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

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

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

If we consider the approximate flux HLL

$$F_{i+\frac{1}{2}} = \begin{pmatrix} F_{i+\frac{1}{2}}^1 \\ F_{i+\frac{1}{2}}^2 \\ F_{i+\frac{1}{2}}^3 \end{pmatrix}$$

then to obtain the HLLC solver just add the following expression for the third component

$$F_{i+\frac{1}{2}}^3 = \begin{cases} F_{i+\frac{1}{2}}^1 * V_L & \text{if } 0 \leq u_*, \\ F_{i+\frac{1}{2}}^1 * V_R & \text{if } u_* < 0, \end{cases}$$

Where

$$u_* = \frac{c_1 h_R (u_R - c_2) - c_2 h_L (u_L - c_1)}{h_R (u_R - c_2) - h_L (u_L - c_1)}$$

Parameters

in	h_L	water height at the left of the interface where the flux is calculated.
in	u_L	velocity (in the x direction) at the left of the interface where the flux is calculated.

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

Modifies

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

Note

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

Implements [Flux](#).

Definition at line 64 of file `f_hllc2.cpp`.

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

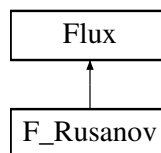
- [Headers/libflux/f_hllc2.hpp](#)
- [Sources/libflux/f_hllc2.cpp](#)

5.25 F_Rusanov Class Reference

Rusanov flux.

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

Inheritance diagram for `F_Rusanov`:



Public Member Functions

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

Additional Inherited Members

5.25.1 Detailed Description

Rusanov flux.

Class that computes Rusanov numerical flux.

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

5.25.2 Constructor & Destructor Documentation

F_Rusanov::F_Rusanov ()

Constructor.

Definition at line 59 of file f_rusanov.cpp.

F_Rusanov::~~F_Rusanov () [virtual]

Destructor.

Definition at line 106 of file f_rusanov.cpp.

5.25.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

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

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

Parameters

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

Modifies

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

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

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

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

Implements [Flux](#).

Definition at line 63 of file f_rusanov.cpp.

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

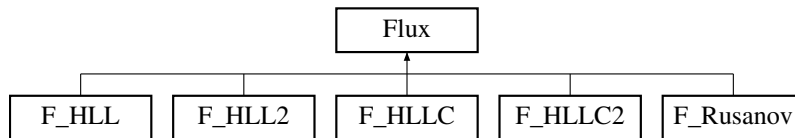
- Headers/libflux/f_rusanov.hpp
- Sources/libflux/f_rusanov.cpp

5.26 Flux Class Reference

Numerical flux.

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

Inheritance diagram for Flux:



Public Member Functions

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

Protected Attributes

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

5.26.1 Detailed Description

Numerical flux.

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

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

5.26.2 Constructor & Destructor Documentation

`Flux::Flux ()`

Constructor.

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

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

Destructor.

Definition at line 116 of file `flux.cpp`.

5.26.3 Member Function Documentation

virtual void Flux::calcul (SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR) [pure virtual]

Function to be specified in each numerical flux.

Implemented in [F_HLLC](#), [F_HLL](#), [F_HLL2](#), [F_HLLC2](#), and [F_Rusanov](#).

SCALAR Flux::get_cfl () const

Gives the CFL value.

Returns

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

Definition at line 106 of file flux.cpp.

SCALAR Flux::get_f1 () const

Gives the first component of the numerical flux.

Returns

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

Definition at line 76 of file flux.cpp.

SCALAR Flux::get_f2 () const

Gives the second component of the numerical flux.

Returns

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

Definition at line 86 of file flux.cpp.

SCALAR Flux::get_f3 () const

Gives the third component of the numerical flux.

Returns

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

Definition at line 96 of file flux.cpp.

void Flux::set_tx (SCALAR tx)

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

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

Parameters

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

Definition at line 66 of file flux.cpp.

5.26.4 Member Data Documentation

SCALAR Flux::cfl [protected]

CFL value.

Definition at line 105 of file flux.hpp.

SCALAR Flux::f1 [protected]

First component of the numerical flux.
Definition at line 99 of file flux.hpp.

SCALAR Flux::f2 [protected]

Second component of the numerical flux.
Definition at line 101 of file flux.hpp.

SCALAR Flux::f3 [protected]

Third component of the numerical flux.
Definition at line 103 of file flux.hpp.

SCALAR Flux::tx [protected]

Value of dt/dx.
Definition at line 107 of file flux.hpp.
The documentation for this class was generated from the following files:

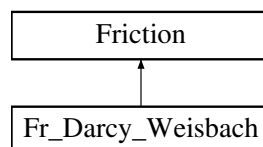
- [Headers/libflux/flux.hpp](#)
- [Sources/libflux/flux.cpp](#)

5.27 Fr_Darcy_Weisbach Class Reference

Darcy-Weisbach law.

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

Inheritance diagram for Fr_Darcy_Weisbach:

**Public Member Functions**

- [Fr_Darcy_Weisbach](#) (Parameters &)
Constructor.
- void [calcul](#) (const TAB &, const TAB &, const TAB &, const TAB &, const TAB &, SCALAR)
Calculates the Darcy-Weisbach friction term.
- void [calculSf](#) (const TAB &, const TAB &, const TAB &)
Calculates the explicit Darcy-Weisbach friction term.
- virtual [~Fr_Darcy_Weisbach](#) ()
Destructor.

Additional Inherited Members**5.27.1 Detailed Description**

Darcy-Weisbach law.

General formulation: $S_f = \frac{fU|U|}{8gh}$. This term is integrated in the code thanks to a semi-implicit method.
Definition at line 71 of file fr_darcy_weisbach.hpp.

5.27.2 Constructor & Destructor Documentation

Fr_Darcy_Weisbach::Fr_Darcy_Weisbach (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file fr_darcy_weisbach.cpp.

Fr_Darcy_Weisbach::~~Fr_Darcy_Weisbach () [virtual]

Destructor.

Definition at line 125 of file fr_darcy_weisbach.cpp.

5.27.3 Member Function Documentation**void Fr_Darcy_Weisbach::calcul (const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt) [virtual]**

Calculates the Darcy-Weisbach friction term.

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

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

where f is the friction coefficient.

Parameters

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

Modifies

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

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

Note

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

Implements [Friction](#).

Definition at line 68 of file fr_darcy_weisbach.cpp.

void Fr_Darcy_Weisbach::calculSf (const TAB & h, const TAB & u, const TAB & v) [virtual]

Calculates the explicit Darcy-Weisbach friction term.

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

Parameters

in	h	water height.
in	u	velocity in the first direction, first component of U in the above formula.
in	v	velocity in the second direction, second component of U in the above formula.

Modifies

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

Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

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

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

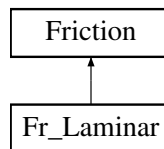
- [Headers/libfrictions/fr_darcy_weisbach.hpp](#)
- [Sources/libfrictions/fr_darcy_weisbach.cpp](#)

5.28 Fr_Laminar Class Reference

Laminar law.

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

Inheritance diagram for `Fr_Laminar`:

**Public Member Functions**

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

Additional Inherited Members

5.28.1 Detailed Description

Laminar law.

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

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

5.28.2 Constructor & Destructor Documentation

Fr_Laminar::Fr_Laminar (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 58 of file fr_laminar.cpp.

Fr_Laminar::~Fr_Laminar () [virtual]

Destructor.

Definition at line 128 of file fr_laminar.cpp.

5.28.3 Member Function Documentation**void Fr_Laminar::calcul (const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt) [virtual]**

Calculates the laminar friction term.

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

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

where v is the friction coefficient.

Parameters

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

Modifies

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

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

Note

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

Implements [Friction](#).

Definition at line 67 of file fr_laminar.cpp.

void Fr_Laminar::calculSf (const TAB & h, const TAB & u, const TAB & v) [virtual]

Calculates the explicit Manning friction term.

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

Parameters

in	h	water height.
in	u	velocity in the first direction, first component of U in the above formula.
in	v	velocity in the second direction, second component of U in the above formula.

Modifies

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

Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

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

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

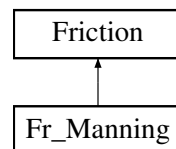
- `Headers/libfrictions/fr_laminar.hpp`
- `Sources/libfrictions/fr_laminar.cpp`

5.29 Fr_Manning Class Reference

Manning law.

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

Inheritance diagram for `Fr_Manning`:

**Public Member Functions**

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

Additional Inherited Members**5.29.1 Detailed Description**

Manning law.

General formulation: $S_f = c^2 \frac{U|U|}{h^{4/3}}$. This term is integrated in the code thanks to a semi-implicit method.

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

5.29.2 Constructor & Destructor Documentation

Fr_Manning::Fr_Manning (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file fr_manning.cpp.

Fr_Manning::~~Fr_Manning () [virtual]

Destructor.

Definition at line 126 of file fr_manning.cpp.

5.29.3 Member Function Documentation**void Fr_Manning::calcul (const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt) [virtual]**

Calculates the Manning friction term.

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

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

where c is the friction coefficient.

Parameters

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

Modifies

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

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

Note

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

Implements [Friction](#).

Definition at line 68 of file fr_manning.cpp.

void Fr_Manning::calculSf (const TAB & h, const TAB & u, const TAB & v) [virtual]

Calculates the explicit Manning friction term.

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

Parameters

in	h	water height.
in	u	velocity in the first direction, first component of U in the above formula.
in	v	velocity in the second direction, second component of U in the above formula.

Modifies

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

Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

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

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

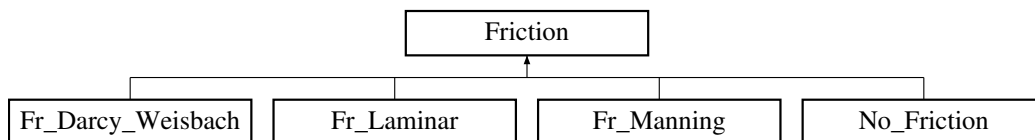
- `Headers/libfrictions/fr_manning.hpp`
- `Sources/libfrictions/fr_manning.cpp`

5.30 Friction Class Reference

Friction law

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

Inheritance diagram for Friction:

**Public Member Functions**

- [Friction](#) ([Parameters](#) &)
Constructor.
- virtual void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, [SCALAR](#))=0
Function to be specified in each friction law.
- virtual [TAB](#) [get_q1mod](#) () const
Gives the discharge in the first direction modified by the friction term.
- virtual [TAB](#) [get_q2mod](#) () const
Gives the discharge in the second direction modified by the friction term.
- virtual void [calculSf](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &)=0
Calculates the explicit friction term. It will be used for computations with erosion.
- virtual [TAB](#) [get_Sf1](#) () const
Gives the explicit friction term in the first direction.
- virtual [TAB](#) [get_Sf2](#) () const
Gives the explicit friction term in the second direction.
- virtual [~Friction](#) ()
Destructor.

Protected Attributes

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

5.30.1 Detailed Description

Friction law

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

Definition at line 72 of file friction.hpp.

5.30.2 Constructor & Destructor Documentation

Friction::Friction (Parameters & *par*)

Constructor.

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

Parameters

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

Warning

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

Initialization of [Friction](#)

Definition at line 59 of file friction.cpp.

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

Destructor.

Deallocation of [Friction::Fric_tab](#), [Friction::q1mod](#), [Friction::q2mod](#), [Friction::Sf1](#), [Friction::Sf2](#)

Definition at line 152 of file friction.cpp.

5.30.3 Member Function Documentation

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

Function to be specified in each friction law.

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

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

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

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

TAB Friction::get_q1mod () const [virtual]

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

Returns

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

Definition at line 112 of file friction.cpp.

TAB Friction::get_q2mod () const [virtual]

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

Returns

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

Definition at line 122 of file friction.cpp.

TAB Friction::get_Sf1 () const [virtual]

Gives the explicit friction term in the first direction.

Returns

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

Definition at line 132 of file friction.cpp.

TAB Friction::get_Sf2 () const [virtual]

Gives the explicit friction term in the second direction.

Returns

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

Definition at line 142 of file friction.cpp.

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

Space step in the first (x) direction.

Definition at line 106 of file friction.hpp.

const SCALAR Friction::DY [protected]

Space step in the second (y) direction.

Definition at line 108 of file friction.hpp.

TAB Friction::Fric_tab [protected]

Array that contains the friction coefficient by cell.

Definition at line 119 of file friction.hpp.

const int Friction::NXCELL [protected]

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

Definition at line 102 of file friction.hpp.

const int Friction::NYCELL [protected]

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

Definition at line 104 of file friction.hpp.

TAB Friction::q1mod [protected]

Discharge in the first direction modified by the friction term.

Definition at line 111 of file friction.hpp.

TAB Friction::q2mod [protected]

Discharge in the second direction modified by the friction term.

Definition at line 113 of file friction.hpp.

TAB Friction::Sf1 [protected]

Explicit friction term in the first direction.

Definition at line 115 of file friction.hpp.

TAB Friction::Sf2 [protected]

Explicit friction term in the second direction.

Definition at line 117 of file friction.hpp.

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

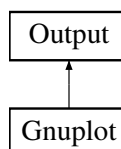
- Headers/libfrictions/[friction.hpp](#)
- Sources/libfrictions/[friction.cpp](#)

5.31 Gnuplot Class Reference

Gnuplot output

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

Inheritance diagram for Gnuplot:



Public Member Functions

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

Additional Inherited Members

5.31.1 Detailed Description

Gnuplot output

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

Definition at line 73 of file gnuplot.hpp.

5.31.2 Constructor & Destructor Documentation

Gnuplot::Gnuplot (Parameters & *par*)

Constructor.

Writes the header of the file 'huz_evolution.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 huz_evolution.dat cannot be opened, the code will exit with failure termination code.

Definition at line 60 of file gnuplot.cpp.

Gnuplot::~Gnuplot () [virtual]

Destructor.

Definition at line 125 of file gnuplot.cpp.

5.31.3 Member Function Documentation

void Gnuplot::write (const TAB & *h*, const TAB & *u*, const TAB & *v*, const TAB & *z*, const SCALAR & *time*) [virtual]

Saves one time step.

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

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

Parameters

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

Implements [Output](#).

Definition at line 89 of file gnuplot.cpp.

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

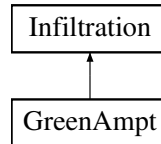
- [Headers/libsave/gnuplot.hpp](#)
- [Sources/libsave/gnuplot.cpp](#)

5.32 GreenAmpt Class Reference

Green-Ampt law.

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

Inheritance diagram for GreenAmpt:



Public Member Functions

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

Constructor.

- [SCALAR capacity](#) (const [SCALAR](#), const [SCALAR](#), const [SCALAR](#), const [SCALAR](#) Kc, const [SCALAR](#) Ks, const [SCALAR](#) dtheta, const [SCALAR](#) Psi, const [SCALAR](#) zcrust)

Calculates the infiltration capacity.

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

Calculates the infiltrated volume.

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

Destructor.

Additional Inherited Members

5.32.1 Detailed Description

Green-Ampt law.

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

Definition at line 72 of file greenampt.hpp.

5.32.2 Constructor & Destructor Documentation

GreenAmpt::GreenAmpt (Parameters & *par*)

Constructor.

Initializes the values for Green-Ampt infiltration.

Parameters

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

Warning

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

Definition at line 59 of file greenampt.cpp.

GreenAmpt::~~GreenAmpt () [virtual]

Destructor.

Definition at line 306 of file greenampt.cpp.

5.32.3 Member Function Documentation

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

Calculates the infiltrated volume.

Parameters

in	<i>h</i>	water height.
in	<i>Vin_tot</i>	total infiltrated volume.
in	<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 260 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:

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

with the effective hydraulic conductivity

$$K_e = \frac{1}{\frac{1}{K_s} * \left(1 - \frac{zcrust * dtheta}{Vin_tot}\right) + zcrust * \frac{dtheta}{Vin_tot} * \frac{1}{K_c}}$$

Parameters

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

Returns

ic: infiltration capacity.

Definition at line 216 of file greenampt.cpp.

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

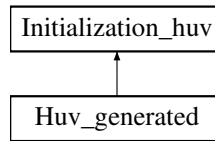
- Headers/librain_infiltration/[greenampt.hpp](#)
- Sources/librain_infiltration/[greenampt.cpp](#)

5.33 Huv_generated Class Reference

No water configuration.

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

Inheritance diagram for Huv_generated:



Public Member Functions

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

Additional Inherited Members

5.33.1 Detailed Description

No water configuration.

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

Definition at line 73 of file huv_generated.hpp.

5.33.2 Constructor & Destructor Documentation

Huv_generated::Huv_generated (Parameters & par)

Constructor.

Parameters

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

Definition at line 60 of file huv_generated.cpp.

Huv_generated::~~Huv_generated () [virtual]

Destructor.

Definition at line 86 of file huv_generated.cpp.

5.33.3 Member Function Documentation

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

Performs the initialization.

Initializes the water height and the velocity at 0.

Parameters

in, out	h	water height.
in, out	u	first component of the velocity.
in, out	v	second component of the velocity.

Implements [Initialization_huv](#).

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

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

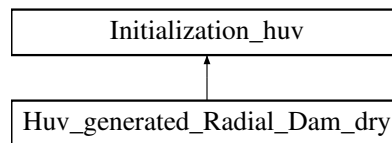
- [Headers/libinitializations/huv_generated.hpp](#)
- [Sources/libinitializations/huv_generated.cpp](#)

5.34 Huv_generated_Radial_Dam_dry Class Reference

Dry radial dam break configuration.

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

Inheritance diagram for `Huv_generated_Radial_Dam_dry`:



Public Member Functions

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

Additional Inherited Members

5.34.1 Detailed Description

Dry radial dam break configuration.

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

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

5.34.2 Constructor & Destructor Documentation

Huv_generated_Radial_Dam_dry::Huv_generated_Radial_Dam_dry (Parameters & par)

Constructor.

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

Parameters

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

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

Huv_generated_Radial_Dam_dry::~~Huv_generated_Radial_Dam_dry () [virtual]

Destructor.

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

5.34.3 Member Function Documentation

void Huv_generated_Radial_Dam_dry::initialization (TAB & *h*, TAB & *u*, TAB & *v*) [virtual]

Performs the initialization.

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

Parameters

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

Implements [Initialization_huv](#).

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

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

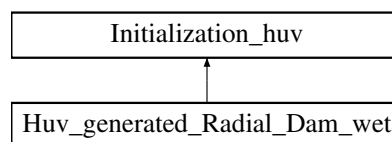
- [Headers/libinitializations/huv_generated_radial_dam_dry.hpp](#)
- [Sources/libinitializations/huv_generated_radial_dam_dry.cpp](#)

5.35 Huv_generated_Radial_Dam_wet Class Reference

Wet radial dam break configuration.

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

Inheritance diagram for `Huv_generated_Radial_Dam_wet`:



Public Member Functions

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

Additional Inherited Members

5.35.1 Detailed Description

Wet radial dam break configuration.

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

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

5.35.2 Constructor & Destructor Documentation

Huv_generated_Radial_Dam_wet::Huv_generated_Radial_Dam_wet (Parameters & par)

Constructor.

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

Parameters

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

Definition at line 60 of file huv_generated_radial_dam_wet.cpp.

Huv_generated_Radial_Dam_wet::~~Huv_generated_Radial_Dam_wet () [virtual]

Destructor.

Definition at line 100 of file huv_generated_radial_dam_wet.cpp.

5.35.3 Member Function Documentation

void Huv_generated_Radial_Dam_wet::initialization (TAB & h, TAB & u, TAB & v) [virtual]

Performs the initialization.

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

Parameters

in, out	h	water height.
in, out	u	first component of the velocity.
in, out	v	second component of the velocity.

Implements [Initialization_huv](#).

Definition at line 79 of file huv_generated_radial_dam_wet.cpp.

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

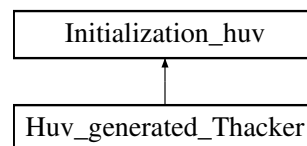
- Headers/libinitializations/[huv_generated_radial_dam_wet.hpp](#)
- Sources/libinitializations/[huv_generated_radial_dam_wet.cpp](#)

5.36 Huv_generated_Thacker Class Reference

Thacker configuration.

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

Inheritance diagram for Huv_generated_Thacker:



Public Member Functions

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

Additional Inherited Members

5.36.1 Detailed Description

Thacker configuration.

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

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

5.36.2 Constructor & Destructor Documentation

`Huv_generated_Thacker::Huv_generated_Thacker (Parameters & par)`

Constructor.

Defines the characteristics of the paraboloid.

Parameters

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

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

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

Destructor.

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

5.36.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Implements [Initialization_huv](#).

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

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

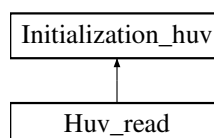
- [Headers/libinitializations/huv_generated_thacker.hpp](#)
- [Sources/libinitializations/huv_generated_thacker.cpp](#)

5.37 Huv_read Class Reference

File configuration.

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

Inheritance diagram for `Huv_read`:



Public Member Functions

- `Huv_read` (Parameters &)
Constructor.
- void `initialization` (TAB &, TAB &, TAB &)
Performs the initialization.
- virtual `~Huv_read` ()
Destructor.

Additional Inherited Members

5.37.1 Detailed Description

File configuration.

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

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

5.37.2 Constructor & Destructor Documentation

`Huv_read::Huv_read (Parameters & par)`

Constructor.

Defines the name of the file for the initialization.

Parameters

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

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

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

Destructor.

Definition at line 201 of file `huv_read.cpp`.

5.37.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Warning

(huv_namefile): ERROR: cannot open the huv file.

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

(huv_namefile): ERROR: line ***.

(huv_namefile): WARNING: line ***.

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

(huv_namefile): ERROR: the value for the point x *** y *** is missing

Note

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

Implements [Initialization_huv](#).

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

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

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

5.38 Hydrostatic Class Reference

Hydrostatic reconstruction

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

Public Member Functions

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

Protected Attributes

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

5.38.1 Detailed Description

Hydrostatic reconstruction

Class that computes the hydrostatic reconstruction.

Definition at line 67 of file `hydrostatic.hpp`.

5.38.2 Constructor & Destructor Documentation**Hydrostatic::Hydrostatic ()**

Constructor.

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

Hydrostatic::~~Hydrostatic () [virtual]

Destructor.

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

5.38.3 Member Function Documentation

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

Calculates the hydrostatic reconstruction.

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

Parameters

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

Modifies

`Hydrostatic::hl_rec`, set to $(hl - \max(0, dz))_+$.

`Hydrostatic::hr_rec`, set to $(hr - \max(0, -dz))_+$.

Definition at line 63 of file hydrostatic.cpp.

SCALAR Hydrostatic::get_hhydro_l ()

Gives the reconstructed water height on the left.

Returns

`Hydrostatic::hl_rec` Hydrostatic reconstruction on the left.

Definition at line 81 of file hydrostatic.cpp.

SCALAR Hydrostatic::get_hhydro_r ()

Gives the reconstructed water height on the right.

Returns

`Hydrostatic::hr_rec` Hydrostatic reconstruction on the right.

Definition at line 92 of file hydrostatic.cpp.

5.38.4 Member Data Documentation**SCALAR Hydrostatic::hl_rec [protected]**

Hydrostatic reconstruction on the left

Definition at line 87 of file hydrostatic.hpp.

SCALAR Hydrostatic::hr_rec [protected]

Hydrostatic reconstruction on the right

Definition at line 89 of file hydrostatic.hpp.

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

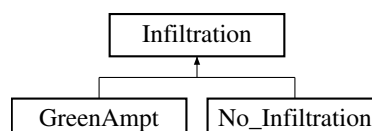
- Headers/libreconstructions/[hydrostatic.hpp](#)
- Sources/libreconstructions/[hydrostatic.cpp](#)

5.39 Infiltration Class Reference

Definition of infiltration law.

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

Inheritance diagram for Infiltration:



Public Member Functions

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

Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)
- [TAB](#) [hmod](#)
- [TAB](#) [Vin](#)

5.39.1 Detailed Description

Definition of infiltration law.

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

Definition at line 71 of file infiltration.hpp.

5.39.2 Constructor & Destructor Documentation

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

Constructor.

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

Parameters

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

Definition at line 60 of file infiltration.cpp.

Infiltration::~~Infiltration () [virtual]

Destructor.

Definition at line 103 of file infiltration.cpp.

5.39.3 Member Function Documentation

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

Function to be specified in each case.

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

TAB Infiltration::get_hmod () const

Gives the modified valued of the water height.

Returns

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

Definition at line 83 of file infiltration.cpp.

TAB Infiltration::get_Vin () const

Gives the infiltrated volume.

Returns

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

Definition at line 93 of file infiltration.cpp.

5.39.4 Member Data Documentation**const SCALAR Infiltration::DX [protected]**

Space step in the first (x) direction.

Definition at line 96 of file infiltration.hpp.

const SCALAR Infiltration::DY [protected]

Space step in the second (y) direction.

Definition at line 98 of file infiltration.hpp.

TAB Infiltration::hmod [protected]

Modified valued of the water height

Definition at line 100 of file infiltration.hpp.

const int Infiltration::NXCELL [protected]

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

Definition at line 92 of file infiltration.hpp.

const int Infiltration::NYCELL [protected]

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

Definition at line 94 of file infiltration.hpp.

TAB Infiltration::Vin [protected]

Infiltrated volume

Definition at line 102 of file infiltration.hpp.

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

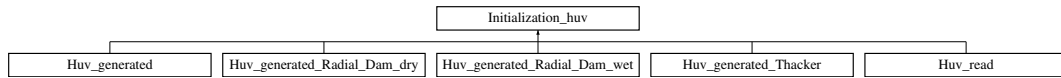
- [Headers/librain_infiltration/infiltration.hpp](#)
- [Sources/librain_infiltration/infiltration.cpp](#)

5.40 Initialization_huv Class Reference

Initialization of h, u and v.

```
#include <initialization_huv.hpp>
```

Inheritance diagram for Initialization_huv:



Public Member Functions

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

Protected Attributes

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

5.40.1 Detailed Description

Initialization of h, u and v.

Class that contains all the common declarations for the initialization of the water height and of the velocity.
Definition at line 71 of file initialization_huv.hpp.

5.40.2 Constructor & Destructor Documentation

Initialization_huv::Initialization_huv (Parameters & par)

Constructor.

Defines the numbers of cells and the space steps.

Parameters

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

Definition at line 59 of file initialization_huv.cpp.

Initialization_huv::~~Initialization_huv () [virtual]

Destructor.

Definition at line 70 of file initialization_huv.cpp.

5.40.3 Member Function Documentation

virtual void Initialization_huv::initialization (TAB &, TAB &, TAB &) [pure virtual]

Function to be specified in each initialization.

Implemented in [Huv_generated_Radial_Dam_wet](#), [Huv_generated_Thacker](#), [Huv_generated_Radial_Dam_dry](#), [Huv_generated](#), and [Huv_read](#).

5.40.4 Member Data Documentation

const SCALAR Initialization_huv::DX [protected]

Space step in the x direction.

Definition at line 90 of file initialization_huv.hpp.

const SCALAR Initialization_huv::DY [protected]

Space step in the y direction.

Definition at line 92 of file initialization_huv.hpp.

const int Initialization_huv::NXCELL [protected]

Number of cells in space in the x direction.

Definition at line 86 of file initialization_huv.hpp.

const int Initialization_huv::NYCELL [protected]

Number of cells in space in the y direction.

Definition at line 88 of file initialization_huv.hpp.

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

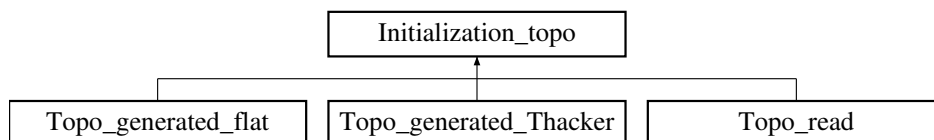
- Headers/libinitializations/[initialization_huv.hpp](#)
- Sources/libinitializations/[initialization_huv.cpp](#)

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

Protected Attributes

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

5.41.1 Detailed Description

Initialization of z.

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

Definition at line 71 of file initialization_topo.hpp.

5.41.2 Constructor & Destructor Documentation

Initialization_topo::Initialization_topo (Parameters & *par*)

Constructor.

Defines the numbers of cells and the space steps.

Parameters

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

Definition at line 59 of file initialization_topo.cpp.

Initialization_topo::~~Initialization_topo () [virtual]

Destructor.

Definition at line 70 of file initialization_topo.cpp.

5.41.3 Member Function Documentation

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

Function to be specified in each initialization.

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

5.41.4 Member Data Documentation

const SCALAR Initialization_topo::DX [protected]

Space step in the x direction.

Definition at line 91 of file initialization_topo.hpp.

const SCALAR Initialization_topo::DY [protected]

Space step in the y direction.

Definition at line 93 of file initialization_topo.hpp.

const int Initialization_topo::NXCELL [protected]

Number of cells in space in the x direction.

Definition at line 87 of file initialization_topo.hpp.

const int Initialization_topo::NYCELL [protected]

Number of cells in space in the y direction.

Definition at line 89 of file initialization_topo.hpp.

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

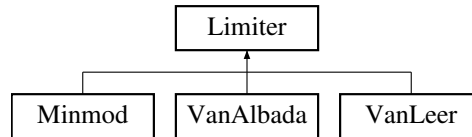
- Headers/libinitializations/[initialization_topo.hpp](#)
- Sources/libinitializations/[initialization_topo.cpp](#)

5.42 Limiter Class Reference

Slope limiter.

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

Inheritance diagram for Limiter:



Public Member Functions

- [Limiter \(\)](#)
Constructor.
- virtual void [calcul \(SCALAR, SCALAR\)=0](#)
Function to be specified in each slope limiter.
- [SCALAR get_rec \(\) const](#)
Gives the reconstructed value.
- virtual [~Limiter \(\)](#)
Destructor.

Protected Attributes

- [SCALAR rec](#)

5.42.1 Detailed Description

Slope limiter.

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

Definition at line 71 of file limiter.hpp.

5.42.2 Constructor & Destructor Documentation

Limiter::Limiter ()

Constructor.

Definition at line 59 of file limiter.cpp.

Limiter::~~Limiter () [virtual]

Destructor.

Definition at line 73 of file limiter.cpp.

5.42.3 Member Function Documentation

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

Function to be specified in each slope limiter.

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

SCALAR Limiter::get_rec () const

Gives the reconstructed value.

Returns

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

Definition at line 63 of file limiter.cpp.

5.42.4 Member Data Documentation**SCALAR Limiter::rec [protected]**

Reconstructed value

Definition at line 90 of file limiter.hpp.

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

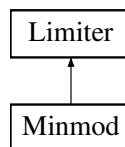
- Headers/liblimitations/[limiter.hpp](#)
- Sources/liblimitations/[limiter.cpp](#)

5.43 Minmod Class Reference

Minmod slope limiter

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

Inheritance diagram for Minmod:

**Public Member Functions**

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

Additional Inherited Members**5.43.1 Detailed Description**

Minmod slope limiter

Class that calculates the minmod slope limiter.

Definition at line 71 of file minmod.hpp.

5.43.2 Constructor & Destructor Documentation**Minmod::Minmod ()**

Constructor.

Definition at line 59 of file minmod.cpp.

Minmod::~Minmod () [virtual]

Destructor.

Definition at line 88 of file minmod.cpp.

5.43.3 Member Function Documentation

void Minmod::calcul (SCALAR a, SCALAR b) [virtual]

Calculates the value of the slope limiter.

Minmod function:

$$\text{minmod}(x,y) = \begin{cases} \min(x,y) & \text{if } x,y \geq 0, \\ \max(x,y) & \text{if } x,y \leq 0, \\ 0 & \text{else.} \end{cases}$$

Parameters

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

Modifies

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

Implements [Limiter](#).

Definition at line 62 of file minmod.cpp.

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

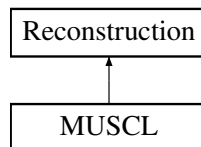
- Headers/liblimitations/[minmod.hpp](#)
- Sources/liblimitations/[minmod.cpp](#)

5.44 MUSCL Class Reference

MUSCL reconstruction

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

Inheritance diagram for MUSCL:



Public Member Functions

- [MUSCL \(Parameters &, TAB &\)](#)

Constructor.

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

Calculates the reconstruction in space.

- [~MUSCL \(\)](#)

Destructor.

Additional Inherited Members

5.44.1 Detailed Description

MUSCL reconstruction

Class that computes MUSCL reconstruction in space.

Definition at line 72 of file muscl.hpp.

5.44.2 Constructor & Destructor Documentation

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

Constructor.

Initializations.

Parameters

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

Definition at line 60 of file muscl.cpp.

MUSCL::~~MUSCL ()

Destructor.

Definition at line 225 of file muscl.cpp.

5.44.3 Member Function Documentation

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

Calculates the reconstruction in space.

Calls the calculation of the second order reconstruction in space with MUSCL formulation, see [van Leer \[1979\]](#) [Bouchut \[2007\]](#).

Parameters

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

out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>h2l</i>	reconstructed water height on the left of the cell in the second direction.
out	<i>u2l</i>	first component of the reconstructed velocity on the left of the cell in the second direction.
out	<i>v2l</i>	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 72 of file `muscl.cpp`.

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

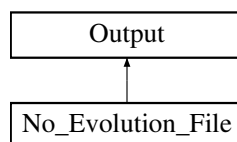
- `Headers/libreconstructions/muscl.hpp`
- `Sources/libreconstructions/muscl.cpp`

5.45 No_Evolution_File Class Reference

No output.

```
#include <no_evolution_file.hpp>
```

Inheritance diagram for `No_Evolution_File`:



Public Member Functions

- [No_Evolution_File](#) (`Parameters &`)
Constructor.
- `void write` (`const TAB &`, `const TAB &`, `const TAB &`, `const TAB &`, `const SCALAR &`)
Saves one time step: nothing to do.
- `virtual ~No_Evolution_File` (`()`)
Destructor.

Additional Inherited Members

5.45.1 Detailed Description

No output.

No output files with time evolution are created.

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

5.45.2 Constructor & Destructor Documentation

No_Evolution_File::No_Evolution_File (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 no_evolution_file.cpp.

No_Evolution_File::~No_Evolution_File () [virtual]

Destructor.

Definition at line 88 of file no_evolution_file.cpp.

5.45.3 Member Function Documentation**void No_Evolution_File::write (const TAB & *h*, const TAB & *u*, const TAB & *v*, const TAB & *z*, const SCALAR & *time*) [virtual]**

Saves one time step: nothing to do.

Does nothing.

Parameters

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

Implements [Output](#).

Definition at line 67 of file no_evolution_file.cpp.

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

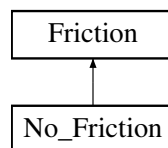
- Headers/libsave/[no_evolution_file.hpp](#)
- Sources/libsave/[no_evolution_file.cpp](#)

5.46 No_Friction Class Reference

No friction.

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

Inheritance diagram for No_Friction:

**Public Member Functions**

- [No_Friction](#) (Parameters &)
Constructor.
- void [calcul](#) (const TAB &, const TAB &, const TAB &, const TAB &, const TAB &, SCALAR)
Does no calculation.
- void [calculSf](#) (const TAB &, const TAB &, const TAB &)
Return the friction term equal to zero.
- virtual [~No_Friction](#) ()
Destructor.

Additional Inherited Members

5.46.1 Detailed Description

No friction.

Does no computation.

Definition at line 71 of file no_friction.hpp.

5.46.2 Constructor & Destructor Documentation

No_Friction::No_Friction (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file no_friction.cpp.

No_Friction::~~No_Friction () [virtual]

Destructor.

Definition at line 123 of file no_friction.cpp.

5.46.3 Member Function Documentation

void No_Friction::calcul (const TAB & *uold*, const TAB & *vold*, const TAB & *hnew*, const TAB & *q1new*, const TAB & *q2new*, SCALAR *dt*) [virtual]

Does no calculation.

No computation (no friction).

Parameters

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

Modifies

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

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

Implements [Friction](#).

Definition at line 68 of file no_friction.cpp.

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

Return the friction term equal to zero.

Explicit friction term: $S_f = 0$.

Parameters

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

Modifies

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

Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

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

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

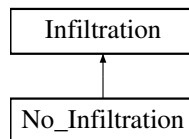
- [Headers/libfrictions/no_friction.hpp](#)
- [Sources/libfrictions/no_friction.cpp](#)

5.47 No_Infiltration Class Reference

No infiltration.

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

Inheritance diagram for `No_Infiltration`:

**Public Member Functions**

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

Additional Inherited Members**5.47.1 Detailed Description**

No infiltration.

The water height and infiltrated volume remain unchanged.

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

5.47.2 Constructor & Destructor Documentation

`No_Infiltration::No_Infiltration (Parameters & par)`

Constructor.

Parameters

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

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

No_Infiltration::~No_Infiltration () [virtual]

Destructor.

Definition at line 90 of file `no_infiltration.cpp`.

5.47.3 Member Function Documentation**void No_Infiltration::calcul (const TAB & h, const TAB & Vin_tot, const SCALAR dt) [virtual]**

Does no infiltration.

No computation (water height and infiltrated volume remain unchanged).

Parameters

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

Modifies

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

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

Implements [Infiltration](#).

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

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

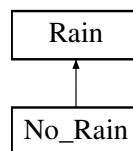
- [Headers/librain_infiltration/no_infiltration.hpp](#)
- [Sources/librain_infiltration/no_infiltration.cpp](#)

5.48 No_Rain Class Reference

No rain.

```
#include <no_rain.hpp>
```

Inheritance diagram for `No_Rain`:

**Public Member Functions**

- [No_Rain \(Parameters &\)](#)
Constructor.
- void [rain_func \(SCALAR, TAB &\)](#)
Sets the rain intensity to zero.
- virtual [~No_Rain \(\)](#)
Destructor.

Additional Inherited Members

5.48.1 Detailed Description

No rain.

Sets the rain intensity to zero.

Definition at line 70 of file no_rain.hpp.

5.48.2 Constructor & Destructor Documentation

No_Rain::No_Rain (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file no_rain.cpp.

No_Rain::~~No_Rain () [virtual]

Destructor.

Definition at line 85 of file no_rain.cpp.

5.48.3 Member Function Documentation

void No_Rain::rain_func (SCALAR *time*, TAB & *Tab_rain*) [virtual]

Sets the rain intensity to zero.

No computation (rain intensity set to zero).

Parameters

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

Implements [Rain](#).

Definition at line 68 of file no_rain.cpp.

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

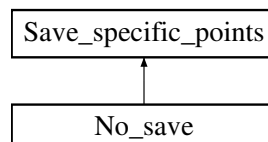
- [Headers/librain_infiltration/no_rain.hpp](#)
- [Sources/librain_infiltration/no_rain.cpp](#)

5.49 No_save Class Reference

No output.

```
#include <no_save.hpp>
```

Inheritance diagram for No_save:



Public Member Functions

- [No_save \(Parameters &\)](#)
Constructor.
- void [save](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))
Nothing to do.
- virtual [~No_save](#) ()
Destructor.

Additional Inherited Members

5.49.1 Detailed Description

No output.

No output files for specific points are created.

Definition at line 69 of file no_save.hpp.

5.49.2 Constructor & Destructor Documentation

No_save::No_save (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 no_save.cpp.

No_save::~~No_save () [virtual]

Destructor.

Definition at line 85 of file no_save.cpp.

5.49.3 Member Function Documentation

void No_save::save (const [TAB](#) & *h*, const [TAB](#) & *u*, const [TAB](#) & *v*, const [SCALAR](#) *time*) [virtual]

Nothing to do.

Does nothing.

Parameters

<i>in</i>	<i>h</i>	the water height (unused).
<i>in</i>	<i>u</i>	first component of the velocity (unused).
<i>in</i>	<i>v</i>	second component of the velocity (unused).
<i>in</i>	<i>time</i>	the current time (unused).

Implements [Save_specific_points](#).

Definition at line 67 of file no_save.cpp.

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

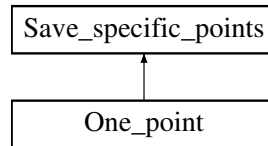
- Headers/libsave/[no_save.hpp](#)
- Sources/libsave/[no_save.cpp](#)

5.50 One_point Class Reference

One point output.

```
#include <one_point.hpp>
```

Inheritance diagram for One_point:



Public Member Functions

- `One_point (Parameters &)`
Constructor.
- `void save (const TAB &, const TAB &, const TAB &, const SCALAR)`
Saves the values at one point.
- `virtual ~One_point ()`
Destructor.

Additional Inherited Members

5.50.1 Detailed Description

One point output.

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

Definition at line 73 of file one_point.hpp.

5.50.2 Constructor & Destructor Documentation

One_point::One_point (Parameters & *par*)

Constructor.

Writes the header of the file 'hu_specific_points.dat'.

Parameters

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

Warning

***: ERROR: Impossible to open the *** file. Verify if the directory *** exists.

Note

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

Definition at line 60 of file one_point.cpp.

One_point::~~One_point () [virtual]

Destructor.

Definition at line 132 of file one_point.cpp.

5.50.3 Member Function Documentation

```
void One_point::save ( const TAB & h, const TAB & u, const TAB & v, const SCALAR time )
[virtual]
```

Saves the values at one point.

Writes the values of `Scheme::h`, `Scheme::u` ($=q1/h$), `Scheme::v` ($=q2/h$), `Scheme::h+Scheme::z` (free surface), `Scheme::z`, $|U| = \sqrt{u^2 + v^2}$, the Froude number $\frac{|U|}{\sqrt{gh}}$, `Scheme::q1`, `Scheme::q2`, and $h|U|$ at the current time in "hu_specific_points.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>u</i>	first component of the velocity.
in	<i>v</i>	second component of the velocity.
in	<i>time</i>	the current time.

Implements [Save_specific_points](#).

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

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

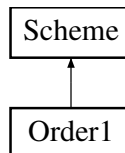
- Headers/libsave/[one_point.hpp](#)
- Sources/libsave/[one_point.cpp](#)

5.51 Order1 Class Reference

Order 1 scheme.

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

Inheritance diagram for Order1:



Public Member Functions

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

Additional Inherited Members

5.51.1 Detailed Description

Order 1 scheme.

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

Definition at line 71 of file `order1.hpp`.

5.51.2 Constructor & Destructor Documentation

Order1::Order1 (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file order1.cpp.

Order1::~~Order1 () [virtual]

Destructor.

Definition at line 257 of file order1.cpp.

5.51.3 Member Function Documentation**void Order1::calcul () [virtual]**

Performs the numerical scheme.

Performs the first order numerical scheme.

Note

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

Warning

order1: WARNING: the computation finished because the maximum number of time steps was reached (see MAX_ITER in [misc.hpp](#))

Implements [Scheme](#).

Definition at line 69 of file order1.cpp.

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

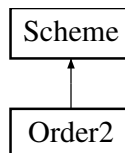
- [Headers/lib schemes/order1.hpp](#)
- [Sources/lib schemes/order1.cpp](#)

5.52 Order2 Class Reference

Order 2 scheme.

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

Inheritance diagram for Order2:

**Public Member Functions**

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

Additional Inherited Members

5.52.1 Detailed Description

Order 2 scheme.

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

Definition at line 71 of file order2.hpp.

5.52.2 Constructor & Destructor Documentation

Order2::Order2 (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 59 of file order2.cpp.

Order2::~~Order2 () [virtual]

Destructor.

Definition at line 97 of file order2.cpp.

5.52.3 Member Function Documentation

void Order2::calcul () [virtual]

Performs the numerical scheme.

Performs the second order numerical scheme.

Note

In DEBUG mode, the programme will save another file with volumes of water.

Warning

order2: WARNING: the computation finished because the maximum number of time steps was reached (see MAX_ITER in [misc.hpp](#))

Implements [Scheme](#).

Definition at line 124 of file order2.cpp.

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

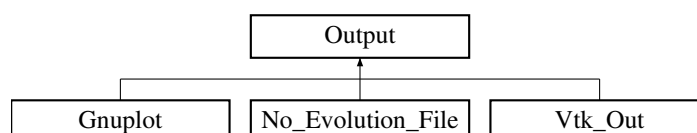
- [Headers/lib schemes/order2.hpp](#)
- [Sources/lib schemes/order2.cpp](#)

5.53 Output Class Reference

Output format

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

Inheritance diagram for Output:



Public Member Functions

- [Output](#) ([Parameters](#) &)
Constructor.
- virtual void [write](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#) &)=0
Function to be specified in each output format.
- void [check_vol](#) (const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &) const
Saves the infiltrated and rain volumes.
- void [result](#) (const [SCALAR](#) &, const clock_t &, const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const int &, const [SCALAR](#) &) const
Saves global values.
- void [initial](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &) const
Saves the initial time.
- void [final](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &) const
Saves the final time.
- [SCALAR boundaries_flux](#) (const [SCALAR](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#) &, const [SCALAR](#) &, const int &, const int &)
Saves the cumulated fluxes on the boundaries.
- void [boundaries_flux_LR](#) (const [SCALAR](#) &, const [TAB](#) &) const
Saves the fluxes on the left and right boundaries.
- void [boundaries_flux_BT](#) (const [SCALAR](#) &, const [TAB](#) &) const
Saves the fluxes on the bottom and top boundaries.
- virtual [~Output](#) ()
Destructor.

Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)
- string [outputDirectory](#)
- string [namefile_check_volume](#)
- string [namefile_res](#)
- string [namefile_init](#)
- string [namefile_final](#)
- string [namefile_Bound_flux](#)
- string [namefile_Bound_flux_BT](#)
- string [namefile_Bound_flux_LR](#)

5.53.1 Detailed Description

Output format

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

Definition at line 70 of file output.hpp.

5.53.2 Constructor & Destructor Documentation

Output::Output (Parameters & *par*)

Constructor.

Defines the names of the outputs.

If run in DEBUG mode, writes the header of the file 'boundaries_flux.dat', 'check_vol.dat', 'flux_boundaries_B←T.dat' and 'flux_boundaries_LR.dat'.

Parameters

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

Warning

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

Note

If 'boundaries_flux.dat', 'check_vol.dat', 'flux_boundaries_BT.dat' or 'flux_boundaries_LR.dat' cannot be opened, the code will exit with failure termination code.

Definition at line 59 of file output.cpp.

Output::~~Output () [virtual]

Destructor.

Definition at line 394 of file output.cpp.

5.53.3 Member Function Documentation

SCALAR Output::boundaries_flux (const SCALAR & *time*, const TAB & *flux_u*, const TAB & *flux_v*, const SCALAR & *dt*, const SCALAR & *dt_first*, const int & *ORDER*, const int & *verif*)

Saves the cumulated fluxes on the boundaries.

Parameters

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

Definition at line 274 of file output.cpp.

void Output::boundaries_flux_BT (const SCALAR & *time*, const TAB & *BT_flux*) const

Saves the fluxes on the bottom and top boundaries.

Parameters

<i>in</i>	<i>time</i>	current time.
-----------	-------------	---------------

in	<i>BT_flux</i>	flux on the bottom and tom boundaries (m ² /s).
----	----------------	--

Definition at line 335 of file output.cpp.

void Output::boundaries_flux_LR (const SCALAR & *time*, const TAB & *LR_flux*) const

Saves the fluxes on the left and right boundaries.

Parameters

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

Definition at line 317 of file output.cpp.

void Output::check_vol (const SCALAR & *time*, const SCALAR & *dt*, const SCALAR & *Vol_rain_tot*, const SCALAR & *Vol_inf*, const SCALAR & *Vol_of*, const SCALAR & *Vol_bound_tot*) const

Saves the infiltrated and rain volumes.

Parameters

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

Definition at line 201 of file output.cpp.

void Output::final (const TAB & *z*, const TAB & *h*, const TAB & *u*, const TAB & *v*) const

Saves the final time.

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

Parameters

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

Warning

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

Note

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

Definition at line 353 of file output.cpp.

void Output::initial (const TAB & *z*, const TAB & *h*, const TAB & *u*, const TAB & *v*) const

Saves the initial time.

Parameters

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

Warning

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

Note

If *huz_initial.dat* cannot be opened, the code will exit with failure termination code.

Definition at line 166 of file *output.cpp*.

```
void Output::result ( const SCALAR & time, const clock_t & cpu, const SCALAR & Vol_rain, const
SCALAR & Vol_inf, const SCALAR & Vol_of, const SCALAR & FROUDE, const int & NBITER, const
SCALAR & vol_output ) const
```

Saves global values.

Parameters

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

Warning

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

Note

If *results.dat* cannot be opened, the code will exit with failure termination code.

Definition at line 222 of file *output.cpp*.

```
virtual void Output::write ( const TAB & , const TAB & , const TAB & , const TAB & , const SCALAR
& ) [pure virtual]
```

Function to be specified in each output format.

Implemented in [Gnuplot](#), [Vtk_Out](#), and [No_Evolution_File](#).

5.53.4 Member Data Documentation

```
const SCALAR Output::DX [protected]
```

Space step in the first (x) direction.

Definition at line 110 of file *output.hpp*.

const SCALAR Output::DY [protected]

Space step in the second (y) direction.

Definition at line 112 of file output.hpp.

string Output::namefile_Bound_flux [protected]

Name of the file where the cumulated boundary fluxes are saved.

Definition at line 124 of file output.hpp.

string Output::namefile_Bound_flux_BT [protected]

Name of the file where the bottom and top boundary fluxes are saved.

Definition at line 126 of file output.hpp.

string Output::namefile_Bound_flux_LR [protected]

Name of the file where the left and right boundary fluxes are saved.

Definition at line 128 of file output.hpp.

string Output::namefile_check_volume [protected]

Name of the file where the verification of volumes is saved.

Definition at line 116 of file output.hpp.

string Output::namefile_final [protected]

Name of the file where the final time is saved.

Definition at line 122 of file output.hpp.

string Output::namefile_init [protected]

Name of the file where the initialization is saved.

Definition at line 120 of file output.hpp.

string Output::namefile_res [protected]

Name of the file where the global results are saved.

Definition at line 118 of file output.hpp.

const int Output::NXCELL [protected]

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

Definition at line 106 of file output.hpp.

const int Output::NYCELL [protected]

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

Definition at line 108 of file output.hpp.

string Output::outputDirectory [protected]

Name of the output directory.

Definition at line 114 of file output.hpp.

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

- Headers/libsave/output.hpp
- Sources/libsave/output.cpp

5.54 Parameters Class Reference

Gets parameters.

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

Public Member Functions

- [Parameters](#) ()
Constructor.
- void [setparameters](#) (const char *)
Sets the parameters.
- virtual [~Parameters](#) ()
Destructor.
- int [get_Nxcell](#) () const
Gives the number of cells in space along x.
- int [get_Nycell](#) () const
Gives the number of cells in space along y.
- [SCALAR get_T](#) () const
Gives the final time.
- int [get_nbtimes](#) () const
Gives the number of times saved.
- int [get_scheme_type](#) () const
Gives the choice of type of scheme (fixed cfl or fixed dt)
- [SCALAR get_dtfix](#) () const
Gives the fixed time step from the parameters.txt file.
- [SCALAR get_cflfix](#) () const
Gives the cfl of the scheme.
- [SCALAR get_dx](#) () const
Gives the space step along x.
- [SCALAR get_dy](#) () const
Gives the space step along y.
- [SCALAR get_L](#) () const
Gives the length of the domain in the first (x) direction.
- [SCALAR get_I](#) () const
Gives the length of the domain in the second (y) direction.
- map< int, int > & [get_Lbound](#) ()
Gives the value corresponding to the left boundary condition.
- map< [SCALAR](#), string > & [get_times_files_Lbound](#) ()
Gives the list of files and time values corresponding to the left boundary condition.
- int [get_type_Lbound](#) () const

- Gives the type (constant or file) of the left boundary condition.*

 - map< int, SCALAR > & `get_left_imp_discharge` ()
 - Gives the value of the imposed discharge in left bc.*
 - map< int, SCALAR > & `get_left_imp_h` ()
 - Gives the value of the imposed water height in left bc.*
 - map< int, int > & `get_Rbound` ()
 - Gives the value corresponding to the right boundary condition.*
 - map< SCALAR, string > & `get_times_files_Rbound` ()
 - Gives the list of files and time values corresponding to the right boundary condition.*
 - int `get_type_Rbound` () const
 - Gives the type (constant or file) of right boundary condition.*
 - map< int, SCALAR > & `get_right_imp_discharge` ()
 - Gives the value of the imposed discharge in right bc.*
 - map< int, SCALAR > & `get_right_imp_h` ()
 - Gives the value of the imposed water height in right bc.*
 - map< int, int > & `get_Bbound` ()
 - Gives the value corresponding to the bottom boundary condition.*
 - map< SCALAR, string > & `get_times_files_Bbound` ()
 - Gives the list of files and time values corresponding to the bottom boundary condition.*
 - int `get_type_Bbound` () const
 - Gives the type (constant or file) of bottom boundary condition.*
 - map< int, SCALAR > & `get_bottom_imp_discharge` ()
 - Gives the value of the imposed discharge in bottom bc.*
 - map< int, SCALAR > & `get_bottom_imp_h` ()
 - Gives the value of the imposed water height in bottom bc.*
 - map< int, int > & `get_Tbound` ()
 - Gives the value corresponding to the top boundary condition.*
 - map< SCALAR, string > & `get_times_files_Tbound` ()
 - Gives the list of files and time values corresponding to the top boundary condition.*
 - int `get_type_Tbound` () const
 - Gives the type (constant or file) of bottom boundary condition.*
 - map< int, SCALAR > & `get_top_imp_discharge` ()
 - Gives the value of the imposed discharge in top bc.*
 - map< int, SCALAR > & `get_top_imp_h` ()
 - Gives the value of the imposed water height in top bc.*
 - int `get_flux` () const
 - Gives the value corresponding to the flux.*
 - int `get_order` () const
 - Gives the order of the scheme.*
 - int `get_rec` () const
 - Gives the value corresponding to the reconstruction.*
 - int `get_fric` () const
 - Gives the value corresponding to the friction law.*
 - int `get_lim` () const
 - Gives the value corresponding to the limiter.*
 - int `get_inf` () const
 - Gives the choice of infiltration model.*

- [SCALAR get_amortENO](#) () const
Gives the value of the amortENO parameter.
- [SCALAR get_modifENO](#) () const
Gives the value of the modifENO parameter.
- [SCALAR get_friccoef](#) () const
Gives the value of the friction coefficient.
- [int get_fric_init](#) () const
Gives the value characterizing the spatialization (or not) of the friction coefficient.
- [string get_KcNameFile](#) (void) const
Gives the full name of the file containing the hydraulic conductivity of the crust.
- [string get_KcNameFileS](#) () const
Gives the name of the file containing the hydraulic conductivity of the crust.
- [string get_KsNameFile](#) (void) const
Gives the full name of the file containing the hydraulic conductivity of the surface.
- [string get_KsNameFileS](#) () const
Gives the name of the file containing the hydraulic conductivity of the surface.
- [string get_dthetaNameFile](#) (void) const
Gives the full name of the file containing the initial water deficit.
- [string get_dthetaNameFileS](#) () const
Gives the name of the file containing the initial water deficit.
- [string get_PsiNameFile](#) (void) const
Gives the full name of the file containing the load pressure.
- [string get_PsiNameFileS](#) () const
Gives the name of the file containing the load pressure.
- [string get_zcrustNameFile](#) (void) const
Gives the full name of the file containing the thickness of the crust.
- [string get_zcrustNameFileS](#) () const
Gives the name of the file containing the thickness of the crust.
- [string get_imaxNameFile](#) (void) const
Gives the full name of the file containing the maximum infiltration rate.
- [string get_imaxNameFileS](#) () const
Gives the name of the file containing the maximum infiltration rate.
- [int get_Kc_init](#) () const
Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the crust.
- [SCALAR get_Kc_coef](#) () const
Gives the value of the hydraulic conductivity of the crust.
- [int get_Ks_init](#) () const
Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the soil.
- [SCALAR get_Ks_coef](#) () const
Gives the value of the hydraulic conductivity of the soil.
- [int get_dtheta_init](#) () const
Gives the value characterizing the spatialization (or not) of the initial water deficit.
- [SCALAR get_dtheta_coef](#) () const
Gives the value of the initial water deficit.
- [int get_Psi_init](#) () const
Gives the value characterizing the spatialization (or not) of the load pressure.
- [SCALAR get_Psi_coef](#) () const

- Gives the value of the load pressure.*
- int `get_zcrust_init` () const

Gives the value characterizing the spatialization (or not) of the thickness of the crust.
 - SCALAR `get_zcrust_coef` () const

Gives the value of the thickness of the crust.
 - int `get_imax_init` () const

Gives the value characterizing the spatialization (or not) of the maximum infiltration rate.
 - SCALAR `get_imax_coef` () const

Gives the value of the maximum infiltration rate.
 - int `get_topo` () const

Gives the value corresponding to the choice of topography.
 - int `get_huv` () const

Gives the value corresponding to the choice of initialization of h, u and v.
 - int `get_rain` () const

Gives the value corresponding to the choice of initialization of rain.
 - string `get_topographyNameFile` (void) const

Gives the full name of the file containing the topography.
 - string `get_topographyNameFileS` () const

Gives the name of the file containing the topography.
 - string `get_huvNameFile` (void) const

Gives the full name of the file containing the water height (h) and the velocities (u and v)
 - string `get_huvNameFileS` (void) const

Gives the name of the file containing the water height (h) and the velocities (u and v)
 - string `get_rainNameFile` (void) const

Gives the full name of the file containing the rain.
 - string `get_rainNameFileS` (void) const

Gives the name of the file containing the rain.
 - string `get_frictionNameFile` (void) const

Gives the full name of the file containing the friction coefficient.
 - string `get_frictionNameFileS` (void) const

Gives the name of the file containing the friction coefficient.
 - string `get_outputDirectory` (void) const

Gives the output directory with the suffix.
 - string `get_suffix` (void) const

Gives the suffix for the 'Outputs' directory.
 - int `get_output` () const

Gives the value corresponding to the choice of the format of the [Output](#) file.
 - int `get_choice_specific_point` () const

Gives the choice of specific points to be saved.
 - int `get_choice_dt_specific_points` () const

Gives the choice of time step for specific points to be saved.
 - SCALAR `get_dt_specific_points` () const

Gives time step for specific points to be saved.
 - void `fill_array` (TAB &, const SCALAR) const

Fills the TAB array with a SCALAR.
 - void `fill_array` (TAB &, string) const

Fills the TAB array with the values contained in a file.

- bool `is_coord_in_file_valid` (const `SCALAR` &x, const `SCALAR` &y, int num_lin, string namefile) const
Verifies if the coordinates x and y exist in the mesh.
- `SCALAR` `get_x_coord` (void) const
Gives x coordinate of the specific point to be saved.
- `SCALAR` `get_y_coord` (void) const
Gives y coordinate of the specific point to be saved.
- string `get_list_pointNameFile` (void) const
Gives the full name of file containing the list of the specific points.
- string `get_list_pointNameFileS` (void) const
Gives the name of the file containing the list of the specific points.
- map< int, int > `fill_array_bc_inhomogeneous` (string, string, char, map< int, `SCALAR` > &, map< int, `SCALAR` > &)
Returns the vector containing the choice of boundary conditions. The two containers will contain the values of discharge and water heights imposed for each point.
- map< `SCALAR`, string > `verif_file_bc_inhomogeneous` (string, string, char, map< int, int > &, map< int, `SCALAR` > &, map< int, `SCALAR` > &)
Returns the list of files and time values corresponding to the boundary condition. Moreover, fills the containers that will contain the choice of boundary conditions, the values of discharge and water heights imposed for each point.
- string `get_path_input_directory` (void) const

Protected Attributes

- `SCALAR` `cfl_fix`
- `SCALAR` `dt_fix`
- int `scheme_type`
- int `Nxcell`
- int `Nycell`
- int `nbtimes`
- `SCALAR` `T`
- `SCALAR` `dx`
- `SCALAR` `dy`
- `SCALAR` `L`
- `SCALAR` `I`
- map< int, int > `Lbound`
- map< int, `SCALAR` > `left_imp_discharge`
- map< int, `SCALAR` > `left_imp_h`
- map< `SCALAR`, string > `left_times_files`
- int `Lbound_type`
- map< int, int > `Rbound`
- map< int, `SCALAR` > `right_imp_discharge`
- map< int, `SCALAR` > `right_imp_h`
- map< `SCALAR`, string > `right_times_files`
- int `Rbound_type`
- map< int, int > `Bbound`
- map< int, `SCALAR` > `bottom_imp_discharge`
- map< int, `SCALAR` > `bottom_imp_h`
- map< `SCALAR`, string > `bottom_times_files`
- int `Bbound_type`

- `map< int, int > Tbound`
- `map< int, SCALAR > top_imp_discharge`
- `map< int, SCALAR > top_imp_h`
- `map< SCALAR, string > top_times_files`
- `int Tbound_type`
- `int flux`
- `int order`
- `int rec`
- `int fric`
- `int lim`
- `int inf`
- `int topo`
- `int huv_init`
- `int rain`
- `int Kc_init`
- `int Ks_init`
- `int dtheta_init`
- `int Psi_init`
- `int zcrust_init`
- `int imax_init`
- `int output_format`
- `SCALAR amortENO`
- `SCALAR modifENO`
- `int fric_init`
- `SCALAR friccoef`
- `SCALAR Kc_coef`
- `SCALAR Ks_coef`
- `SCALAR dtheta_coef`
- `SCALAR Psi_coef`
- `SCALAR zcrust_coef`
- `SCALAR imax_coef`
- `string topography_namefile`
- `string topo_NF`
- `string huv_namefile`
- `string huv_NF`
- `string fric_namefile`
- `string fric_NF`
- `string rain_namefile`
- `string rain_NF`
- `string Kc_namefile`
- `string Kc_NF`
- `string Ks_namefile`
- `string Ks_NF`
- `string dtheta_namefile`
- `string dtheta_NF`
- `string Psi_namefile`
- `string Psi_NF`
- `string zcrust_namefile`
- `string zcrust_NF`

- string `imax_namefile`
- string `imax_NF`
- string `output_directory`
- string `suffix_outputs`
- int `Choice_points`
- SCALAR `x_coord`
- SCALAR `y_coord`
- string `list_point_NF`
- string `list_point_namefile`
- VECT `list_points_x`
- VECT `list_points_y`
- int `Choice_dt_specific_points`
- SCALAR `dt_specific_points`
- string `path_input_directory`

5.54.1 Detailed Description

Gets parameters.

Class that reads the parameters, checks their values and contains all the common declarations to get the values of the parameters.

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

5.54.2 Constructor & Destructor Documentation

Parameters::Parameters ()

Constructor.

Definition at line 3189 of file `parameters.cpp`.

Parameters::~~Parameters () [virtual]

Destructor.

Definition at line 3192 of file `parameters.cpp`.

5.54.3 Member Function Documentation

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

Fills the TAB array with a SCALAR.

Fills an array with a constant value.

Parameters

<code>in, out</code>	<code><i>myarray</i></code>	array to fill.
<code>in</code>	<code><i>myvalue</i></code>	value.

Definition at line 2610 of file `parameters.cpp`.

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

Fills the TAB array with the values contained in a file.

Fills an array with the values given in the file

Parameters

in, out	<i>myarray</i>	array to fill.
in	<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 2624 of file parameters.cpp.

map< int, int > Parameters::fill_array_bc_inhomogeneous (string *Bc_NF*, string *path_input_directory*, char *BC*, map< int, SCALAR > & *imp_q*, map< int, SCALAR > & *imp_h*)

Returns the vector containing the choice of boundary conditions. The two containers will contain the values of discharge and water heights imposed for each point.

Extracts from the *Bc_NF* file the type of boundary condition, discharge and water heights imposed for each point.

Parameters

in	<i>Bc_NF</i>	name of the file containing the values to be verified.
in	<i>path_input_↔ directory</i>	path of directory containing <i>Bc_NF</i> file.
in	<i>BC</i>	represents the boundary condition which we handle
out	<i>container</i>	containing discharges [m ³ /s]
out	<i>container</i>	containing water heights [m]

Warning

***: ERROR: cannot open the file.
 ***: ERROR: the number of data in this file is too big/small.
 ***: ERROR: the value of data in this file is too big/small.
 ***: ERROR: the value for the point x = *** is missing.
 ***: ERROR: the values q and h for the point x = *** are missing.
 ***: ERROR: line ***.

Returns

the vector containing the choice of boundary conditions.

Note

If the containers cannot be filled correctly, the code will exit with failure termination code.

Definition at line 2893 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 2178 of file parameters.cpp.

map< int, int > & Parameters::get_Bbound ()

Gives the value corresponding to the bottom boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 2034 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_bottom_imp_discharge ()

Gives the value of the imposed discharge in bottom bc.

Returns

The value of the imposed discharge per cell in the bottom boundary condition, that is [Parameters::bottom↔_imp_discharge / Parameters::L](#).

Definition at line 2057 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_bottom_imp_h ()

Gives the value of the imposed water height in bottom bc.

Returns

The value of the imposed water height in the bottom boundary condition [Parameters::bottom_imp_h](#).

Definition at line 2067 of file parameters.cpp.

SCALAR Parameters::get_cflfix () const

Gives the cfl of the scheme.

Returns

The fixed cfl [Parameters::cfl_fix](#).

Definition at line 1874 of file parameters.cpp.

int Parameters::get_choice_dt_specific_points () const

Gives the choice of time step for specific points to be saved.

Returns

The choice of times saved for the specific points.

Definition at line 2873 of file parameters.cpp.

int Parameters::get_choice_specific_point () const

Gives the choice of specific points to be saved.

Returns

The choice for saving specific points.

Definition at line 2863 of file parameters.cpp.

SCALAR Parameters::get_dt_specific_points () const

Gives time step for specific points to be saved.

Returns

The time step for saving specific points.

Definition at line 2883 of file parameters.cpp.

SCALAR Parameters::get_dtfix () const

Gives the fixed time step from the parameters.txt file.

Returns

The fixed space step Parameters::dx_fix.

Definition at line 1864 of file parameters.cpp.

SCALAR Parameters::get_dtheta_coef () const

Gives the value of the initial water deficit.

Returns

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

Definition at line 2438 of file parameters.cpp.

int Parameters::get_dtheta_init () const

Gives the value characterizing the spatialization (or not) of the initial water deficit.

Returns

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

Definition at line 2428 of file parameters.cpp.

string Parameters::get_dthetaNameFile (void) const

Gives the full name of the file containing the initial water deficit.

Returns

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

Definition at line 2448 of file parameters.cpp.

string Parameters::get_dthetaNameFileS () const

Gives the name of the file containing the initial water deficit.

Returns

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

Definition at line 2458 of file parameters.cpp.

SCALAR Parameters::get_dx () const

Gives the space step along x.

Returns

The space step in the first (x) direction [Parameters::dx](#).

Definition at line 1884 of file parameters.cpp.

SCALAR Parameters::get_dy () const

Gives the space step along y.

Returns

The space step in the second (y) direction [Parameters::dy](#).

Definition at line 1894 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 2127 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 2157 of file parameters.cpp.

int Parameters::get_fric_init () const

Gives the value characterizing the spatialization (or not) of the friction coefficient.

Returns

The value corresponding to the friction coefficient [Parameters::fric_init](#).

Definition at line 2208 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 2238 of file parameters.cpp.

string Parameters::get_frictionNameFile (void) const

Gives the full name of the file containing the friction coefficient.

Returns

The friction coefficient path + Input directory [Parameters::fric_namefile](#).

Definition at line 2218 of file parameters.cpp.

string Parameters::get_frictionNameFileS (void) const

Gives the name of the file containing the friction coefficient.

Returns

The friction coefficient namefile (inside the Input directory) [Parameters::fric_NF](#).

Definition at line 2228 of file parameters.cpp.

int Parameters::get_huv () const

Gives the value corresponding to the choice of initialization of h, u and v.

Returns

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

Definition at line 2278 of file parameters.cpp.

string Parameters::get_huvNameFile (void) const

Gives the full name of the file containing the water height (h) and the velocities (u and v)

Returns

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

Definition at line 2288 of file parameters.cpp.

string Parameters::get_huvNameFileS (void) const

Gives the name of the file containing the water height (h) and the velocities (u and v)

Returns

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

Definition at line 2298 of file parameters.cpp.

SCALAR Parameters::get_imax_coef () const

Gives the value of the maximum infiltration rate.

Returns

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

Definition at line 2558 of file parameters.cpp.

int Parameters::get_imax_init () const

Gives the value characterizing the spatialization (or not) of the maximum infiltration rate.

Returns

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

Definition at line 2548 of file parameters.cpp.

string Parameters::get_imaxNameFile (void) const

Gives the full name of the file containing the maximum infiltration rate.

Returns

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

Definition at line 2568 of file parameters.cpp.

string Parameters::get_imaxNameFileS () const

Gives the name of the file containing the maximum infiltration rate.

Returns

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

Definition at line 2578 of file parameters.cpp.

int Parameters::get_inf () const

Gives the choice of infiltration model.

Returns

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

Definition at line 2198 of file parameters.cpp.

SCALAR Parameters::get_Kc_coef () const

Gives the value of the hydraulic conductivity of the crust.

Returns

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

Definition at line 2358 of file parameters.cpp.

int Parameters::get_Kc_init () const

Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the crust.

Returns

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

Definition at line 2348 of file parameters.cpp.

string Parameters::get_KcNameFile (void) const

Gives the full name of the file containing the hydraulic conductivity of the crust.

Returns

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

Definition at line 2368 of file parameters.cpp.

string Parameters::get_KcNameFileS () const

Gives the name of the file containing the hydraulic conductivity of the crust.

Returns

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

Definition at line 2378 of file parameters.cpp.

SCALAR Parameters::get_Ks_coef () const

Gives the value of the hydraulic conductivity of the soil.

Returns

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

Definition at line 2398 of file parameters.cpp.

int Parameters::get_Ks_init () const

Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the soil.

Returns

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

Definition at line 2388 of file parameters.cpp.

string Parameters::get_KsNameFile (void) const

Gives the full name of the file containing the hydraulic conductivity of the surface.

Returns

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

Definition at line 2408 of file parameters.cpp.

string Parameters::get_KsNameFileS () const

Gives the name of the file containing the hydraulic conductivity of the surface.

Returns

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

Definition at line 2418 of file parameters.cpp.

SCALAR Parameters::get_L () const

Gives the length of the domain in the first (x) direction.

Returns

the length of the domain in the first (x) direction.

Definition at line 1904 of file parameters.cpp.

SCALAR Parameters::get_I () const

Gives the length of the domain in the second (y) direction.

Returns

the length of the domain in the second (y) direction.

Definition at line 1914 of file parameters.cpp.

map< int, int > & Parameters::get_Lbound ()

Gives the value corresponding to the left boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 1935 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_left_imp_discharge ()

Gives the value of the imposed discharge in left bc.

Returns

The value of the imposed discharge per cell in the left boundary condition, that is [Parameters::left_imp_discharge](#) / [Parameters::l](#).

Definition at line 1955 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_left_imp_h ()

Gives the value of the imposed water height in left bc.

Returns

The value of the imposed water height in the left boundary condition [Parameters::left_imp_h](#).

Definition at line 1966 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 2168 of file parameters.cpp.

string Parameters::get_list_pointNameFile (void) const

Gives the full name of file containing the list of the specific points.

Returns

The name of file containing the list of the specific points path for the initialization + Input directory [Parameters::list_point_namefile](#).

Definition at line 2844 of file parameters.cpp.

string Parameters::get_list_pointNameFileS (void) const

Gives the name of the file containing the list of the specific points.

Returns

The name of file containing list of the specific points for the initialization (inside the Input directory) [Parameters::list_point_NF](#).

Definition at line 2853 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 2188 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 1844 of file parameters.cpp.

int Parameters::get_Nxcell () const

Gives the number of cells in space along x.

Returns

The number of cells in space in the first (x) direction [Parameters::Nxcell](#).

Definition at line 1814 of file parameters.cpp.

int Parameters::get_Nycell () const

Gives the number of cells in space along y.

Returns

The number of cells in space in the second (y) direction [Parameters::Nycell](#).

Definition at line 1824 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 2137 of file parameters.cpp.

int Parameters::get_output () const

Gives the value corresponding to the choice of the format of the [Output](#) file.

Returns

The type of output [Parameters::output_format](#).

Definition at line 2598 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 2338 of file parameters.cpp.

string Parameters::get_path_input_directory (void) const

Returns

The Name of the input directory.

Definition at line 3177 of file parameters.cpp.

SCALAR Parameters::get_Psi_coef () const

Gives the value of the load pressure.

Returns

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

Definition at line 2478 of file parameters.cpp.

int Parameters::get_Psi_init () const

Gives the value characterizing the spatialization (or not) of the load pressure.

Returns

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

Definition at line 2468 of file parameters.cpp.

string Parameters::get_PsiNameFile (void) const

Gives the full name of the file containing the load pressure.

Returns

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

Definition at line 2488 of file parameters.cpp.

string Parameters::get_PsiNameFileS () const

Gives the name of the file containing the load pressure.

Returns

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

Definition at line 2498 of file parameters.cpp.

int Parameters::get_rain () const

Gives the value corresponding to the choice of initialization of rain.

Returns

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

Definition at line 2308 of file parameters.cpp.

string Parameters::get_rainNameFile (void) const

Gives the full name of the file containing the rain.

Returns

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

Definition at line 2318 of file parameters.cpp.

string Parameters::get_rainNameFileS (void) const

Gives the name of the file containing the rain.

Returns

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

Definition at line 2328 of file parameters.cpp.

map< int, int > & Parameters::get_Rbound ()

Gives the value corresponding to the right boundary condition.

Returns

The value corresponding to the right boundary condition [Parameters::Rbound](#).

Definition at line 1984 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 2147 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_right_imp_discharge ()

Gives the value of the imposed discharge in right bc.

Returns

The value of the imposed discharge per cell in the right boundary condition, that is [Parameters::right_imp_discharge](#) / [Parameters::l](#).

Definition at line 2004 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_right_imp_h ()

Gives the value of the imposed water height in right bc.

Returns

The value of the imposed water height in the right boundary condition [Parameters::right_imp_h](#).

Definition at line 2014 of file parameters.cpp.

int Parameters::get_scheme_type () const

Gives the choice of type of scheme (fixed cfl or fixed dt)

Returns

The type of scheme [Parameters::scheme_type](#).

Definition at line 1854 of file parameters.cpp.

string Parameters::get_suffix (void) const

Gives the suffix for the 'Outputs' directory.

Returns

The suffix (for the output directory) [Parameters::suffix_outputs](#).

Definition at line 2588 of file parameters.cpp.

SCALAR Parameters::get_T () const

Gives the final time.

Returns

The final time [Parameters::T](#).

Definition at line 1834 of file parameters.cpp.

map< int, int > & Parameters::get_Tbound ()

Gives the value corresponding to the top boundary condition.

Returns

The value corresponding to the top boundary condition [Parameters::Tbound](#).

Definition at line 2087 of file parameters.cpp.

map< SCALAR, string > & Parameters::get_times_files_Bbound ()

Gives the list of files and time values corresponding to the bottom boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 2044 of file parameters.cpp.

map< SCALAR, string > & Parameters::get_times_files_Lbound ()

Gives the list of files and time values corresponding to the left boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 1945 of file parameters.cpp.

map< SCALAR, string > & Parameters::get_times_files_Rbound ()

Gives the list of files and time values corresponding to the right boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 1994 of file parameters.cpp.

map< SCALAR, string > & Parameters::get_times_files_Tbound ()

Gives the list of files and time values corresponding to the top boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 2097 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_top_imp_discharge ()

Gives the value of the imposed discharge in top bc.

Returns

The value of the imposed discharge per cell in the top boundary condition, that is [Parameters::top_imp_discharge](#) / [Parameters::L](#).

Definition at line 2108 of file parameters.cpp.

map< int, SCALAR > & Parameters::get_top_imp_h ()

Gives the value of the imposed water height in top bc.

Returns

The value of the imposed water height in the bottom boundary condition [Parameters::top_imp_h](#).

Definition at line 2118 of file parameters.cpp.

int Parameters::get_topo () const

Gives the value corresponding to the choice of topography.

Returns

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

Definition at line 2268 of file parameters.cpp.

string Parameters::get_topographyNameFile (void) const

Gives the full name of the file containing the topography.

Returns

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

Definition at line 2248 of file parameters.cpp.

string Parameters::get_topographyNameFileS (void) const

Gives the name of the file containing the topography.

Returns

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

Definition at line 2258 of file parameters.cpp.

int Parameters::get_type_Bbound () const

Gives the type (constant or file) of bottom boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 2023 of file parameters.cpp.

int Parameters::get_type_Lbound () const

Gives the type (constant or file) of the left boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 1924 of file parameters.cpp.

int Parameters::get_type_Rbound () const

Gives the type (constant or file) of right boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 1975 of file parameters.cpp.

int Parameters::get_type_Tbound () const

Gives the type (constant or file) of bottom boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 2076 of file parameters.cpp.

SCALAR Parameters::get_x_coord (void) const

Gives x coordinate of the specific point to be saved.

Returns

x coordinate of the specific point to be saved.

Definition at line 2824 of file parameters.cpp.

SCALAR Parameters::get_y_coord (void) const

Gives y coordinate of the specific point to be saved.

Returns

y coordinate of the specific point to be saved.

Definition at line 2834 of file parameters.cpp.

SCALAR Parameters::get_zcrust_coef () const

Gives the value of the thickness of the crust.

Returns

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

Definition at line 2518 of file parameters.cpp.

int Parameters::get_zcrust_init () const

Gives the value characterizing the spatialization (or not) of the thickness of the crust.

Returns

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

Definition at line 2508 of file parameters.cpp.

string Parameters::get_zcrustNameFile (void) const

Gives the full name of the file containing the thickness of the crust.

Returns

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

Definition at line 2528 of file parameters.cpp.

string Parameters::get_zcrustNameFileS () const

Gives the name of the file containing the thickness of the crust.

Returns

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

Definition at line 2538 of file parameters.cpp.

bool Parameters::is_coord_in_file_valid (const SCALAR & x, const SCALAR & y, int num_lin, string namefile) const

Verifies if the coordinates x and y exist in the mesh.

Fills an array with the values given in the file

Parameters

in	<i>x</i>	coordinate of the specific point.
in	<i>y</i>	coordinate of the specific point.
in	<i>line_number</i>	the line number where the error appears.
in	<i>namefile</i>	name of the file containing the values to be verified.

Warning

***: ERROR: .
 ***: ERROR: the value of data in this file is too big/small.
 ***: ERROR: line ***.

Note

-1 is used to not display the line number in the error message .
 if the coordinates are not valid, the code will return with false.

Definition at line 2739 of file parameters.cpp.

void Parameters::setparameters (const char * FILENAME)

Sets the parameters.

Gets all the parameters from the file FILENAME, check and affect them. The values used by FullSWOF_2D are saved in the file parameters.dat. These values are also printed in the terminal when the code is run.

Parameters

<i>in</i>	<i>FILENAME</i>	name of the paramters file.
-----------	-----------------	-----------------------------

Warning

parameters.txt: ERROR: ***.
 parameters.txt: WARNING: ***.
 ERROR: the *** file *** does not exists in the directory Inputs.
 Impossible to open the *** file. Verify if the directory *** exists.

Note

If a value cannot be affected correctly, the code will exit with failure termination code.
 If parameters.dat cannot be opened, the code will exit with failure termination code.

Definition at line 61 of file parameters.cpp.

map< double, string > Parameters::verif_file_bc_inhomogeneous (string *Bc_NF*, string *path_input_directory*, char *BC*, map< int, int > & *vchoice*, map< int, SCALAR > & *imp_q*, map< int, SCALAR > & *imp_h*)

Returns the list of files and time values corresponding to the boundary condition. Moreover, fills the containers that will contain the choice of boundary conditions, the values of discharge and water heights imposed for each point.

Extracts from the *Bc_NF* file the list of files and time values corresponding to the boundary condition.

Parameters

<i>in</i>	<i>Bc_NF</i>	name of the file containing the values to be verified.
<i>in</i>	<i>path_input_directory</i>	path of directory containing <i>Bc_NF</i> file.
<i>in</i>	<i>BC</i>	represents the boundary condition which we handle
<i>out</i>	<i>container</i>	containing the choice of boundary conditions at time equal to 0s.
<i>out</i>	<i>container</i>	containing discharges [m3/s] at time equal to 0s.
<i>out</i>	<i>container</i>	containing water heights [m] at time equal to 0s.

Warning

***: ERROR: cannot open the file.
 ***: ERROR: the number of data in this file is too big/small.
 ***: ERROR: the value of data in this file is too big/small.
 ***: ERROR: the value for the point x = *** is missing.
 ***: ERROR: the values q and h for the point x = *** are missing.
 ***: ERROR: line ***.
 ***: ERROR: the times are decreasing.
 (rain_namefile): ERROR: the first time must be t = 0.

Returns

the list of files and time values corresponding to the boundary condition.

Note

If the containers cannot be filled correctly, the code will exit with failure termination code.

Definition at line 3055 of file parameters.cpp.

5.54.4 Member Data Documentation

SCALAR Parameters::amortENO [protected]

Parameter for eno.

Definition at line 463 of file parameters.hpp.

map<int,int> Parameters::Bbound [protected]

Bottom boundary condition.

Definition at line 411 of file parameters.hpp.

int Parameters::Bbound_type [protected]

Type (constant or file) of bottom boundary condition

Definition at line 419 of file parameters.hpp.

map<int,SCALAR> Parameters::bottom_imp_discharge [protected]

Imposed discharge on the bottom boundary.

Definition at line 413 of file parameters.hpp.

map<int,SCALAR> Parameters::bottom_imp_h [protected]

Imposed water height on the bottom boundary.

Definition at line 415 of file parameters.hpp.

map<SCALAR,string> Parameters::bottom_times_files [protected]

List of files and time values corresponding to the bottom boundary condition.

Definition at line 417 of file parameters.hpp.

SCALAR Parameters::cfl_fix [protected]

Value of the fixed cfl.

Definition at line 369 of file parameters.hpp.

int Parameters::Choice_dt_specific_points [protected]

Choice between saving specific points at each time or only for a given time step.

Definition at line 539 of file parameters.hpp.

int Parameters::Choice_points [protected]

Choice of the number of specific points to be saved.

Definition at line 527 of file parameters.hpp.

SCALAR Parameters::dt_fix [protected]

Value of the fixed time step.

Definition at line 371 of file parameters.hpp.

SCALAR Parameters::dt_specific_points [protected]

Time step to save the list of the specific points.

Definition at line 541 of file parameters.hpp.

SCALAR Parameters::dtheta_coef [protected]

Value of dtheta.

Definition at line 475 of file parameters.hpp.

int Parameters::dtheta_init [protected]

Type of initialization of dtheta.

Definition at line 453 of file parameters.hpp.

string Parameters::dtheta_namefile [protected]

Name of the file for dtheta: Inputs/file.

Definition at line 507 of file parameters.hpp.

string Parameters::dtheta_NF [protected]

Name of the file for dtheta without 'Inputs'.

Definition at line 509 of file parameters.hpp.

SCALAR Parameters::dx [protected]

Space step in the first (x) direction.

Definition at line 383 of file parameters.hpp.

SCALAR Parameters::dy [protected]

Space step in the second (y) direction.

Definition at line 385 of file parameters.hpp.

int Parameters::flux [protected]

Numerical flux.

Definition at line 431 of file parameters.hpp.

int Parameters::fric [protected]

Friction.

Definition at line 437 of file parameters.hpp.

int Parameters::fric_init [protected]

Type of friction coefficient.

Definition at line 467 of file parameters.hpp.

string Parameters::fric_namefile [protected]

Name of the file for the friction coefficient: Inputs/file.

Definition at line 491 of file parameters.hpp.

string Parameters::fric_NF [protected]

Name of the file for the friction coefficient without 'Inputs'.

Definition at line 493 of file parameters.hpp.

SCALAR Parameters::friccoef [protected]

Friction coefficient.

Definition at line 469 of file parameters.hpp.

int Parameters::huv_init [protected]

Type of initial conditions for h and u,v.

Definition at line 445 of file parameters.hpp.

string Parameters::huv_namefile [protected]

Name of the file for the initialization of h and u,v: Inputs/file.

Definition at line 487 of file parameters.hpp.

string Parameters::huv_NF [protected]

Name of the file for the initialization of h and u,v without 'Inputs'.

Definition at line 489 of file parameters.hpp.

SCALAR Parameters::imax_coef [protected]

Value of imax.

Definition at line 481 of file parameters.hpp.

int Parameters::imax_init [protected]

Type of initialization of imax.

Definition at line 459 of file parameters.hpp.

string Parameters::imax_namefile [protected]

Name of the file for imax: Inputs/file.

Definition at line 519 of file parameters.hpp.

string Parameters::imax_NF [protected]

Name of the file for imax without 'Inputs'.

Definition at line 521 of file parameters.hpp.

int Parameters::inf [protected]

Type of infiltration.

Definition at line 441 of file parameters.hpp.

SCALAR Parameters::Kc_coef [protected]

Value of Kc.

Definition at line 471 of file parameters.hpp.

int Parameters::Kc_init [protected]

Type of initialization of Kc.

Definition at line 449 of file parameters.hpp.

string Parameters::Kc_namefile [protected]

Name of the file for Kc: Inputs/file.

Definition at line 499 of file parameters.hpp.

string Parameters::Kc_NF [protected]

Name of the file for Kc without 'Inputs'.

Definition at line 501 of file parameters.hpp.

SCALAR Parameters::Ks_coef [protected]

Value of Ks.

Definition at line 473 of file parameters.hpp.

int Parameters::Ks_init [protected]

Type of initialization of Ks.

Definition at line 451 of file parameters.hpp.

string Parameters::Ks_namefile [protected]

Name of the file for Ks: Inputs/file.

Definition at line 503 of file parameters.hpp.

string Parameters::Ks_NF [protected]

Name of the file for Ks without 'Inputs'.

Definition at line 505 of file parameters.hpp.

SCALAR Parameters::L [protected]

Length of the domain in the first (x) direction.

Definition at line 387 of file parameters.hpp.

SCALAR Parameters::l [protected]

Length of the domain in the second (y) direction.

Definition at line 389 of file parameters.hpp.

map<int,int> Parameters::Lbound [protected]

Left boundary condition.

Definition at line 391 of file parameters.hpp.

int Parameters::Lbound_type [protected]

Type (constant or file) of left boundary condition

Definition at line 399 of file parameters.hpp.

map<int,SCALAR> Parameters::left_imp_discharge [protected]

Imposed discharge on the left boundary.

Definition at line 393 of file parameters.hpp.

map<int,SCALAR> Parameters::left_imp_h [protected]

Imposed water height on the left boundary.

Definition at line 395 of file parameters.hpp.

map<SCALAR,string> Parameters::left_times_files [protected]

List of files and time values corresponding to the left boundary condition.

Definition at line 397 of file parameters.hpp.

int Parameters::lim [protected]

Slope limiter.

Definition at line 439 of file parameters.hpp.

string Parameters::list_point_namefile [protected]

Name of file containing the list of the specific points without 'Inputs'.

Definition at line 535 of file parameters.hpp.

string Parameters::list_point_NF [protected]

Name of file containing the list of the specific points.

Definition at line 533 of file parameters.hpp.

VECT Parameters::list_points_x [protected]

The list of the specific points.

Definition at line 537 of file parameters.hpp.

VECT Parameters::list_points_y [protected]

Definition at line 537 of file parameters.hpp.

SCALAR Parameters::modifENO [protected]

Parameter for eno_modif.

Definition at line 465 of file parameters.hpp.

int Parameters::nbtimes [protected]

Number of times saved.

Definition at line 379 of file parameters.hpp.

int Parameters::Nxcell [protected]

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

Definition at line 375 of file parameters.hpp.

int Parameters::Nycell [protected]

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

Definition at line 377 of file parameters.hpp.

int Parameters::order [protected]

Order of the numerical scheme.

Definition at line 433 of file parameters.hpp.

string Parameters::output_directory [protected]

Name of the output directory Outputs+suffix.

Definition at line 523 of file parameters.hpp.

int Parameters::output_format [protected]

Type of output.

Definition at line 461 of file parameters.hpp.

string Parameters::path_input_directory [protected]

Name of the input directory

Definition at line 543 of file parameters.hpp.

SCALAR Parameters::Psi_coef [protected]

Value of Psi.

Definition at line 477 of file parameters.hpp.

int Parameters::Psi_init [protected]

Type of initialization of Psi.

Definition at line 455 of file parameters.hpp.

string Parameters::Psi_namefile [protected]

Name of the file for Psi: Inputs/file.

Definition at line 511 of file parameters.hpp.

string Parameters::Psi_NF [protected]

Name of the file for Psi without 'Inputs'.

Definition at line 513 of file parameters.hpp.

int Parameters::rain [protected]

Type of rain.

Definition at line 447 of file parameters.hpp.

string Parameters::rain_namefile [protected]

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

Definition at line 495 of file parameters.hpp.

string Parameters::rain_NF [protected]

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

Definition at line 497 of file parameters.hpp.

map<int,int> Parameters::Rbound [protected]

Right boundary condition.

Definition at line 401 of file parameters.hpp.

int Parameters::Rbound_type [protected]

Type (constant or file) of right boundary condition

Definition at line 409 of file parameters.hpp.

int Parameters::rec [protected]

Reconstruction.

Definition at line 435 of file parameters.hpp.

map<int,SCALAR> Parameters::right_imp_discharge [protected]

Imposed discharge on the right boundary.

Definition at line 403 of file parameters.hpp.

map<int,SCALAR> Parameters::right_imp_h [protected]

Imposed water height on the right boundary.

Definition at line 405 of file parameters.hpp.

map<SCALAR,string> Parameters::right_times_files [protected]

List of files and time values corresponding to the right boundary condition.

Definition at line 407 of file parameters.hpp.

int Parameters::scheme_type [protected]

Type of scheme (fixed cfl or time step).

Definition at line 373 of file parameters.hpp.

string Parameters::suffix_outputs [protected]

Suffix for the output directory.

Definition at line 525 of file parameters.hpp.

SCALAR Parameters::T [protected]

Final time.

Definition at line 381 of file parameters.hpp.

map<int,int> Parameters::Tbound [protected]

Top boundary condition.

Definition at line 421 of file parameters.hpp.

int Parameters::Tbound_type [protected]

Type (constant or file) of top boundary condition

Definition at line 429 of file parameters.hpp.

map<int,SCALAR> Parameters::top_imp_discharge [protected]

Imposed discharge on the top boundary.

Definition at line 423 of file parameters.hpp.

map<int,SCALAR> Parameters::top_imp_h [protected]

Imposed water height on the top boundary.

Definition at line 425 of file parameters.hpp.

map<SCALAR,string> Parameters::top_times_files [protected]

List of files and time values corresponding to the top boundary condition.

Definition at line 427 of file parameters.hpp.

int Parameters::topo [protected]

Type of topography.

Definition at line 443 of file parameters.hpp.

string Parameters::topo_NF [protected]

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

Definition at line 485 of file parameters.hpp.

string Parameters::topography_namefile [protected]

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

Definition at line 483 of file parameters.hpp.

SCALAR Parameters::x_coord [protected]

x coordinate of the specific point to be saved.

Definition at line 529 of file parameters.hpp.

SCALAR Parameters::y_coord [protected]

y coordinate of the specific point to be saved.

Definition at line 531 of file parameters.hpp.

SCALAR Parameters::zcrust_coef [protected]

Value of zcrust.

Definition at line 479 of file parameters.hpp.

int Parameters::zcrust_init [protected]

Type of initialization of zcrust.

Definition at line 457 of file parameters.hpp.

string Parameters::zcrust_namefile [protected]

Name of the file for zcrust: Inputs/file.

Definition at line 515 of file parameters.hpp.

string Parameters::zcrust_NF [protected]

Name of the file for zcrust without 'Inputs'.

Definition at line 517 of file parameters.hpp.

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

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

5.55 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.

5.55.1 Detailed Description

Parser to read the entries

Class that reads the input file written as description <variable>:: value # comment and keep the values after the ":" ignoring the comments that begin with a "#".

Definition at line 80 of file parser.hpp.

5.55.2 Constructor & Destructor Documentation**Parser::Parser (const char * *FILENAME*)**

Constructor.

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

Parameters

in	<i>FILENAME</i>	name of the paramters file.
----	-----------------	-----------------------------

Warning

Impossible to open the *** file.

Note

If the parameters file cannot be opened, the code will exit with failure termination code.

Definition at line 57 of file parser.cpp.

Parser::~~Parser () [virtual]

Destructor.

Definition at line 161 of file parser.cpp.

5.55.3 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:

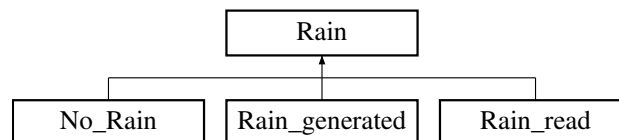
- Headers/libparser/[parser.hpp](#)
- Sources/libparser/[parser.cpp](#)

5.56 Rain Class Reference

Initialization of the rain.

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

Inheritance diagram for Rain:



Public Member Functions

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

Protected Attributes

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

5.56.1 Detailed Description

Initialization of the rain.

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

Definition at line 70 of file rain.hpp.

5.56.2 Constructor & Destructor Documentation

Rain::Rain (Parameters & *par*)

Constructor.

Defines the number of cells and the space steps.

Parameters

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

Definition at line 58 of file rain.cpp.

Rain::~~Rain () [virtual]

Destructor.

Definition at line 68 of file rain.cpp.

5.56.3 Member Function Documentation

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

Function to be specified in each case.

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

5.56.4 Member Data Documentation

const SCALAR Rain::DX [protected]

Space step in the first (x) direction (unused).

Definition at line 89 of file rain.hpp.

const SCALAR Rain::DY [protected]

Space step in the second (y) direction (unused).

Definition at line 91 of file rain.hpp.

const int Rain::NXCELL [protected]

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

Definition at line 85 of file rain.hpp.

const int Rain::NYCELL [protected]

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

Definition at line 87 of file rain.hpp.

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

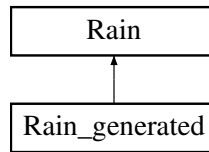
- [Headers/librain_infiltration/rain.hpp](#)
- [Sources/librain_infiltration/rain.cpp](#)

5.57 Rain_generated Class Reference

Constant rain configuration.

```
#include <rain_generated.hpp>
```

Inheritance diagram for Rain_generated:



Public Member Functions

- [Rain_generated](#) (Parameters &)
Constructor.
- void [rain_func](#) (SCALAR, TAB &)
Performs the constant initialization.
- virtual [~Rain_generated](#) ()
Destructor.

Additional Inherited Members

5.57.1 Detailed Description

Constant rain configuration.

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

Definition at line 73 of file rain_generated.hpp.

5.57.2 Constructor & Destructor Documentation

Rain_generated::Rain_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 rain_generated.cpp.

Rain_generated::~~Rain_generated () [virtual]

Destructor.

Definition at line 85 of file rain_generated.cpp.

5.57.3 Member Function Documentation

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

Performs the constant initialization.

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

Parameters

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

Implements [Rain](#).

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

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

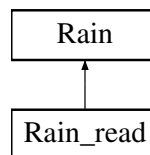
- Headers/librain_infiltration/[rain_generated.hpp](#)
- Sources/librain_infiltration/[rain_generated.cpp](#)

5.58 Rain_read Class Reference

File configuration.

```
#include <rain_read.hpp>
```

Inheritance diagram for `Rain_read`:



Public Member Functions

- [Rain_read](#) ([Parameters](#) &)
Constructor.
- void [rain_func](#) ([SCALAR](#), [TAB](#) &)
Performs the initialization.
- virtual [~Rain_read](#) ()
Destructor.

Additional Inherited Members

5.58.1 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`.

5.58.2 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

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

Warning

(rain_namefile): ERROR: cannot open the rain file.

(rain_namefile): ERROR: line ***.

(rain_namefile): ERROR: the first time must be t = 0.

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

Rain_read::~~Rain_read () [virtual]

Destructor.

Definition at line 151 of file rain_read.cpp.

5.58.3 Member Function Documentation**void Rain_read::rain_func (SCALAR *time*, TAB & *Tab_rain*) [virtual]**

Performs the initialization.

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

Parameters

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

Note

As the times read in the file must start with $t = 0$, *Tab_rain* is initialized.

Implements [Rain](#).

Definition at line 131 of file rain_read.cpp.

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

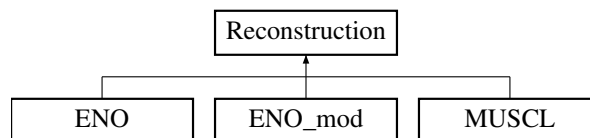
- Headers/librain_infiltration/[rain_read.hpp](#)
- Sources/librain_infiltration/[rain_read.cpp](#)

5.59 Reconstruction Class Reference

Reconstruction of the variables

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

Inheritance diagram for Reconstruction:

**Public Member Functions**

- [Reconstruction](#) (Parameters &, TAB &)

Constructor.

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

Function to be specified in each reconstruction.

- virtual [~Reconstruction](#) ()

Destructor.

Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- [TAB z1r](#)
- [TAB z1l](#)

- [TAB z2r](#)
- [TAB z2l](#)
- [TAB delta_z1](#)
- [TAB delta_z2](#)
- [Choice_limiter](#) * [limiter](#)

5.59.1 Detailed Description

Reconstruction of the variables

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

Definition at line 74 of file reconstruction.hpp.

5.59.2 Constructor & Destructor Documentation

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

Constructor.

Defines the number of cells, the slope limiter, and initializes [Reconstruction::z1l](#), [Reconstruction::z2l](#), [Reconstruction::z1r](#), [Reconstruction::z2r](#), [Reconstruction::delta_z1](#), [Reconstruction::delta_z2](#).

Parameters

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

Definition at line 59 of file reconstruction.cpp.

Reconstruction::~~Reconstruction () [virtual]

Destructor.

Definition at line 109 of file reconstruction.cpp.

5.59.3 Member Function Documentation

virtual void Reconstruction::calcul (TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB & , TAB &) [pure virtual]

Function to be specified in each reconstruction.

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

5.59.4 Member Data Documentation

TAB Reconstruction::delta_z1 [protected]

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

Definition at line 101 of file reconstruction.hpp.

TAB Reconstruction::delta_z2 [protected]

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

Definition at line 103 of file reconstruction.hpp.

Choice_limiter* Reconstruction::limiter [protected]

Slope limiter

Definition at line 105 of file reconstruction.hpp.

const int Reconstruction::NXCELL [protected]

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

Definition at line 89 of file reconstruction.hpp.

const int Reconstruction::NYCELL [protected]

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

Definition at line 91 of file reconstruction.hpp.

TAB Reconstruction::z1l [protected]

Reconstructed topography on the left boundary in the first direction

Definition at line 95 of file reconstruction.hpp.

TAB Reconstruction::z1r [protected]

Reconstructed topography on the right boundary in the first direction

Definition at line 93 of file reconstruction.hpp.

TAB Reconstruction::z2l [protected]

Reconstructed topography on the left boundary in the second direction

Definition at line 99 of file reconstruction.hpp.

TAB Reconstruction::z2r [protected]

Reconstructed topography on the right boundary in the second direction

Definition at line 97 of file reconstruction.hpp.

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

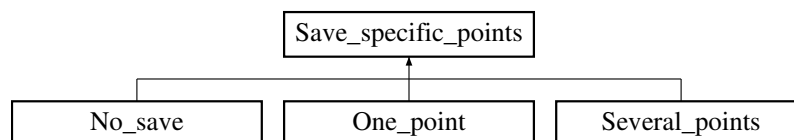
- Headers/libreconstructions/[reconstruction.hpp](#)
- Sources/libreconstructions/[reconstruction.cpp](#)

5.60 Save_specific_points Class Reference

Specific points to save.

```
#include <save_specific_points.hpp>
```

Inheritance diagram for Save_specific_points:



Public Member Functions

- [Save_specific_points](#) (Parameters &)
Constructor.
- virtual void [save](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))=0
Function which writes in a file the values at the specific points.
- virtual [~Save_specific_points](#) ()
Destructor.

Protected Attributes

- const [SCALAR DX](#)
- const [SCALAR DY](#)
- int [row](#)
- int [column](#)
- [SCALAR T_output](#)
- const [SCALAR DT_SPECIFIC_POINTS](#)

5.60.1 Detailed Description

Specific points to save.

Class that contains all the common declarations for the saving of specific points.

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

5.60.2 Constructor & Destructor Documentation

Save_specific_points::Save_specific_points (Parameters & *par*)

Constructor.

Virtual class that defines the common parts for saving one or several specific points.

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

Save_specific_points::~~Save_specific_points () [virtual]

Destructor.

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

5.60.3 Member Function Documentation

virtual void Save_specific_points::save (const TAB & , const TAB & , const TAB & , const SCALAR) [pure virtual]

Function which writes in a file the values at the specific points.

Implemented in [One_point](#), [Several_points](#), and [No_save](#).

5.60.4 Member Data Documentation

int Save_specific_points::column [protected]

Second index of the array from the space variable.

Definition at line 91 of file `save_specific_points.hpp`.

const SCALAR Save_specific_points::DT_SPECIFIC_POINTS [protected]

Time step to save the list of the specific points.

Definition at line 95 of file `save_specific_points.hpp`.

const SCALAR Save_specific_points::DX [protected]

Space step in the first (x) direction.

Definition at line 85 of file `save_specific_points.hpp`.

const SCALAR Save_specific_points::DY [protected]

Space step in the second (y) direction.

Definition at line 87 of file `save_specific_points.hpp`.

int Save_specific_points::row [protected]

First index of the array from the space variable.

Definition at line 89 of file save_specific_points.hpp.

SCALAR Save_specific_points::T_output [protected]

Value of the fixed time step.

Definition at line 93 of file save_specific_points.hpp.

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

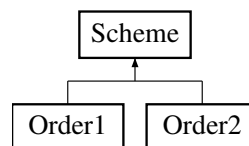
- Headers/libsave/save_specific_points.hpp
- Sources/libsave/save_specific_points.cpp

5.61 Scheme Class Reference

Numerical scheme.

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

Inheritance diagram for Scheme:



Public Member Functions

- [Scheme](#) ([Parameters](#) &)
Constructor.
- virtual void [calcul](#) ()=0
Function to be specified in each numerical scheme.
- void [allocation](#) ()
Allocation of spatialized variables.
- void [deallocation](#) ()
Deallocation of variables.
- void [maincalcfux](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#) &)
Main calculation of the flux.
- void [maincalcscheme](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [SCALAR](#), [SCALAR](#), int)
Main calculation of the scheme.
- void [boundary](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [SCALAR](#), const int, const int)
Calls the boundary conditions and affects the boundary values.
- [SCALAR](#) [froude_number](#) ([TAB](#), [TAB](#), [TAB](#))
Returns the Froude number.
- virtual [~Scheme](#) ()
Destructor.

Protected Attributes

- const int `NXCELL`
- const int `NYCELL`
- const int `ORDER`
- const `SCALAR T`
- const int `NBTIMES`
- const int `SCHEME_TYPE`
- const `SCALAR DX`
- const `SCALAR DY`
- const `SCALAR CFL_FIX`
- `SCALAR DT_FIX`
- `SCALAR tx`
- `SCALAR ty`
- `SCALAR T_output`
- `SCALAR dt_output`
- const `SCALAR FRICCOEF`
- map< int, `SCALAR` > & `L_IMP_Q`
- map< int, `SCALAR` > & `L_IMP_H`
- map< `SCALAR`, string > `left_times_files`
- map< `SCALAR`, string >::const_iterator `p_left_times_files`
- map< int, int > `L_choice_bound`
- int `Lbound_type`
- bool `is_Lbound_changed`
- map< int, `SCALAR` > & `R_IMP_Q`
- map< int, `SCALAR` > & `R_IMP_H`
- map< `SCALAR`, string > `right_times_files`
- map< `SCALAR`, string >::const_iterator `p_right_times_files`
- map< int, int > `R_choice_bound`
- int `Rbound_type`
- bool `is_Rbound_changed`
- map< int, `SCALAR` > & `B_IMP_Q`
- map< int, `SCALAR` > & `B_IMP_H`
- map< `SCALAR`, string > `bottom_times_files`
- map< `SCALAR`, string >::const_iterator `p_bottom_times_files`
- map< int, int > `B_choice_bound`
- int `Bbound_type`
- bool `is_Bbound_changed`
- map< int, `SCALAR` > & `T_IMP_Q`
- map< int, `SCALAR` > & `T_IMP_H`
- map< `SCALAR`, string > `top_times_files`
- map< `SCALAR`, string >::const_iterator `p_top_times_files`
- map< int, int > `T_choice_bound`
- int `Tbound_type`
- bool `is_Tbound_changed`
- `Parameters` & `fs2d_par`
- `TAB z`
- `TAB h`
- `TAB u`

- TAB v
- TAB q1
- TAB q2
- TAB hs
- TAB us
- TAB vs
- TAB qs1
- TAB qs2
- TAB f1
- TAB f2
- TAB f3
- TAB g1
- TAB g2
- TAB g3
- TAB h1left
- TAB h1right
- TAB h2left
- TAB h2right
- TAB delz1
- TAB delz2
- TAB delzc1
- TAB delzc2
- TAB h1r
- TAB u1r
- TAB v1r
- TAB h1l
- TAB u1l
- TAB v1l
- TAB h2r
- TAB u2r
- TAB v2r
- TAB h2l
- TAB u2l
- TAB v2l
- TAB Rain
- TAB Vin_tot
- time_t start
- time_t end
- SCALAR timecomputation
- clock_t cpu_time
- int n
- SCALAR Fr
- SCALAR dt1
- SCALAR dt_max
- SCALAR cur_time
- SCALAR dt_first
- SCALAR Volrain_Tot
- SCALAR Total_volume_outflow
- SCALAR height_of_tot

- SCALAR height_Vinf_tot
- SCALAR Vol_inf_tot_cumul
- SCALAR Vol_of_tot
- Choice_condition * Lbound
- Choice_condition * Rbound
- Choice_condition * Bbound
- Choice_condition * Tbound
- Choice_output * out
- int verif
- Choice_save_specific_points * out_specific_points

5.61.1 Detailed Description

Numerical scheme.

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

Definition at line 116 of file scheme.hpp.

5.61.2 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 60 of file scheme.cpp.

Scheme::~~Scheme () [virtual]

Destructor.

Definition at line 859 of file scheme.cpp.

5.61.3 Member Function Documentation

void Scheme::allocation ()

Allocation of spatialized variables.

Allocation of [Scheme::z](#), [Scheme::h](#), [Scheme::u](#), [Scheme::v](#), [Scheme::q1](#), [Scheme::q2](#), [Scheme::Vin_tot](#), [Scheme::hs](#), [Scheme::us](#), [Scheme::vs](#), [Scheme::qs1](#), [Scheme::qs2](#), [Scheme::f1](#), [Scheme::f2](#), [Scheme::f3](#), [Scheme::g1](#), [Scheme::g2](#), [Scheme::g3](#), [Scheme::h1left](#), [Scheme::h1l](#), [Scheme::u1l](#), [Scheme::v1l](#), [Scheme::h1right](#), [Scheme::h1r](#), [Scheme::u1r](#), [Scheme::v1r](#), [Scheme::h2left](#), [Scheme::h2l](#), [Scheme::u2l](#), [Scheme::v2l](#), [Scheme::h2right](#), [Scheme::h2r](#), [Scheme::u2r](#), [Scheme::v2r](#), [Scheme::delz1](#), [Scheme::delz2](#), [Scheme::delzc1](#), [Scheme::delzc2](#), [Scheme::Rain](#).

Definition at line 584 of file scheme.cpp.

void Scheme::boundary (TAB & *h_tmp*, TAB & *u_tmp*, TAB & *v_tmp*, SCALAR *time_tmp*, const int *NODEX*, const int *NODEY*)

Calls the boundary conditions and affects the boundary values.

Parameters

in, out	<i>h_tmp</i>	water height.
in, out	<i>u_tmp</i>	first component of the velocity.
in, out	<i>v_tmp</i>	second component of the velocity.
in	<i>time_tmp</i>	current time.
in	<i>NODEX</i>	number of space cells in the first (x) direction.
in	<i>NODEY</i>	number of space cells in the second (y) direction.

Definition at line 385 of file scheme.cpp.

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

Function to be specified in each numerical scheme.

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

void Scheme::deallocation ()

Deallocation of variables.

Deallocation of [Scheme::z](#), [Scheme::h](#), [Scheme::u](#), [Scheme::v](#), [Scheme::q1](#), [Scheme::q2](#), [Scheme::Vin](#), [Scheme::hs](#), [Scheme::us](#), [Scheme::vs](#), [Scheme::qs1](#), [Scheme::qs2](#), [Scheme::f1](#), [Scheme::f2](#), [Scheme::f3](#), [Scheme::g1](#), [Scheme::g2](#), [Scheme::g3](#), [Scheme::h1left](#), [Scheme::h1](#), [Scheme::u1l](#), [Scheme::v1l](#), [Scheme::h1right](#), [Scheme::h1r](#), [Scheme::u1r](#), [Scheme::v1r](#), [Scheme::h2left](#), [Scheme::h2l](#), [Scheme::u2l](#), [Scheme::v2l](#), [Scheme::h2right](#), [Scheme::h2r](#), [Scheme::u2r](#), [Scheme::v2r](#), [Scheme::delz1](#), [Scheme::delz2](#), [Scheme::delzc1](#), [Scheme::delzc2](#), [Scheme::Rain](#).

Definition at line 722 of file scheme.cpp.

SCALAR Scheme::froude_number (TAB *h_s*, TAB *u_s*, TAB *v_s*)

Returns the Froude number.

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

Parameters

in	<i>h_s</i>	water height.
in	<i>u_s</i>	first component of the velocity.
in	<i>v_s</i>	second component of the velocity.

Returns

The mean Froude number $\frac{\sqrt{u_s^2 + v_s^2}}{\sqrt{gh_s}}$.

Definition at line 547 of file scheme.cpp.

void Scheme::maincalcflux (SCALAR *cflfix*, SCALAR *T*, SCALAR *curtime*, SCALAR *dt_max*, SCALAR *dt*, SCALAR & *dt_cal*)

Main calculation of the flux.

First part. Construction of variables for hydrostatic reconstruction. Fluxes in the two directions. Computation of the time step for the fixed cfl. This calculation is called once at the order 1, and twice at the second order.

Parameters

in	<i>cflfix</i>	fixed cfl.
----	---------------	------------

in	<i>T</i>	final time (unused).
in	<i>curtime</i>	current time.
in	<i>dt_max</i>	maximum value of the time step.
in	<i>dt</i>	time step.
out	<i>dt_cal</i>	effective time step.

Warning

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

Definition at line 173 of file scheme.cpp.

void Scheme::maincalcscheme (TAB & *he*, TAB & *ve1*, TAB & *ve2*, TAB & *qe1*, TAB & *qe2*, TAB & *hes*, TAB & *ves1*, TAB & *ves2*, TAB & *qes1*, TAB & *qes2*, TAB & *Vin*, SCALAR *curtime*, SCALAR *dt*, int *n*)

Main calculation of the scheme.

Second part. Computation of h, u and v. This calculation is called once at the order 1, and twice at the second order.

Parameters

in	<i>he</i>	water height.
in	<i>ve1</i>	first component of the velocity.
in	<i>ve2</i>	second component of the velocity.
in	<i>qe1</i>	first component of the discharge (unused).
in	<i>qe2</i>	second component of the discharge (unused).
out	<i>hes</i>	water height.
out	<i>ves1</i>	first component of the velocity.
out	<i>ves2</i>	second component of the velocity.
out	<i>qes1</i>	first component of the discharge.
out	<i>qes2</i>	second component of the discharge.
out	<i>Vin</i>	infiltrated volume
in	<i>curtime</i>	current time.
in	<i>dt</i>	time step.
in	<i>n</i>	number of iterations (unused).

Note

In DEBUG mode, the programme will save three other files with boundaries fluxes.

Definition at line 253 of file scheme.cpp.

5.61.4 Member Data Documentation

map<int,int> Scheme::B_choice_bound [protected]

Bottom boundary condition.

Definition at line 216 of file scheme.hpp.

map<int,SCALAR>& Scheme::B_IMP_H [protected]

Imposed water height on the bottom boundary.

Definition at line 210 of file scheme.hpp.

map<int,SCALAR>& Scheme::B_IMP_Q [protected]

Imposed discharge on the bottom boundary.
Definition at line 208 of file scheme.hpp.

Choice_condition* Scheme::Bbound [protected]

The choice of the bottom boundary condition.
Definition at line 352 of file scheme.hpp.

int Scheme::Bbound_type [protected]

Type (constant or file) of bottom boundary condition
Definition at line 218 of file scheme.hpp.

map<SCALAR,string> Scheme::bottom_times_files [protected]

List of files and time values corresponding to the bottom boundary condition.
Definition at line 212 of file scheme.hpp.

const SCALAR Scheme::CFL_FIX [protected]

Value of the fixed cfl.
Definition at line 166 of file scheme.hpp.

clock_t Scheme::cpu_time [protected]

CPU time.
Definition at line 322 of file scheme.hpp.

SCALAR Scheme::cur_time [protected]

The current simulation time.
Definition at line 332 of file scheme.hpp.

TAB Scheme::delz1 [protected]

Variations of the topography along x.
Definition at line 280 of file scheme.hpp.

TAB Scheme::delz2 [protected]

Variations of the topography along y.
Definition at line 282 of file scheme.hpp.

TAB Scheme::delzc1 [protected]

Difference between the reconstructed topographies on the left and on the right boundary of a cell along x.
Definition at line 284 of file scheme.hpp.

TAB Scheme::delzc2 [protected]

Difference between the reconstructed topographies on the left and on the right boundary of a cell along y.
Definition at line 286 of file scheme.hpp.

SCALAR Scheme::dt1 [protected]

Time step in case of fixed cfl.

Definition at line 328 of file scheme.hpp.

SCALAR Scheme::dt_first [protected]

Space step in the first step in the method Heun.

Definition at line 334 of file scheme.hpp.

SCALAR Scheme::DT_FIX [protected]

Value of the fixed time step.

Definition at line 168 of file scheme.hpp.

SCALAR Scheme::dt_max [protected]

Maximum time step in case of fixed cfl.

Definition at line 330 of file scheme.hpp.

SCALAR Scheme::dt_output [protected]

Time step to save the data (evolution file).

Definition at line 176 of file scheme.hpp.

const SCALAR Scheme::DX [protected]

Space step in the first (x) direction.

Definition at line 162 of file scheme.hpp.

const SCALAR Scheme::DY [protected]

Space step in the second (y) direction.

Definition at line 164 of file scheme.hpp.

time_t Scheme::end [protected]

End of timer.

Definition at line 318 of file scheme.hpp.

TAB Scheme::f1 [protected]

First component of the numerical flux along x.

Definition at line 260 of file scheme.hpp.

TAB Scheme::f2 [protected]

Second component of the numerical flux along x.

Definition at line 262 of file scheme.hpp.

TAB Scheme::f3 [protected]

Third component of the numerical flux along x.

Definition at line 264 of file scheme.hpp.

SCALAR Scheme::Fr [**protected**]

Mean Froude number.

Definition at line 326 of file scheme.hpp.

const SCALAR Scheme::FRICCOEF [**protected**]

Friction coefficient.

Definition at line 178 of file scheme.hpp.

Parameters& Scheme::fs2d_par [**protected**]

Parameters object to read the files of boundary conditions

Definition at line 236 of file scheme.hpp.

TAB Scheme::g1 [**protected**]

First component of the numerical flux along y.

Definition at line 266 of file scheme.hpp.

TAB Scheme::g2 [**protected**]

Second component of the numerical flux along y.

Definition at line 268 of file scheme.hpp.

TAB Scheme::g3 [**protected**]

Third component of the numerical flux along y.

Definition at line 270 of file scheme.hpp.

TAB Scheme::h [**protected**]

Water height.

Definition at line 240 of file scheme.hpp.

TAB Scheme::h1l [**protected**]

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

Definition at line 294 of file scheme.hpp.

TAB Scheme::h1left [**protected**]

Hydrostatic reconstruction on the left along x.

Definition at line 272 of file scheme.hpp.

TAB Scheme::h1r [**protected**]

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

Definition at line 288 of file scheme.hpp.

TAB Scheme::h1right [**protected**]

Hydrostatic reconstruction on the right along x.

Definition at line 274 of file scheme.hpp.

TAB Scheme::h2l [protected]

Water height on the cell located at the left of the boundary along y.
Definition at line 306 of file scheme.hpp.

TAB Scheme::h2left [protected]

Hydrostatic reconstruction on the left along y.
Definition at line 276 of file scheme.hpp.

TAB Scheme::h2r [protected]

Water height on the cell located at the right of the boundary along y.
Definition at line 300 of file scheme.hpp.

TAB Scheme::h2right [protected]

Hydrostatic reconstruction on the right along y.
Definition at line 278 of file scheme.hpp.

SCALAR Scheme::height_of_tot [protected]

Cumulative water height on the whole domain
Definition at line 340 of file scheme.hpp.

SCALAR Scheme::height_Vinf_tot [protected]

Cumulative height of infiltrated water on the whole domain
Definition at line 342 of file scheme.hpp.

TAB Scheme::hs [protected]

Water height after one step of the scheme.
Definition at line 250 of file scheme.hpp.

bool Scheme::is_Bbound_changed [protected]

This variable is true if the boundary has been updated
Definition at line 220 of file scheme.hpp.

bool Scheme::is_Lbound_changed [protected]

This variable is true if the boundary has been updated
Definition at line 192 of file scheme.hpp.

bool Scheme::is_Rbound_changed [protected]

This variable is true if the boundary has been updated
Definition at line 206 of file scheme.hpp.

bool Scheme::is_Tbound_changed [protected]

This variable is true if the boundary has been updated
Definition at line 234 of file scheme.hpp.

map<int,int> Scheme::L_choice_bound [protected]

Right boundary condition.

Definition at line 188 of file scheme.hpp.

map<int,SCALAR>& Scheme::L_IMP_H [protected]

Imposed water height on the left boundary.

Definition at line 182 of file scheme.hpp.

map<int,SCALAR>& Scheme::L_IMP_Q [protected]

Imposed discharge on the left boundary.

Definition at line 180 of file scheme.hpp.

Choice_condition* Scheme::Lbound [protected]

The choice of the left boundary condition.

Definition at line 348 of file scheme.hpp.

int Scheme::Lbound_type [protected]

Type (constant or file) of left boundary condition

Definition at line 190 of file scheme.hpp.

map<SCALAR,string> Scheme::left_times_files [protected]

List of files and time values corresponding to the left boundary condition.

Definition at line 184 of file scheme.hpp.

int Scheme::n [protected]

Iterator for the loop in time.

Definition at line 324 of file scheme.hpp.

const int Scheme::NBTIMES [protected]

Number of times saved.

Definition at line 158 of file scheme.hpp.

const int Scheme::NXCELL [protected]

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

Definition at line 150 of file scheme.hpp.

const int Scheme::NYCELL [protected]

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

Definition at line 152 of file scheme.hpp.

const int Scheme::ORDER [protected]

Order of the numerical scheme.

Definition at line 154 of file scheme.hpp.

Choice_output* Scheme::out [protected]

The choice of output.

Definition at line 356 of file scheme.hpp.

Choice_save_specific_points* Scheme::out_specific_points [protected]

The choice of output for the specific points.

Definition at line 360 of file scheme.hpp.

map<SCALAR,string>::const_iterator Scheme::p_bottom_times_files [protected]

Iterator pointing to a file and time value corresponding to the bottom boundary condition

Definition at line 214 of file scheme.hpp.

map<SCALAR,string>::const_iterator Scheme::p_left_times_files [protected]

Iterator pointing to a file and time value corresponding to the left boundary condition

Definition at line 186 of file scheme.hpp.

map<SCALAR,string>::const_iterator Scheme::p_right_times_files [protected]

Iterator pointing to a file and time value corresponding to the right boundary condition

Definition at line 200 of file scheme.hpp.

map<SCALAR,string>::const_iterator Scheme::p_top_times_files [protected]

Iterator pointing to a file and time value corresponding to the top boundary condition

Definition at line 228 of file scheme.hpp.

TAB Scheme::q1 [protected]

First component of the discharge.

Definition at line 246 of file scheme.hpp.

TAB Scheme::q2 [protected]

Second component of the discharge.

Definition at line 248 of file scheme.hpp.

TAB Scheme::qs1 [protected]

First component of the discharge after one step of the scheme.

Definition at line 256 of file scheme.hpp.

TAB Scheme::qs2 [protected]

Second component of the discharge after one step of the scheme.

Definition at line 258 of file scheme.hpp.

map<int,int> Scheme::R_choice_bound [protected]

Right boundary condition.

Definition at line 202 of file scheme.hpp.

map<int,SCALAR>& Scheme::R_IMP_H [protected]

Imposed water height on the right boundary.
Definition at line 196 of file scheme.hpp.

map<int,SCALAR>& Scheme::R_IMP_Q [protected]

Imposed discharge on the right boundary.
Definition at line 194 of file scheme.hpp.

TAB Scheme::Rain [protected]

Rain intensity.
Definition at line 312 of file scheme.hpp.

Choice_condition* Scheme::Rbound [protected]

The choice of the right boundary condition.
Definition at line 350 of file scheme.hpp.

int Scheme::Rbound_type [protected]

Type (constant or file) of right boundary condition
Definition at line 204 of file scheme.hpp.

map<SCALAR,string> Scheme::right_times_files [protected]

List of files and time values corresponding to the right boundary condition.
Definition at line 198 of file scheme.hpp.

const int Scheme::SCHEME_TYPE [protected]

Type of scheme (fixed cfl or time step).
Definition at line 160 of file scheme.hpp.

time_t Scheme::start [protected]

Beginning of timer.
Definition at line 316 of file scheme.hpp.

const SCALAR Scheme::T [protected]

Final time.
Definition at line 156 of file scheme.hpp.

map<int,int> Scheme::T_choice_bound [protected]

Top boundary condition.
Definition at line 230 of file scheme.hpp.

map<int,SCALAR>& Scheme::T_IMP_H [protected]

Imposed water height on the top boundary.
Definition at line 224 of file scheme.hpp.

map<int,SCALAR>& Scheme::T_IMP_Q [protected]

Imposed discharge on the top boundary.

Definition at line 222 of file scheme.hpp.

SCALAR Scheme::T_output [protected]

Time to save the data (evolution file).

Definition at line 174 of file scheme.hpp.

Choice_condition* Scheme::Tbound [protected]

The choice of the top boundary condition.

Definition at line 354 of file scheme.hpp.

int Scheme::Tbound_type [protected]

Type (constant or file) of top boundary condition

Definition at line 232 of file scheme.hpp.

SCALAR Scheme::timecomputation [protected]

Duration of the computation.

Definition at line 320 of file scheme.hpp.

map<SCALAR,string> Scheme::top_times_files [protected]

List of files and time values corresponding to the top boundary condition.

Definition at line 226 of file scheme.hpp.

SCALAR Scheme::Total_volume_outflow [protected]

Cumulative outflow volume at the boundary.

Definition at line 338 of file scheme.hpp.

SCALAR Scheme::tx [protected]

Ratio dt/dx.

Definition at line 170 of file scheme.hpp.

SCALAR Scheme::ty [protected]

Ratio dt/dy.

Definition at line 172 of file scheme.hpp.

TAB Scheme::u [protected]

First component of the velocity.

Definition at line 242 of file scheme.hpp.

TAB Scheme::u1l [protected]

First component of the velocity on the cell located at the left of the boundary along x.

Definition at line 296 of file scheme.hpp.

TAB Scheme::u1r [protected]

First component of the velocity on the cell located at the right of the boundary along x.
Definition at line 290 of file scheme.hpp.

TAB Scheme::u2l [protected]

First component of the velocity on the cell located at the left of the boundary along y.
Definition at line 308 of file scheme.hpp.

TAB Scheme::u2r [protected]

First component of the velocity on the cell located at the right of the boundary along y.
Definition at line 302 of file scheme.hpp.

TAB Scheme::us [protected]

First component of the velocity after one step of the scheme.
Definition at line 252 of file scheme.hpp.

TAB Scheme::v [protected]

Second component of the velocity.
Definition at line 244 of file scheme.hpp.

TAB Scheme::v1l [protected]

Second component of the velocity on the cell located at the left of the boundary along x.
Definition at line 298 of file scheme.hpp.

TAB Scheme::v1r [protected]

Second component of the velocity on the cell located at the right of the boundary along x.
Definition at line 292 of file scheme.hpp.

TAB Scheme::v2l [protected]

Second component of the velocity on the cell located at the left of the boundary along y.
Definition at line 310 of file scheme.hpp.

TAB Scheme::v2r [protected]

Second component of the velocity on the cell located at the right of the boundary along y.
Definition at line 304 of file scheme.hpp.

int Scheme::verif [protected]

Flag for the time step.
Definition at line 358 of file scheme.hpp.

TAB Scheme::Vin_tot [protected]

Cumulative volume of infiltrated water (at each point).
Definition at line 314 of file scheme.hpp.

SCALAR Scheme::Vol_inf_tot_cumul [protected]

Cumulative volume of water infiltrated.

Definition at line 344 of file scheme.hpp.

SCALAR Scheme::Vol_of_tot [protected]

Cumulative streammed volume.

Definition at line 346 of file scheme.hpp.

SCALAR Scheme::Volrain_Tot [protected]

Cumulative volume of rain on the whole domain.

Definition at line 336 of file scheme.hpp.

TAB Scheme::vs [protected]

Second component of the velocity after one step of the scheme.

Definition at line 254 of file scheme.hpp.

TAB Scheme::z [protected]

Topography.

Definition at line 238 of file scheme.hpp.

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

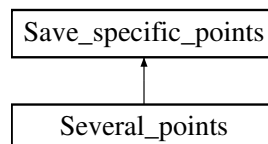
- Headers/libschemas/[scheme.hpp](#)
- Sources/libschemas/[scheme.cpp](#)

5.62 Several_points Class Reference

Several points output.

```
#include <several_points.hpp>
```

Inheritance diagram for Several_points:



Public Member Functions

- [Several_points](#) ([Parameters](#) &)
Constructor.
- void [save](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))
Saves the values at the specific points.
- virtual [~Several_points](#) ()
Destructor.

Additional Inherited Members

5.62.1 Detailed Description

Several points output.

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

Definition at line 73 of file several_points.hpp.

5.62.2 Constructor & Destructor Documentation

Several_points::Several_points (Parameters & *par*)

Constructor.

Writes the header of the file 'hu_specific_points.dat'.

Parameters

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

Warning

***: ERROR: Impossible to open the *** file. Verify if the directory *** exists.

***: WARNING: line *** ; a commentary should begin with the # symbol.

Note

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

x coordinate of the specific point to be saved.

y coordinate of the specific point to be saved.

Name of file containing the list of the specific points.

Definition at line 60 of file several_points.cpp.

Several_points::~~Several_points () [virtual]

Destructor.

Definition at line 188 of file several_points.cpp.

5.62.3 Member Function Documentation

void Several_points::save (const TAB & *h*, const TAB & *u*, const TAB & *v*, const SCALAR *time*) [virtual]

Saves the values at the specific points.

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

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

Parameters

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

<code>in</code>	<code>time</code>	the current time.
-----------------	-------------------	-------------------

Implements [Save_specific_points](#).

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

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

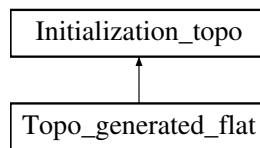
- Headers/libsave/[several_points.hpp](#)
- Sources/libsave/[several_points.cpp](#)

5.63 Topo_generated_flat Class Reference

Flat configuration.

```
#include <topo_generated_flat.hpp>
```

Inheritance diagram for `Topo_generated_flat`:



Public Member Functions

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

Additional Inherited Members

5.63.1 Detailed Description

Flat configuration.

Class that initializes a flat topography, with value 0.

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

5.63.2 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 60 of file `topo_generated_flat.cpp`.

Topo_generated_flat::~~Topo_generated_flat () [virtual]

Destructor.

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

5.63.3 Member Function Documentation

void Topo_generated_flat::initialization (TAB & topo) [virtual]

Performs the initialization.

Initializes the topography to 0.

Parameters

in, out	topo	topography.
---------	------	-------------

Implements [Initialization_topo](#).

Definition at line 69 of file topo_generated_flat.cpp.

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

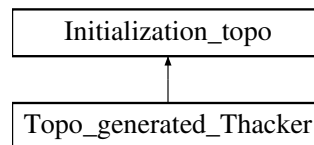
- Headers/libinitializations/[topo_generated_flat.hpp](#)
- Sources/libinitializations/[topo_generated_flat.cpp](#)

5.64 Topo_generated_Thacker Class Reference

Thacker configuration.

```
#include <topo_generated_thacker.hpp>
```

Inheritance diagram for Topo_generated_Thacker:



Public Member Functions

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

Additional Inherited Members

5.64.1 Detailed Description

Thacker configuration.

Class that initializes a topography for Thacker's benchmark (shape of a paraboloid of revolution).

Definition at line 73 of file topo_generated_thacker.hpp.

5.64.2 Constructor & Destructor Documentation

Topo_generated_Thacker::Topo_generated_Thacker (Parameters & par)

Constructor.

Defines the parameters of the paraboloid.

Parameters

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

Definition at line 60 of file topo_generated_thacker.cpp.

Topo_generated_Thacker::~~Topo_generated_Thacker () [virtual]

Destructor.

Definition at line 98 of file topo_generated_thacker.cpp.

5.64.3 Member Function Documentation**void Topo_generated_Thacker::initialization (TAB & topo) [virtual]**

Performs the initialization.

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

Parameters

in, out	topo	topography.
---------	------	-------------

Implements [Initialization_topo](#).

Definition at line 78 of file topo_generated_thacker.cpp.

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

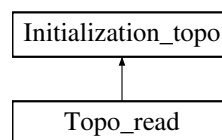
- [Headers/libinitializations/topo_generated_thacker.hpp](#)
- [Sources/libinitializations/topo_generated_thacker.cpp](#)

5.65 Topo_read Class Reference

File configuration.

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

Inheritance diagram for Topo_read:

**Public Member Functions**

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

Additional Inherited Members**5.65.1 Detailed Description**

File configuration.

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

Definition at line 72 of file topo_read.hpp.

5.65.2 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 60 of file topo_read.cpp.

Topo_read::~~Topo_read () [virtual]

Destructor.

Definition at line 187 of file topo_read.cpp.

5.65.3 Member Function Documentation

void Topo_read::initialization (TAB & *topo*) [virtual]

Performs the initialization.

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

Parameters

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

Warning

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

Note

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

Implements [Initialization_topo](#).

Definition at line 72 of file topo_read.cpp.

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

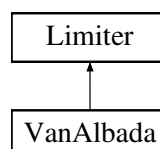
- Headers/libinitializations/[topo_read.hpp](#)
- Sources/libinitializations/[topo_read.cpp](#)

5.66 VanAlbada Class Reference

Van Albada slope limiter.

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

Inheritance diagram for VanAlbada:



Public Member Functions

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

Additional Inherited Members

5.66.1 Detailed Description

Van Albada slope limiter.

Class that calculates Van Albada slope limiter.

Definition at line 70 of file vanalbada.hpp.

5.66.2 Constructor & Destructor Documentation

VanAlbada::VanAlbada ()

Constructor.

Definition at line 59 of file vanalbada.cpp.

VanAlbada::~VanAlbada () [virtual]

Destructor.

Definition at line 85 of file vanalbada.cpp.

5.66.3 Member Function Documentation

void VanAlbada::calcul (SCALAR *a*, SCALAR *b*) [virtual]

Calculates the value of the slope limiter.

Van Albada function:

$$VA(x,y) = \begin{cases} 0 & \text{if } \text{sign}(x) \neq \text{sign}(y), \\ \frac{x(y^2 + \varepsilon) + y(x^2 + \varepsilon)}{x^2 + y^2 + 2\varepsilon} & \text{else,} \end{cases}$$

with $0 \leq \varepsilon \ll 1$.

Parameters

<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 vanalbada.cpp.

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

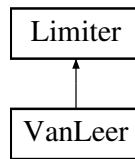
- Headers/liblimitations/[vanalbada.hpp](#)
- Sources/liblimitations/[vanalbada.cpp](#)

5.67 VanLeer Class Reference

Van Leer slope limiter.

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

Inheritance diagram for VanLeer:



Public Member Functions

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

Additional Inherited Members

5.67.1 Detailed Description

Van Leer slope limiter.

Class that calculates Van Leer slope limiter.

Definition at line 70 of file vanleer.hpp.

5.67.2 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.

5.67.3 Member Function Documentation

void VanLeer::calcul (SCALAR a, SCALAR b) [virtual]

Calculates the value of the slope limiter.

Van Leer function:

$$VL(x,y) = \begin{cases} 0 & \text{if } xy \leq 0, \\ \frac{2xy}{x+y} & \text{else.} \end{cases}$$

Parameters

<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:

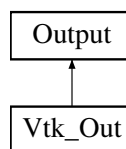
- Headers/liblimitations/[vanleer.hpp](#)
- Sources/liblimitations/[vanleer.cpp](#)

5.68 Vtk_Out Class Reference

VTK output.

```
#include <vtk_out.hpp>
```

Inheritance diagram for `Vtk_Out`:

**Public Member Functions**

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

Additional Inherited Members**5.68.1 Detailed Description**

VTK output.

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

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

5.68.2 Constructor & Destructor Documentation

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

Constructor.

Defines the output name `huz_evolution.dat`

Parameters

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

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

Vtk_Out::~Vtk_Out () [virtual]

Destructor.

Definition at line 259 of file `vtk_out.cpp`.

5.68.3 Member Function Documentation**void Vtk_Out::write (const TAB & *h*, const TAB & *u*, const TAB & *v*, const TAB & *z*, const SCALAR & *time*) [virtual]**

Saves one time step.

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

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

Parameters

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

Note

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

Implements [Output](#).

Definition at line 76 of file `vtk_out.cpp`.

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

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

Chapter 6

File Documentation

6.1 Headers/libboundaryconditions/bc_imp_discharge.hpp File Reference

Imposed discharge.

```
#include "boundary_condition.hpp"
```

Classes

- class [Bc_imp_discharge](#)
Imposed discharge.

Macros

- #define [BC_IMP_DISCHARGE_HPP](#)

6.1.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison ulrich.razafison@math.cnrs.fr (2011)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.01

Date

2015-10-29

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 - BRGM (France)

6.1.2 Macro Definition Documentation

```
#define BC_IMP_DISCHARGE_HPP
```

Definition at line 63 of file `bc_imp_discharge.hpp`.

6.2 Headers/`libboundaryconditions/bc_imp_height.hpp` File Reference

Imposed water height.

```
#include "boundary_condition.hpp"
```

Classes

- class [Bc_imp_height](#)
Imposed water height.

6.2.1 Detailed Description

Imposed water height.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2010-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: imposed water height (and discharge if necessary), based on the modified method of characteristics and Riemann invariants.

See also

Olivier Delestre Ph.D thesis Annexe A [Delestre \[2010\]](#) <http://tel.archives-ouvertes.fr/tel-00587197>

Copyright

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

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

6.3 Headers/`libboundaryconditions/bc_neumann.hpp` File Reference

Neumann condition.

```
#include "boundary_condition.hpp"
```

Classes

- class [Bc_Neumann](#)
Neumann condition.

6.3.1 Detailed Description

Neumann condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

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

Copyright

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

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

6.4 Headers/libboundaryconditions/bc_periodic.hpp File Reference

Periodic condition.

```
#include "boundary_condition.hpp"
```

Classes

- class [Bc_periodic](#)
Periodic condition.

6.4.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2010)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: periodic condition.

Copyright

License Cecill-V2

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

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

6.5 Headers/libboundaryconditions/bc_wall.hpp File Reference

Wall condition.

```
#include "boundary_condition.hpp"
```

Classes

- class [Bc_wall](#)
Wall condition.

6.5.1 Detailed Description

Wall condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: wall condition (the discharge at the boundary is null).

Copyright

License Cecill-V2

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

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

6.6 Headers/libboundaryconditions/boundary_condition.hpp File Reference

Boundary condition.

```
#include "parameters.hpp"
```

Classes

- class [Boundary_condition](#)
Boundary condition.

6.6.1 Detailed Description

Boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2017-03-20

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 - BRGM (France)

6.7 Headers/libboundaryconditions/choice_condition.hpp File Reference

Choice of boundary condition.

```
#include "boundary_condition.hpp"
#include "bc_imp_height.hpp"
#include "bc_wall.hpp"
#include "bc_neumann.hpp"
#include "bc_periodic.hpp"
#include "bc_imp_discharge.hpp"
```

Classes

- class [Choice_condition](#)
Choice of boundary condition.

Macros

- #define [CHOICE_CONDITION_HPP](#)

6.7.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-09-11

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 - BRGM (France)

6.7.2 Macro Definition Documentation

#define CHOICE_CONDITION_HPP

Definition at line 84 of file choice_condition.hpp.

6.8 Headers/libflux/choice_flux.hpp File Reference

Choice of numerical flux.

```
#include "flux.hpp"  
#include "f_rusanov.hpp"  
#include "f_h11.hpp"  
#include "f_h112.hpp"  
#include "f_h11c.hpp"  
#include "f_h11c2.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 (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.01

Date

2016-01-04

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 - BRGM (France)

6.8.2 Macro Definition Documentation

#define CHOICE_FLUX_HPP

Definition at line 84 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)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

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

Copyright

License Cecill-V2

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

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

6.10 Headers/libflux/f_hll2.hpp File Reference

HLL flux.

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

Classes

- class [F_HLL2](#)

HLL flux.

6.10.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

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

Copyright

License Cecill-V2

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

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

6.11 Headers/libflux/f_hllc.hpp File Reference

HLLC flux.

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


Classes

- class [F_HLLC](#)
HLLC flux.

6.11.1 Detailed Description

HLLC flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.01

Date

2016-01-04

Numerical flux: Harten, Lax, van Leer reduced formulation with restoration of the Contact Surface.

Copyright

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

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

6.12 Headers/libflux/f_hllc2.hpp File Reference

HLLC flux.

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

Classes

- class [F_HLLC2](#)

6.12.1 Detailed Description

HLLC flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.01

Date

2016-01-04

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 - BRGM (France)

6.13 Headers/libflux/f_rusanov.hpp File Reference

Rusanov flux.

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

Classes

- class [F_Rusanov](#)

Rusanov flux.

6.13.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - BRGM (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)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the numerical fluxes.

Copyright

License Cecill-V2

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

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

6.15 Headers/libfrictions/choice_friction.hpp File Reference

Choice of friction law.

```
#include "friction.hpp"
#include "fr_manning.hpp"
#include "fr_darcy_weisbach.hpp"
#include "no_friction.hpp"
#include "fr_laminar.hpp"
```

Classes

- class [Choice_friction](#)
Choice of friction law.

Macros

- #define [CHOICE_FRICTION_HPP](#)

6.15.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

License Cecill-V2

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

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

6.15.2 Macro Definition Documentation

#define CHOICE_FRICTION_HPP

Definition at line 80 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)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - BRGM (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

1.06.00

Date

2015-02-19

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)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Manning.

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - BRGM (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)Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the friction laws.

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - BRGM (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

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: does no computation.

Copyright

License Cecill-V2

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

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

6.21 Headers/libinitializations/choice_init_huv.hpp File Reference

Choice of initialization for h, u and v.

```
#include "initialization_huv.hpp"
#include "huv_read.hpp"
#include "huv_generated.hpp"
#include "huv_generated_thacker.hpp"
#include "huv_generated_radial_dam_dry.hpp"
#include "huv_generated_radial_dam_wet.hpp"
```

Classes

- class [Choice_init_huv](#)

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

Macros

- #define [CHOICE_INIT_HUV_HPP](#)

6.21.1 Detailed Description

Choice of initialization for h, u and v.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the water height and of the velocity.

Copyright

License Cecill-V2

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

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

6.21.2 Macro Definition Documentation

#define CHOICE_INIT_HUV_HPP

Definition at line 83 of file choice_init_huv.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"
```

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)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

License Cecill-V2

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

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

6.22.2 Macro Definition Documentation

#define CHOICE_INIT_TOPO_HPP

Definition at line 75 of file choice_init_topo.hpp.

6.23 Headers/libinitializations/huv_generated.hpp File Reference

No water configuration.

```
#include "initialization_huv.hpp"
```

Classes

- class [Huv_generated](#)

No water configuration.

6.23.1 Detailed Description

No water configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a dry domain.

Copyright

License Cecill-V2

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

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

6.24 Headers/libinitializations/huv_generated_radial_dam_dry.hpp File Reference

Dry radial dam break configuration.

```
#include "initialization_huv.hpp"
```

Classes

- class [Huv_generated_Radial_Dam_dry](#)
Dry radial dam break configuration.

6.24.1 Detailed Description

Dry radial dam break configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a dry domain.

Copyright

License Cecill-V2

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

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

6.25 Headers/libinitializations/huv_generated_radial_dam_wet.hpp File Reference

Wet radial dam break configuration.

```
#include "initialization_huv.hpp"
```

Classes

- class [Huv_generated_Radial_Dam_wet](#)
Wet radial dam break configuration.

6.25.1 Detailed Description

Wet radial dam break configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a wet domain.

Copyright

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

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

6.26 Headers/libinitializations/huv_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_huv.hpp"
```

Classes

- class [Huv_generated_Thacker](#)
Thacker configuration.

6.26.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of Thacker's benchmark.

Copyright

License Cecill-V2

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

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

6.27 Headers/libinitializations/huv_read.hpp File Reference

File configuration.

```
#include "initialization_huv.hpp"
```

Classes

- class [Huv_read](#)

File configuration.

6.27.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: the values are read in a file.

Copyright

License Cecill-V2

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

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

6.28 Headers/libinitializations/initialization_huv.hpp File Reference

Initialization of h, u and v

```
#include "parameters.hpp"
```

Classes

- class [Initialization_huv](#)
Initialization of h, u and v.

6.28.1 Detailed Description

Initialization of h, u and v

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the water height and of the velocity.

Copyright

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

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

6.29 Headers/libinitializations/initialization_topo.hpp File Reference

Initialization of z

```
#include "parameters.hpp"
```

Classes

- class [Initialization_topo](#)
Initialization of z.

6.29.1 Detailed Description

Initialization of z

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-201())

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the topography.

Copyright

License Cecill-V2

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

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

6.30 Headers/libinitializations/topo_generated_flat.hpp File Reference

Flat configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_generated_flat](#)

Flat configuration.

6.30.1 Detailed Description

Flat configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the topography is flat, its value is 0.

Copyright

License Cecill-V2

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

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

6.31 Headers/libinitializations/topo_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_generated_Thacker](#)
Thacker configuration.

6.31.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: topography with a shape of a paraboloid of revolution for Thacker's Benchmark.

Copyright

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

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

6.32 Headers/libinitializations/topo_read.hpp File Reference

File configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_read](#)
File configuration.

6.32.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the values are read in a file.

Copyright

License Cecill-V2

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

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

6.33 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](#)

6.33.1 Detailed Description

Choice of slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

License Cecill-V2

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

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

6.33.2 Macro Definition Documentation

#define CHOICE_LIMITER_HPP

Definition at line 75 of file choice_limiter.hpp.

6.34 Headers/liblimitations/limiter.hpp File Reference

Slope limiter.

```
#include "parameters.hpp"
```

Classes

- class [Limiter](#)
Slope limiter.

6.34.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the slope limiters.

Copyright

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

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

6.35 Headers/liblimitations/minmod.hpp File Reference

Minmod limiter

```
#include "limiter.hpp"
```

Classes

- class [Minmod](#)
Minmod slope limiter

6.35.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: minmod.

Copyright

License Cecill-V2

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

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

6.36 Headers/liblimitations/vanalbada.hpp File Reference

Van Albada limiter.

```
#include "limiter.hpp"
```

Classes

- class [VanAlbada](#)

Van Albada slope limiter.

6.36.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Albada.

Copyright

License Cecill-V2

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

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

6.37 Headers/liblimitations/vanleer.hpp File Reference

Van Leer limiter.

```
#include "limiter.hpp"
```

Classes

- class [VanLeer](#)

Van Leer slope limiter.

6.37.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Leer.

Copyright

License Cecill-V2

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

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

6.38 Headers/libparameters/misc.hpp File Reference

Definitions.

```
#include <vector>
#include <iostream>
#include <cmath>
#include <fstream>
#include <string>
#include <cstring>
#include <cstdlib>
#include <iomanip>
#include <sstream>
#include <cstdio>
#include <unistd.h>
#include <ctime>
#include <map>
```

Macros

- #define `max(a, b)` $(a \geq b ? a : b)$
- #define `min(a, b)` $(a \leq b ? a : b)$
- #define `GRAV` 9.81
- #define `GRAV_DEM` 4.905
- #define `CONST_CFL_X` 0.5
- #define `CONST_CFL_Y` 0.5
- #define `HE_CA` 1.e-12
- #define `VE_CA` 1.e-12
- #define `MAX_CFL_X` 0.
- #define `MAX_CFL_Y` 0.
- #define `MAX_ITER` 1000000000
- #define `NB_CHAR` 256
- #define `ZERO` 0.
- #define `IE_CA` 1.e-8
- #define `EPSILON` 1.e-13
- #define `VERSION` "FullSWOF_2D version 1.08.00, 2018-01-31"
- #define `RATIO_CLOSE_CELL` 1.e-3
- #define `MAX_SCAL` DBL_MAX

Typedefs

- typedef double `SCALAR`
- typedef vector< vector< `SCALAR` > > `TAB`
- typedef vector< `SCALAR` > `VECT`

6.38.1 Detailed Description

Definitions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
 Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.08.00

Date

2018-01-31

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)

6.38.2 Macro Definition Documentation

#define CONST_CFL_X 0.5

Definition at line 75 of file misc.hpp.

#define CONST_CFL_Y 0.5

Definition at line 76 of file misc.hpp.

#define EPSILON 1.e-13

Definition at line 86 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-12

Definition at line 77 of file misc.hpp.

#define IE_CA 1.e-8

Definition at line 85 of file misc.hpp.

#define max(a, b) (a>=b?a:b)

Definition at line 70 of file misc.hpp.

#define MAX_CFL_X 0.

Definition at line 79 of file misc.hpp.

#define MAX_CFL_Y 0.

Definition at line 80 of file misc.hpp.

#define MAX_ITER 1000000000

Definition at line 81 of file misc.hpp.

#define MAX_SCAL DBL_MAX

Definition at line 97 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 83 of file misc.hpp.

```
#define RATIO_CLOSE_CELL 1.e-3
```

Definition at line 90 of file misc.hpp.

```
#define VE_CA 1.e-12
```

Definition at line 78 of file misc.hpp.

```
#define VERSION "FullSWOF_2D version 1.08.00, 2018-01-31"
```

Definition at line 87 of file misc.hpp.

```
#define ZERO 0.
```

Definition at line 84 of file misc.hpp.

6.38.3 Typedef Documentation

```
typedef double SCALAR
```

Definition at line 94 of file misc.hpp.

```
typedef vector< vector< SCALAR > > TAB
```

Definition at line 99 of file misc.hpp.

```
typedef vector<SCALAR> VECT
```

Definition at line 101 of file misc.hpp.

6.39 Headers/libparameters/parameters.hpp File Reference

Gets parameters.

```
#include "misc.hpp"  
#include "parser.hpp"
```

Classes

- class [Parameters](#)

Gets parameters.

6.39.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2011-2017)

Version

1.07.01

Date

2017-05-09

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 - BRGM (France)

6.40 Headers/libparser/parser.hpp File Reference

Parser

```
#include "misc.hpp"
```

Classes

- class [Parser](#)

Parser to read the entries

6.40.1 Detailed Description

Parser

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2010-2015)

Version

1.06.00

Date

2015-02-19

Reads the input file.

Copyright

License Cecill-V2

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

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

6.41 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](#)

6.41.1 Detailed Description

Choice of infiltration law.

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter: calls the chosen infiltration law.

Copyright

License Cecill-V2

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

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

6.41.2 Macro Definition Documentation

#define CHOICE_INFILTRATION_HPP

Definition at line 72 of file choice_infiltration.hpp.

6.42 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](#)

6.42.1 Detailed Description

Choice of initialization for the rain.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the rain.

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - BRGM (France)

6.42.2 Macro Definition Documentation

#define CHOICE_RAIN_HPP

Definition at line 75 of file choice_rain.hpp.

6.43 Headers/librain_infiltration/greenampt.hpp File Reference

Green-Ampt law.

```
#include "infiltration.hpp"
```

Classes

- class [GreenAmpt](#)
Green-Ampt law.

6.43.1 Detailed Description

Green-Ampt law.

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.44 Headers/librain_infiltration/infiltration.hpp File Reference

Infiltration law

```
#include "parameters.hpp"
```

Classes

- class [Infiltration](#)

Definition of infiltration law.

6.44.1 Detailed Description

Infiltration law

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.45 Headers/librain_infiltration/no_infiltration.hpp File Reference

No infiltration.

```
#include "infiltration.hpp"
```

Classes

- class [No_Infiltration](#)
No infiltration.

6.45.1 Detailed Description

No infiltration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.46 Headers/librain_infiltration/no_rain.hpp File Reference

No rain.

```
#include "rain.hpp"
```

Classes

- class [No_Rain](#)
No rain.

6.46.1 Detailed Description

No rain.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Rain: there is no rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.47 Headers/librain_infiltration/rain.hpp File Reference

Rain

```
#include "parameters.hpp"
```

Classes

- class [Rain](#)
Initialization of the rain.

6.47.1 Detailed Description

Rain

Author

Marie Rousseau ma.rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the initialization of the rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.48 Headers/librain_infiltration/rain_generated.hpp File Reference

Constant rain configuration.

```
#include "rain.hpp"
```

Classes

- class [Rain_generated](#)
Constant rain configuration.

6.48.1 Detailed Description

Constant rain configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the value is equals to 0.00001 m/s = 36 mm/h, constant during the simulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.49 Headers/librain_infiltration/rain_read.hpp File Reference

File configuration.

```
#include "rain.hpp"
```

Classes

- class [Rain_read](#)
File configuration.

6.49.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the values are read in a file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.50 Headers/libreconstructions/choice_reconstruction.hpp File Reference

Choice of reconstruction.

```
#include "reconstruction.hpp"  
#include "muscl.hpp"  
#include "eno.hpp"  
#include "eno_mod.hpp"
```

Classes

- class [Choice_reconstruction](#)
Choice of reconstruction.

Macros

- #define [CHOICE_RECONSTRUCTION](#)

6.50.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen reconstruction.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.50.2 Macro Definition Documentation

#define CHOICE_RECONSTRUCTION

Definition at line 76 of file choice_reconstruction.hpp.

6.51 Headers/libreconstructions/eno.hpp File Reference

ENO reconstruction

```
#include "reconstruction.hpp"
```

Classes

- class [ENO](#)
ENO reconstruction

6.51.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.52 Headers/libreconstructions/eno_mod.hpp File Reference

Modified ENO reconstruction.

```
#include "reconstruction.hpp"
```

Classes

- class [ENO_mod](#)

Modified ENO reconstruction.

6.52.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.53 Headers/libreconstructions/hydrostatic.hpp File Reference

Hydrostatic reconstruction

```
#include "misc.hpp"
```

Classes

- class [Hydrostatic](#)
Hydrostatic reconstruction

6.53.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.54 Headers/libreconstructions/muscl.hpp File Reference

MUSCL reconstruction

```
#include "reconstruction.hpp"
```

Classes

- class [MUSCL](#)
MUSCL reconstruction

6.54.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.55 Headers/libreconstructions/reconstruction.hpp File Reference

Reconstruction

```
#include "parameters.hpp"  
#include "choice_limiter.hpp"
```

Classes

- class [Reconstruction](#)

Reconstruction of the variables

6.55.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the reconstructions.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.56 Headers/libsave/choice_output.hpp File Reference

Choice of output format.

```
#include "output.hpp"  
#include "gnuplot.hpp"  
#include "vtk_out.hpp"  
#include "no_evolution_file.hpp"
```

Classes

- class [Choice_output](#)
Choice of output format.

Macros

- #define [CHOICE_OUTPUT_HPP](#)

6.56.1 Detailed Description

Choice of output format.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

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 - BRGM (France)

6.56.2 Macro Definition Documentation

#define CHOICE_OUTPUT_HPP

Definition at line 74 of file choice_output.hpp.

6.57 Headers/libsave/choice_save_specific_points.hpp File Reference

Choice of the output of the specific points.

```
#include "save_specific_points.hpp"  
#include "one_point.hpp"  
#include "several_points.hpp"  
#include "no_save.hpp"
```

Classes

- class [Choice_save_specific_points](#)
Choice of the output of the specific points.

Macros

- #define [CHOICE_SAVE_SPECIFIC_POINTS_HPP](#)

6.57.1 Detailed Description

Choice of the output of the specific points.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

From the value of the corresponding parameter, calls the savings at the specific points.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.57.2 Macro Definition Documentation

#define CHOICE_SAVE_SPECIFIC_POINTS_HPP

Definition at line 74 of file choice_save_specific_points.hpp.

6.58 Headers/libsave/gnuplot.hpp File Reference

Gnuplot output

```
#include "output.hpp"
```

Classes

- class [Gnuplot](#)
Gnuplot output

6.58.1 Detailed Description

Gnuplot output

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

Output format: optimized for Gnuplot.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.59 Headers/libsave/no_evolution_file.hpp File Reference

No output.

```
#include "output.hpp"
```

Classes

- class [No_Evolution_File](#)

No output.

6.59.1 Detailed Description

No output.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

No output files with time evolution

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.60 Headers/libsave/no_save.hpp File Reference

No save.

```
#include "save_specific_points.hpp"
```

Classes

- class [No_save](#)

No output.

6.60.1 Detailed Description

No save.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

No specific point to save

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.61 Headers/libsave/one_point.hpp File Reference

One point output.

```
#include "save_specific_points.hpp"
```

Classes

- class [One_point](#)

One point output.

6.61.1 Detailed Description

One point output.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

Output format: optimized for Gnuplot.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.62 Headers/libsave/output.hpp File Reference

Output format

```
#include "parameters.hpp"
```

Classes

- class [Output](#)

Output format

6.62.1 Detailed Description

Output format

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

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 - BRGM (France)

6.63 Headers/libsave/save_specific_points.hpp File Reference

Specific points to save.

```
#include "parameters.hpp"
```

Classes

- class [Save_specific_points](#)
Specific points to save.

6.63.1 Detailed Description

Specific points to save.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

Common part for all the output of the specific points.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.64 Headers/libsave/several_points.hpp File Reference

Saving several specific points.

```
#include "save_specific_points.hpp"
```

Classes

- class [Several_points](#)
Several points output.

6.64.1 Detailed Description

Saving several specific points.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

Output format: optimized for Gnuplot.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.65 Headers/libsave/vtk_out.hpp File Reference

VTK output

```
#include "output.hpp"
```

Classes

- class [Vtk_Out](#)
VTK output.

6.65.1 Detailed Description

VTK output

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

Output format: optimized for software compatible with vtk format (example: paraview).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.66 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](#)

6.66.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical scheme.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.66.2 Macro Definition Documentation

#define CHOICE_SCHEME_HPP

Definition at line 72 of file choice_scheme.hpp.

6.67 Headers/libschemas/order1.hpp File Reference

Order 1 scheme.

```
#include "scheme.hpp"
```

Classes

- class [Order1](#)

Order 1 scheme.

6.67.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 1.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.68 Headers/libschemas/order2.hpp File Reference

Order 2 scheme.

```
#include "scheme.hpp"
```

Classes

- class [Order2](#)

Order 2 scheme.

6.68.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 2.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.69 Headers/libschemas/scheme.hpp File Reference

Numerical scheme.

```
#include "parameters.hpp"
#include "hydrostatic.hpp"
#include "choice_condition.hpp"
#include "choice_flux.hpp"
#include "choice_friction.hpp"
#include "choice_infiltration.hpp"
#include "choice_init_topo.hpp"
#include "choice_init_huv.hpp"
#include "choice_rain.hpp"
#include "choice_output.hpp"
#include "choice_reconstruction.hpp"
#include "choice_save_specific_points.hpp"
```

Classes

- class [Scheme](#)
Numerical scheme.

6.69.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-05-09

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 - BRGM (France)

6.70 Sources/FullSWOF_2D.cpp File Reference

Main function.

```
#include "choice_scheme.hpp"
```

Functions

- int [main](#) (int argc, char **argv)

6.70.1 Detailed Description

Main function.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Runs the programm.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.70.2 Function Documentation

int main (int *argc*, char ** *argv*)

Main function

Declare the scheme and executes the program.

Returns

0 if the program finished correctly.

Note

The name of the input file (Inputs/parameters.txt) is written here.

Definition at line 58 of file FullSWOF_2D.cpp.

6.71 Sources/libboundaryconditions/bc_imp_discharge.cpp File Reference

Imposed discharge.

```
#include "bc_imp_discharge.hpp"
```

6.71.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison ulrich.razafison@math.cnrs.fr (2011)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-03-20

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 - BRGM (France)

6.72 Sources/libboundaryconditions/bc_imp_height.cpp File Reference

Imposed water height.

```
#include "bc_imp_height.hpp"
```

6.72.1 Detailed Description

Imposed water height.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2010-2015)

Version

1.07.01

Date

2017-03-20

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 - BRGM (France)

6.73 Sources/libboundaryconditions/bc_neumann.cpp File Reference

Neumann condition.

```
#include "bc_neumann.hpp"
```

6.73.1 Detailed Description

Neumann condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-03-20

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 - BRGM (France)

6.74 Sources/libboundaryconditions/bc_periodic.cpp File Reference

Periodic condition.

```
#include "bc_periodic.hpp"
```

6.74.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2010)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-03-20

Boundary condition: periodic condition.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.75 Sources/libboundaryconditions/bc_wall.cpp File Reference

Wall condition.

```
#include "bc_wall.hpp"
```

6.75.1 Detailed Description

Wall condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-03-20

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 - BRGM (France)

6.76 Sources/libboundaryconditions/boundary_condition.cpp File Reference

Boundary condition.

```
#include "boundary_condition.hpp"
```

6.76.1 Detailed Description

Boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-05-09

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 - BRGM (France)

6.77 Sources/libboundaryconditions/choice_condition.cpp File Reference

Choice of boundary condition.

```
#include "choice_condition.hpp"
```

6.77.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-04-12

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 - BRGM (France)

6.78 Sources/libflux/choice_flux.cpp File Reference

Choice of numerical flux.

```
#include "choice_flux.hpp"
```

6.78.1 Detailed Description

Choice of numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.79 Sources/libflux/f_hll.cpp File Reference

HLL flux.

```
#include "f_hll.hpp"
```

6.79.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.80 Sources/libflux/f_hll2.cpp File Reference

HLL flux.

```
#include "f_hll2.hpp"
```

6.80.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.81 Sources/libflux/f_hllc.cpp File Reference

HLLC flux.

```
#include "f_hllc.hpp"
```

6.81.1 Detailed Description

HLLC flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Numerical flux: Harten, Lax, van Leer formulation with restoration of the Contact Surface.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.82 Sources/libflux/f_hllc2.cpp File Reference

```
#include "f_hllc2.hpp"
```

6.83 Sources/libflux/f_rusanov.cpp File Reference

Rusanov flux.

```
#include "f_rusanov.hpp"
```

6.83.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.84 Sources/libflux/flux.cpp File Reference

Numerical flux.

```
#include "flux.hpp"
```

6.84.1 Detailed Description

Numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.85 Sources/libfrictions/choice_friction.cpp File Reference

Choice of friction law.

```
#include "choice_friction.hpp"
```

6.85.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.86 Sources/libfrictions/fr_darcy_weisbach.cpp File Reference

Darcy-Weisbach law.

```
#include "fr_darcy_weisbach.hpp"
```

6.86.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.87 Sources/libfrictions/fr_laminar.cpp File Reference

Laminar law.

```
#include "fr_laminar.hpp"
```

6.87.1 Detailed Description

Laminar law.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2014-2015)

Version

1.06.00

Date

2015-02-19

Friction law: laminar.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.88 Sources/libfrictions/fr_manning.cpp File Reference

Manning law.

```
#include "fr_manning.hpp"
```

6.88.1 Detailed Description

Manning law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Manning.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.89 Sources/libfrictions/friction.cpp File Reference

Friction law

```
#include "friction.hpp"
```

6.89.1 Detailed Description

Friction law

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the friction laws.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.90 Sources/libfrictions/no_friction.cpp File Reference

No friction.

```
#include "no_friction.hpp"
```

6.90.1 Detailed Description

No friction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: does no computation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.91 Sources/libinitializations/choice_init_huv.cpp File Reference

Choice of initialization for h, u and v.

```
#include "choice_init_huv.hpp"
```

6.91.1 Detailed Description

Choice of initialization for h, u and v.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.92 Sources/libinitializations/choice_init_topo.cpp File Reference

Choice of initialization for the topography.

```
#include "choice_init_topo.hpp"
```

6.92.1 Detailed Description

Choice of initialization for the topography.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.93 Sources/libinitializations/huv_generated.cpp File Reference

No water configuration.

```
#include "huv_generated.hpp"
```

6.93.1 Detailed Description

No water configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.94 Sources/libinitializations/huv_generated_radial_dam_dry.cpp File Reference

Dry radial dam break configuration.

```
#include "huv_generated_radial_dam_dry.hpp"
```

6.94.1 Detailed Description

Dry radial dam break configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Initialization of the water height and of the velocity: case of a radial dam break on a dry domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.95 Sources/libinitializations/huv_generated_radial_dam_wet.cpp File Reference

Wet radial dam break configuration.

```
#include "huv_generated_radial_dam_wet.hpp"
```

6.95.1 Detailed Description

Wet radial dam break configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Initialization of the water height and of the velocity: case of a radial dam break on a wet domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.96 Sources/libinitializations/huv_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "huv_generated_thacker.hpp"
```

6.96.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.97 Sources/libinitializations/huv_read.cpp File Reference

File configuration.

```
#include "huv_read.hpp"
```

6.97.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.98 Sources/libinitializations/initialization_huv.cpp File Reference

Initialization of h, u and v

```
#include "initialization_huv.hpp"
```

6.98.1 Detailed Description

Initialization of h, u and v

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the water height and of the velocity.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.99 Sources/libinitializations/initialization_topo.cpp File Reference

Initialization of z

```
#include "initialization_topo.hpp"
```

6.99.1 Detailed Description

Initialization of z

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the topography.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.100 Sources/libinitializations/topo_generated_flat.cpp File Reference

Flat configuration.

```
#include "topo_generated_flat.hpp"
```

6.100.1 Detailed Description

Flat configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the topography is flat, its value is 0.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.101 Sources/libinitializations/topo_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "topo_generated_thacker.hpp"
```

6.101.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: topography with a shape of a paraboloid of revolution for Thacker's Benchmark.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.102 Sources/libinitializations/topo_read.cpp File Reference

File configuration.

```
#include "topo_read.hpp"
```

6.102.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the values are read in a file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.103 Sources/liblimitations/choice_limiter.cpp File Reference

Choice of slope limiter.

```
#include "choice_limiter.hpp"
```

6.103.1 Detailed Description

Choice of slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.104 Sources/liblimitations/limiter.cpp File Reference

Slope limiter.

```
#include "limiter.hpp"
```

6.104.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the slope limiters.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.105 Sources/liblimitations/minmod.cpp File Reference

Minmod limiter

```
#include "minmod.hpp"
```

6.105.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: minmod.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.106 Sources/liblimitations/vanalbada.cpp File Reference

Van Albada limiter.

```
#include "vanalbada.hpp"
```

6.106.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Albada.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.107 Sources/liblimitations/vanleer.cpp File Reference

Van Leer limiter.

```
#include "vanleer.hpp"
```

6.107.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Leer.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.108 Sources/libparameters/parameters.cpp File Reference

Gets parameters.

```
#include "parameters.hpp"
```

6.108.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2011-2017)

Frederic Darboux frederic.darboux@orleans.inra.fr (2014)

Version

1.07.01

Date

2017-09-11

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 - BRGM (France)

6.109 Sources/libparser/parser.cpp File Reference

Parser

```
#include "parser.hpp"
```

6.109.1 Detailed Description

Parser

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2010-2015)

Version

1.06.00

Date

2015-02-19

Reads the input file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.110 Sources/librain_infiltration/choice_infiltration.cpp File Reference

Choice of infiltration law.

```
#include "choice_infiltration.hpp"
```

6.110.1 Detailed Description

Choice of infiltration law.

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter: calls the chosen infiltration law.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.111 Sources/librain_infiltration/choice_rain.cpp File Reference

Choice of initialization for the rain.

```
#include "choice_rain.hpp"
```

6.111.1 Detailed Description

Choice of initialization for the rain.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.112 Sources/librain_infiltration/greenampt.cpp File Reference

Green-Ampt law.

```
#include "greenampt.hpp"
```

6.112.1 Detailed Description

Green-Ampt law.

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.113 Sources/librain_infiltration/infiltration.cpp File Reference

Infiltration law

```
#include "infiltration.hpp"
```

6.113.1 Detailed Description

Infiltration law

Author

Marie Rousseau M.Rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.01

Date

2015-03-10

Common part for the infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.114 Sources/librain_infiltration/no_infiltration.cpp File Reference

No infiltration.

```
#include "no_infiltration.hpp"
```

6.114.1 Detailed Description

No infiltration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.115 Sources/librain_infiltration/no_rain.cpp File Reference

No rain.

```
#include "no_rain.hpp"
```

6.115.1 Detailed Description

No rain.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Rain: there is no rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.116 Sources/librain_infiltration/rain.cpp File Reference

Rain

```
#include "rain.hpp"
```

6.116.1 Detailed Description

Rain

Author

Marie Rousseau ma.rousseau@brgm.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the initialization of the rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.117 Sources/librain_infiltration/rain_generated.cpp File Reference

Constant rain configuration.

```
#include "rain_generated.hpp"
```

6.117.1 Detailed Description

Constant rain configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the value is equals to 0.00001 m/s = 36 mm/h, constant during the simulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.118 Sources/librain_infiltration/rain_read.cpp File Reference

File configuration.

```
#include "rain_read.hpp"
```

6.118.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the values are read in a file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.119 Sources/libreconstructions/choice_reconstruction.cpp File Reference

Choice of reconstruction.

```
#include "choice_reconstruction.hpp"
```

6.119.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

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 - BRGM (France)

6.120 Sources/libreconstructions/eno.cpp File Reference

ENO reconstruction

```
#include "eno.hpp"
```

6.120.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Linear reconstruction: ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.121 Sources/libreconstructions/eno_mod.cpp File Reference

Modified ENO reconstruction.

```
#include "eno_mod.hpp"
```

6.121.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.122 Sources/libreconstructions/hydrostatic.cpp File Reference

Hydrostatic reconstruction

```
#include "hydrostatic.hpp"
```

6.122.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.123 Sources/libreconstructions/muscl.cpp File Reference

MUSCL reconstruction

```
#include "muscl.hpp"
```

6.123.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2016-12-22

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.124 Sources/libreconstructions/reconstruction.cpp File Reference

Reconstruction

```
#include "reconstruction.hpp"
```

6.124.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the reconstructions.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.125 Sources/libsave/choice_output.cpp File Reference

Choice of output format.

```
#include "choice_output.hpp"
```

6.125.1 Detailed Description

Choice of output format.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

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 - BRGM (France)

6.126 Sources/libsave/choice_save_specific_points.cpp File Reference

Choice of the output of the specific points.

```
#include "choice_save_specific_points.hpp"
```

6.126.1 Detailed Description

Choice of the output of the specific points.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

From the value of the corresponding parameter, calls the savings at the specific points.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.127 Sources/libsave/gnuplot.cpp File Reference

Gnuplot output

```
#include "gnuplot.hpp"
```

6.127.1 Detailed Description

Gnuplot output

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

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 - BRGM (France)

6.128 Sources/libsave/no_evolution_file.cpp File Reference

No output.

```
#include "no_evolution_file.hpp"
```

6.128.1 Detailed Description

No output.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

No output files with time evolution

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.129 Sources/libsave/no_save.cpp File Reference

No output.

```
#include "no_save.hpp"
```

6.129.1 Detailed Description

No output.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

No specific point to save

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.130 Sources/libsave/one_point.cpp File Reference

One point output.

```
#include "one_point.hpp"
```

6.130.1 Detailed Description

One point output.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-16

Output format: optimized for Gnuplot .

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.131 Sources/libsave/output.cpp File Reference

Output format

```
#include "output.hpp"
```

6.131.1 Detailed Description

Output format

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

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 - BRGM (France)

6.132 Sources/libsave/save_specific_points.cpp File Reference

Specific points to save.

```
#include "save_specific_points.hpp"
```

6.132.1 Detailed Description

Specific points to save.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-03

Common part for all the outputs of the specific points.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.133 Sources/libsave/several_points.cpp File Reference

Saving several specific points.

```
#include "several_points.hpp"
```

6.133.1 Detailed Description

Saving several specific points.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2017)

Version

1.07.01

Date

2017-02-16

Output format: optimized for Gnuplot (for hu_specific_points.dat).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.134 Sources/libsave/vtk_out.cpp File Reference

VTK output

```
#include "vtk_out.hpp"
```

6.134.1 Detailed Description

VTK output

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-02-28

Output format: optimized for software compatible with vtk format (example: paraview).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.135 Sources/libschemas/choice_scheme.cpp File Reference

Choice of numerical scheme.

```
#include "choice_scheme.hpp"
```

6.135.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical scheme.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.136 Sources/libschemas/order1.cpp File Reference

Order 1 scheme.

```
#include "order1.hpp"
```

6.136.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-01-05

Numerical scheme: at order 1.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.137 Sources/libschemas/order2.cpp File Reference

Order 2 scheme.

```
#include "order2.hpp"
```

6.137.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-01-05

Numerical scheme: at order 2.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - BRGM (France)

6.138 Sources/libschemas/scheme.cpp File Reference

Numerical scheme.

```
#include "scheme.hpp"
```

6.138.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Christian Laguerre christian.laguerre@math.cnrs.fr (2012-2015)

Version

1.07.01

Date

2017-12-13

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 - BRGM (France)

6.139 Tools/ConvertFormat/asc2xyz.c File Reference

ArcGIS ASCII -> FullSWOF_2D XYZ.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <float.h>
#include <ctype.h>
```

Macros

- #define [ASC_HEADER_NBLINE](#) 6
- #define [TRUE](#) 1
- #define [FALSE](#) 0
- #define [LENGTH_OF_LINE](#) 120

Typedefs

- typedef float [OUTDATA_TYPE](#)

Functions

- int [stricmp](#) (char const *a, char const *b)
- int [main](#) (int argc, char *argv[])

6.139.1 Detailed Description

ArcGIS ASCII -> FullSWOF_2D XYZ.

Author

Frederic Darboux frederic.darboux@inra.fr (2017)

Version

0.00.04

Date

2017-12-08

Converts an ArcGIS ASCII file to a XYZ file for FullSWOF_2D.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA

6.139.2 Macro Definition Documentation

#define ASC_HEADER_NBLINE 6

Definition at line 72 of file asc2xyz.c.

#define FALSE 0

Definition at line 74 of file asc2xyz.c.

#define LENGTH_OF_LINE 120

Definition at line 76 of file asc2xyz.c.

#define TRUE 1

Definition at line 73 of file asc2xyz.c.

6.139.3 Typedef Documentation**typedef float OUTDATA_TYPE**

Definition at line 78 of file asc2xyz.c.

6.139.4 Function Documentation**int main (int *argc*, char * *argv* [])**

Definition at line 84 of file asc2xyz.c.

int strcmp (char const * *a*, char const * *b*)

Definition at line 246 of file asc2xyz.c.

6.140 Tools/ConvertFormat/xyz2asc.c File Reference

FullSWOF_2D XYZ -> ArcGIS ASCII.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <float.h>
#include <ctype.h>
```

Macros

- #define FALSE 0
- #define TRUE 1
- #define ASCHEADER "#ASCGRID"
- #define ASCHEADER_NBCHAR 9
- #define ASCHEADER_NBLINE 6
- #define LENGTH_OF_LINE 120

Typedefs

- typedef float OUTDATA_TYPE

Functions

- int `stricmp` (char const *a, char const *b)
- void `headerformat` (void)
- int `main` (int argc, char *argv[])

6.140.1 Detailed Description

FullSWOF_2D XYZ -> ArcGIS ASCII.

Author

Frederic Darboux frederic.darboux@inra.fr (2016-2017)

Version

0.00.06

Date

2017-10-20

Converts a XYZ file for FullSWOF_2D to an ArcGIS ASCII file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA

6.140.2 Macro Definition Documentation

#define ASCHEADER "#ASCGRID"

Definition at line 73 of file xyz2asc.c.

#define ASCHEADER_NBCHAR 9

Definition at line 74 of file xyz2asc.c.

#define ASCHEADER_NBLINE 6

Definition at line 75 of file xyz2asc.c.

#define FALSE 0

Definition at line 71 of file xyz2asc.c.

#define LENGTH_OF_LINE 120

Definition at line 77 of file xyz2asc.c.

#define TRUE 1

Definition at line 72 of file xyz2asc.c.

6.140.3 Typedef Documentation

typedef float OUTDATA_TYPE

Definition at line 79 of file xyz2asc.c.

6.140.4 Function Documentation

void headerformat (void)

Definition at line 245 of file xyz2asc.c.

int main (int *argc*, char * *argv*[])

Definition at line 88 of file xyz2asc.c.

int strcmp (char const * *a*, char const * *b*)

Definition at line 236 of file xyz2asc.c.

6.141 Tools/ExtractWindow/cropxyz.c File Reference

Crop a FullSWOF_2D XYZ file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Macros

- #define [LENGTH_OF_LINE](#) 120

Functions

- int [strcmp](#) (char const *a, char const *b)
- int [main](#) (int argc, char *argv[])

6.141.1 Detailed Description

Crop a FullSWOF_2D XYZ file.

Author

Frederic Darboux frederic.darboux@inra.fr (2017)

Version

0.00.02

Date

2017-10-19

Crop a XYZ file for FullSWOF_2D, i.e. extract a subset of rectangular shape.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA

6.141.2 Macro Definition Documentation

#define LENGTH_OF_LINE 120

Definition at line 66 of file cropxyz.c.

6.141.3 Function Documentation

int main (int *argc*, char * *argv*[])

Definition at line 72 of file cropxyz.c.

int stricmp (char const * *a*, char const * *b*)

Definition at line 246 of file asc2xyz.c.

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. [82](#), [84](#)
- 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](#). [88](#)
- P. Batten, N. Clarke, C. Lambert, and D. M. Causon. On the choice of wavespeeds for the HLLC Riemann solver. *SIAM Journal on Scientific Computing*, 18(6):1553–1570, 1997. doi:[10.1137/S1064827593260140](#). [55](#)
- 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/b93802](#). [50](#), [52](#), [54](#), [57](#), [61](#)
- F. Bouchut. Chapter 4 efficient numerical finite volume schemes for shallow water models. In V. Zeitlin, editor, *Nonlinear Dynamics of Rotating Shallow Water: Methods and Advances*, volume 2 of *Edited Series on Advances in Nonlinear Science and Complexity*, pages 189 – 256. Elsevier Science, 2007. doi:[10.1016/S1574-6909\(06\)02004-1](#). URL <http://www.sciencedirect.com/science/article/B8JG3-4PS6TNS-6/2/f4c3dbcf476626c1cb6f353e9bc66cc9>. [50](#), [52](#), [98](#)
- M.-O. Bristeau and B. Coussin. Boundary conditions for the shallow water equations solved by kinetic schemes. Technical Report RR-4282, INRIA, 2001. [1](#), [26](#)
- O. Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles*. PhD thesis, Université d'Orléans, Orléans, France, July 2010. <http://tel.archives-ouvertes.fr/tel-00531377/en/>. [184](#), [235](#)
- 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. [82](#), [84](#)
- A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy. Some results on uniformly high-order accurate essentially nonoscillatory schemes. *Applied Numerical Mathematics*, 2(3-5):347 – 377, 1986. ISSN 0168-9274. doi:[10.1016/0168-9274\(86\)90039-5](#). URL <http://www.sciencedirect.com/science/article/B6TYD-45GVV8T-J/2/830ca2dce5e3fb34ace9fa53ae5ab76b>. Special Issue in Honor of Milt Rose's Sixtieth Birthday. [50](#)
- 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. [50](#)
- A. Y. Le Roux. Conditions aux limites et problèmes hyperboliques : un point de vue numérique. Available in the document of the Conférence at IHP-Paris, 2001. [1](#), [26](#)
- C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439 – 471, 1988. ISSN 0021-9991. doi:[10.1016/0021-9991\(88\)90177-5](#). URL <http://www.sciencedirect.com/science/article/B6WHY-4DD1T6W-MM/2/bef99e4b67bd7132c8af1984c34ce57e>. [50](#)
- 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](#). [66](#), [72](#)

- 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>. 85, 177
- E. F. Toro. *Shock-Capturing Methods for Free-Surface Shallow Flows*. John Wiley & Sons, 2001. 57, 59
- B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101 – 136, 1979. ISSN 0021-9991. doi:[10.1016/0021-9991\(79\)90145-1](https://doi.org/10.1016/0021-9991(79)90145-1). URL <http://www.sciencedirect.com/science/article/B6WHY-4DD1N8T-C5/2/9b051d1cfcff715a3d0f4b7b7b0397cc>. 98

Index

- ~Bc_Neumann
 - Bc_Neumann, [21](#)
- ~Bc_imp_discharge
 - Bc_imp_discharge, [17](#)
- ~Bc_imp_height
 - Bc_imp_height, [20](#)
- ~Bc_periodic
 - Bc_periodic, [23](#)
- ~Bc_wall
 - Bc_wall, [25](#)
- ~Boundary_condition
 - Boundary_condition, [27](#)
- ~Choice_condition
 - Choice_condition, [29](#)
- ~Choice_flux
 - Choice_flux, [33](#)
- ~Choice_friction
 - Choice_friction, [35](#)
- ~Choice_infiltration
 - Choice_infiltration, [37](#)
- ~Choice_init_huv
 - Choice_init_huv, [39](#)
- ~Choice_init_topo
 - Choice_init_topo, [40](#)
- ~Choice_limiter
 - Choice_limiter, [41](#)
- ~Choice_output
 - Choice_output, [42](#)
- ~Choice_rain
 - Choice_rain, [45](#)
- ~Choice_reconstruction
 - Choice_reconstruction, [46](#)
- ~Choice_save_specific_points
 - Choice_save_specific_points, [48](#)
- ~Choice_scheme
 - Choice_scheme, [49](#)
- ~ENO
 - ENO, [50](#)
- ~ENO_mod
 - ENO_mod, [52](#)
- ~F_HLL
 - F_HLL, [53](#)
- ~F_HLL2
 - F_HLL2, [55](#)
- ~F_HLLC
 - F_HLLC, [57](#)
- ~F_HLLC2
 - F_HLLC2, [59](#)
- ~F_Rusanov
 - F_Rusanov, [61](#)
- ~Flux
 - Flux, [62](#)
- ~Fr_Darcy_Weisbach
 - Fr_Darcy_Weisbach, [66](#)
- ~Fr_Laminar
 - Fr_Laminar, [69](#)
- ~Fr_Manning
 - Fr_Manning, [72](#)
- ~Friction
 - Friction, [74](#)
- ~Gnuplot
 - Gnuplot, [77](#)
- ~GreenAmpt
 - GreenAmpt, [78](#)
- ~Huv_generated
 - Huv_generated, [81](#)
- ~Huv_generated_Radial_Dam_dry
 - Huv_generated_Radial_Dam_dry, [83](#)
- ~Huv_generated_Radial_Dam_wet
 - Huv_generated_Radial_Dam_wet, [84](#)
- ~Huv_generated_Thacker
 - Huv_generated_Thacker, [85](#)
- ~Huv_read
 - Huv_read, [86](#)
- ~Hydrostatic
 - Hydrostatic, [87](#)
- ~Infiltration
 - Infiltration, [90](#)
- ~Initialization_huv
 - Initialization_huv, [92](#)
- ~Initialization_topo
 - Initialization_topo, [94](#)
- ~Limiter
 - Limiter, [95](#)
- ~MUSCL
 - MUSCL, [98](#)
- ~Minmod
 - Minmod, [96](#)
- ~No_Evolution_File
 - No_Evolution_File, [101](#)
- ~No_Friction
 - No_Friction, [102](#)

- ~No_Infiltration
 - No_Infiltration, 104
- ~No_Rain
 - No_Rain, 105
- ~No_save
 - No_save, 106
- ~One_point
 - One_point, 107
- ~Order1
 - Order1, 110
- ~Order2
 - Order2, 111
- ~Output
 - Output, 113
- ~Parameters
 - Parameters, 123
- ~Parser
 - Parser, 149
- ~Rain
 - Rain, 151
- ~Rain_generated
 - Rain_generated, 152
- ~Rain_read
 - Rain_read, 153
- ~Reconstruction
 - Reconstruction, 155
- ~Save_specific_points
 - Save_specific_points, 157
- ~Scheme
 - Scheme, 161
- ~Several_points
 - Several_points, 174
- ~Topo_generated_Thacker
 - Topo_generated_Thacker, 177
- ~Topo_generated_flat
 - Topo_generated_flat, 175
- ~Topo_read
 - Topo_read, 178
- ~VanAlbada
 - VanAlbada, 179
- ~VanLeer
 - VanLeer, 180
- ~Vtk_Out
 - Vtk_Out, 182
- ASC_HEADER_NBLINE
 - asc2xyz.c, 269
- ASCHEADER
 - xyz2asc.c, 271
- ASCHEADER_NBCHAR
 - xyz2asc.c, 271
- ASCHEADER_NBLINE
 - xyz2asc.c, 271
- allocation
 - Scheme, 161
- amortENO
 - Parameters, 141
- asc2xyz.c
 - ASC_HEADER_NBLINE, 269
 - FALSE, 269
 - LENGTH_OF_LINE, 270
 - main, 270
 - OUTDATA_TYPE, 270
 - stricmp, 270
 - TRUE, 270
- B_IMP_H
 - Scheme, 163
- B_IMP_Q
 - Scheme, 163
- B_choice_bound
 - Scheme, 163
- BC_IMP_DISCHARGE_HPP
 - bc_imp_discharge.hpp, 184
- Bbound
 - Parameters, 141
 - Scheme, 164
- Bbound_type
 - Parameters, 141
 - Scheme, 164
- Bc_Neumann, 20
 - ~Bc_Neumann, 21
 - Bc_Neumann, 21
 - calcul, 21
- Bc_imp_discharge, 15
 - ~Bc_imp_discharge, 17
 - Bc_imp_discharge, 16
 - calcul, 17
 - getValueOfPolynomial, 18
 - getValueofDerivativeOfPolynomial, 17
 - newtonSolver, 18
- bc_imp_discharge.hpp
 - BC_IMP_DISCHARGE_HPP, 184
- Bc_imp_height, 19
 - ~Bc_imp_height, 20
 - Bc_imp_height, 19
 - calcul, 20
- Bc_periodic, 22
 - ~Bc_periodic, 23
 - Bc_periodic, 23
 - calcul, 23
- Bc_wall, 24
 - ~Bc_wall, 25
 - Bc_wall, 24
 - calcul, 25
- bottom_imp_discharge
 - Parameters, 141
- bottom_imp_h

- Parameters, 141
- bottom_times_files
 - Parameters, 141
 - Scheme, 164
- boundaries_flux
 - Choice_output, 42
 - Output, 113
- boundaries_flux_BT
 - Choice_output, 43
 - Output, 113
- boundaries_flux_LR
 - Choice_output, 43
 - Output, 114
- boundary
 - Scheme, 161
- Boundary_condition, 25
 - ~Boundary_condition, 27
 - Boundary_condition, 26
 - calcul, 27
 - Choice_Bbound, 27
 - Choice_Lbound, 27
 - Choice_Rbound, 27
 - Choice_Tbound, 28
 - get_hbound, 27
 - get_unormbound, 27
 - get_utanbound, 27
 - hbound, 28
 - NXCELL, 28
 - NYCELL, 28
 - unormbound, 28
 - unormfix, 28
 - utanbound, 28
- CFL_FIX
 - Scheme, 164
- CHOICE_CONDITION_HPP
 - choice_condition.hpp, 188
- CHOICE_FLUX_HPP
 - choice_flux.hpp, 189
- CHOICE_FRICTION_HPP
 - choice_friction.hpp, 194
- CHOICE_INFILTRATION_HPP
 - choice_infiltration.hpp, 214
- CHOICE_INIT_HUV_HPP
 - choice_init_huv.hpp, 198
- CHOICE_INIT_TOPO_HPP
 - choice_init_topo.hpp, 199
- CHOICE_LIMITER_HPP
 - choice_limiter.hpp, 207
- CHOICE_OUTPUT_HPP
 - choice_output.hpp, 224
- CHOICE_RAIN_HPP
 - choice_rain.hpp, 215
- CHOICE_RECONSTRUCTION
 - choice_reconstruction.hpp, 220
- CHOICE_SAVE_SPECIFIC_POINTS_HPP
 - choice_save_specific_points.hpp, 225
- CHOICE_SCHEME_HPP
 - choice_scheme.hpp, 231
- CONST_CFL_X
 - misc.hpp, 211
- CONST_CFL_Y
 - misc.hpp, 211
- calcul
 - Bc_Neumann, 21
 - Bc_imp_discharge, 17
 - Bc_imp_height, 20
 - Bc_periodic, 23
 - Bc_wall, 25
 - Boundary_condition, 27
 - Choice_condition, 29
 - Choice_flux, 33
 - Choice_friction, 35
 - Choice_infiltration, 38
 - Choice_limiter, 41
 - Choice_reconstruction, 46
 - Choice_scheme, 49
 - ENO, 50
 - ENO_mod, 52
 - F_HLL, 54
 - F_HLL2, 55
 - F_HLLC, 57
 - F_HLLC2, 59
 - F_Rusanov, 61
 - Flux, 63
 - Fr_Darcy_Weisbach, 66
 - Fr_Laminar, 69
 - Fr_Manning, 72
 - Friction, 74
 - GreenAmpt, 79
 - Hydrostatic, 88
 - Infiltration, 90
 - Limiter, 95
 - MUSCL, 98
 - Minmod, 97
 - No_Friction, 102
 - No_Infiltration, 104
 - Order1, 110
 - Order2, 111
 - Reconstruction, 155
 - Scheme, 162
 - VanAlbada, 179
 - VanLeer, 180
- calculSf
 - Choice_friction, 36
 - Fr_Darcy_Weisbach, 66
 - Fr_Laminar, 69

- Fr_Manning, [72](#)
- Friction, [74](#)
- No_Friction, [102](#)
- capacity
 - GreenAmpt, [80](#)
- cfl
 - Flux, [63](#)
- cfl_fix
 - Parameters, [141](#)
- check_vol
 - Choice_output, [43](#)
 - Output, [114](#)
- Choice_Bbound
 - Boundary_condition, [27](#)
- Choice_Lbound
 - Boundary_condition, [27](#)
- Choice_Rbound
 - Boundary_condition, [27](#)
- Choice_Tbound
 - Boundary_condition, [28](#)
- Choice_condition, [28](#)
 - ~Choice_condition, [29](#)
 - calcul, [29](#)
 - Choice_condition, [29](#)
 - get_hbound, [31](#)
 - get_unormbound, [31](#)
 - get_utanbound, [31](#)
 - setChoice, [31](#)
 - setXY, [32](#)
- choice_condition.hpp
 - CHOICE_CONDITION_HPP, [188](#)
- Choice_dt_specific_points
 - Parameters, [141](#)
- Choice_flux, [32](#)
 - ~Choice_flux, [33](#)
 - calcul, [33](#)
 - Choice_flux, [33](#)
 - get_cfl, [33](#)
 - get_f1, [33](#)
 - get_f2, [34](#)
 - get_f3, [34](#)
 - set_tx, [34](#)
- choice_flux.hpp
 - CHOICE_FLUX_HPP, [189](#)
- Choice_friction, [34](#)
 - ~Choice_friction, [35](#)
 - calcul, [35](#)
 - calculSf, [36](#)
 - Choice_friction, [35](#)
 - get_Sf1, [36](#)
 - get_Sf2, [36](#)
 - get_q1mod, [36](#)
 - get_q2mod, [36](#)
- choice_friction.hpp
 - CHOICE_FRICTION_HPP, [194](#)
- Choice_infiltration, [37](#)
 - ~Choice_infiltration, [37](#)
 - calcul, [38](#)
 - Choice_infiltration, [37](#)
 - get_Vin, [38](#)
 - get_hmod, [38](#)
- choice_infiltration.hpp
 - CHOICE_INFILTRATION_HPP, [214](#)
- Choice_init_huv, [38](#)
 - ~Choice_init_huv, [39](#)
 - Choice_init_huv, [39](#)
 - initialization, [39](#)
- choice_init_huv.hpp
 - CHOICE_INIT_HUV_HPP, [198](#)
- Choice_init_topo, [39](#)
 - ~Choice_init_topo, [40](#)
 - Choice_init_topo, [40](#)
 - initialization, [40](#)
- choice_init_topo.hpp
 - CHOICE_INIT_TOPO_HPP, [199](#)
- Choice_limiter, [40](#)
 - ~Choice_limiter, [41](#)
 - calcul, [41](#)
 - Choice_limiter, [41](#)
 - get_rec, [41](#)
- choice_limiter.hpp
 - CHOICE_LIMITER_HPP, [207](#)
- Choice_output, [41](#)
 - ~Choice_output, [42](#)
 - boundaries_flux, [42](#)
 - boundaries_flux_BT, [43](#)
 - boundaries_flux_LR, [43](#)
 - check_vol, [43](#)
 - Choice_output, [42](#)
 - final, [43](#)
 - initial, [44](#)
 - result, [44](#)
 - write, [44](#)
- choice_output.hpp
 - CHOICE_OUTPUT_HPP, [224](#)
- Choice_points
 - Parameters, [141](#)
- Choice_rain, [45](#)
 - ~Choice_rain, [45](#)
 - Choice_rain, [45](#)
 - rain_func, [45](#)
- choice_rain.hpp
 - CHOICE_RAIN_HPP, [215](#)
- Choice_reconstruction, [45](#)
 - ~Choice_reconstruction, [46](#)
 - calcul, [46](#)

- Choice_reconstruction, [46](#)
- choice_reconstruction.hpp
 - CHOICE_RECONSTRUCTION, [220](#)
- Choice_save_specific_points, [47](#)
 - ~Choice_save_specific_points, [48](#)
 - Choice_save_specific_points, [48](#)
 - save, [48](#)
- choice_save_specific_points.hpp
 - CHOICE_SAVE_SPECIFIC_POINTS_HPP, [225](#)
- Choice_scheme, [48](#)
 - ~Choice_scheme, [49](#)
 - calcul, [49](#)
 - Choice_scheme, [49](#)
- choice_scheme.hpp
 - CHOICE_SCHEME_HPP, [231](#)
- column
 - Save_specific_points, [157](#)
- cpu_time
 - Scheme, [164](#)
- cropxyz.c
 - LENGTH_OF_LINE, [273](#)
 - main, [273](#)
 - stricmp, [273](#)
- cur_time
 - Scheme, [164](#)
- DT_FIX
 - Scheme, [165](#)
- DT_SPECIFIC_POINTS
 - Save_specific_points, [157](#)
- DX
 - Friction, [75](#)
 - Infiltration, [91](#)
 - Initialization_huv, [93](#)
 - Initialization_topo, [94](#)
 - Output, [115](#)
 - Rain, [151](#)
 - Save_specific_points, [157](#)
 - Scheme, [165](#)
- DY
 - Friction, [75](#)
 - Infiltration, [91](#)
 - Initialization_huv, [93](#)
 - Initialization_topo, [94](#)
 - Output, [115](#)
 - Rain, [151](#)
 - Save_specific_points, [157](#)
 - Scheme, [165](#)
- deallocation
 - Scheme, [162](#)
- delta_z1
 - Reconstruction, [155](#)
- delta_z2
 - Reconstruction, [155](#)
- delz1
 - Scheme, [164](#)
- delz2
 - Scheme, [164](#)
- delzc1
 - Scheme, [164](#)
- delzc2
 - Scheme, [164](#)
- dt1
 - Scheme, [164](#)
- dt_first
 - Scheme, [165](#)
- dt_fix
 - Parameters, [141](#)
- dt_max
 - Scheme, [165](#)
- dt_output
 - Scheme, [165](#)
- dt_specific_points
 - Parameters, [141](#)
- dtheta_NF
 - Parameters, [142](#)
- dtheta_coef
 - Parameters, [141](#)
- dtheta_init
 - Parameters, [142](#)
- dtheta_namefile
 - Parameters, [142](#)
- dx
 - Parameters, [142](#)
- dy
 - Parameters, [142](#)
- ENO, [49](#)
 - ~ENO, [50](#)
 - calcul, [50](#)
 - ENO, [50](#)
- ENO_mod, [51](#)
 - ~ENO_mod, [52](#)
 - calcul, [52](#)
 - ENO_mod, [52](#)
- EPSILON
 - misc.hpp, [211](#)
- end
 - Scheme, [165](#)
- f1
 - Flux, [63](#)
 - Scheme, [165](#)
- f2
 - Flux, [64](#)
 - Scheme, [165](#)
- f3
 - Flux, [64](#)

- Scheme, 165
- F_HLL, 53
 - ~F_HLL, 53
 - calcul, 54
 - F_HLL, 53
- F_HLL2, 55
 - ~F_HLL2, 55
 - calcul, 55
 - F_HLL2, 55
- F_HLLC, 56
 - ~F_HLLC, 57
 - calcul, 57
 - F_HLLC, 57
- F_HLLC2, 58
 - ~F_HLLC2, 59
 - calcul, 59
 - F_HLLC2, 59
- F_Rusanov, 60
 - ~F_Rusanov, 61
 - calcul, 61
 - F_Rusanov, 61
- FALSE
 - asc2xyz.c, 269
 - xyz2asc.c, 271
- FRICCOEF
 - Scheme, 166
- fill_array
 - Parameters, 123
- fill_array_bc_inhomogeneous
 - Parameters, 124
- final
 - Choice_output, 43
 - Output, 114
- Flux, 61
 - ~Flux, 62
 - calcul, 63
 - cfl, 63
 - f1, 63
 - f2, 64
 - f3, 64
 - Flux, 62
 - get_cfl, 63
 - get_f1, 63
 - get_f2, 63
 - get_f3, 63
 - set_tx, 63
 - tx, 64
- flux
 - Parameters, 142
- Fr
 - Scheme, 165
- Fr_Darcy_Weisbach, 64
 - ~Fr_Darcy_Weisbach, 66
 - calcul, 66
 - calculSf, 66
 - Fr_Darcy_Weisbach, 65
- Fr_Laminar, 67
 - ~Fr_Laminar, 69
 - calcul, 69
 - calculSf, 69
 - Fr_Laminar, 68
- Fr_Manning, 70
 - ~Fr_Manning, 72
 - calcul, 72
 - calculSf, 72
 - Fr_Manning, 71
- fric
 - Parameters, 142
- fric_NF
 - Parameters, 142
- fric_init
 - Parameters, 142
- fric_namefile
 - Parameters, 142
- Fric_tab
 - Friction, 75
- friccoef
 - Parameters, 142
- Friction, 73
 - ~Friction, 74
 - calcul, 74
 - calculSf, 74
 - DX, 75
 - DY, 75
 - Fric_tab, 75
 - Friction, 74
 - get_Sf1, 75
 - get_Sf2, 75
 - get_q1mod, 74
 - get_q2mod, 75
 - NXCELL, 75
 - NYCELL, 75
 - q1mod, 76
 - q2mod, 76
 - Sf1, 76
 - Sf2, 76
- froude_number
 - Scheme, 162
- fs2d_par
 - Scheme, 166
- FullSWOF_2D.cpp
 - main, 234
- g1
 - Scheme, 166
- g2
 - Scheme, 166

- g3
 - Scheme, [166](#)
- GRAV
 - misc.hpp, [211](#)
- GRAV_DEM
 - misc.hpp, [211](#)
- get_Bbound
 - Parameters, [125](#)
- get_Kc_coef
 - Parameters, [129](#)
- get_Kc_init
 - Parameters, [129](#)
- get_KcNameFile
 - Parameters, [130](#)
- get_KcNameFileS
 - Parameters, [130](#)
- get_Ks_coef
 - Parameters, [130](#)
- get_Ks_init
 - Parameters, [130](#)
- get_KsNameFile
 - Parameters, [130](#)
- get_KsNameFileS
 - Parameters, [130](#)
- get_L
 - Parameters, [131](#)
- get_Lbound
 - Parameters, [131](#)
- get_Nxcell
 - Parameters, [132](#)
- get_Nycell
 - Parameters, [132](#)
- get_Psi_coef
 - Parameters, [133](#)
- get_Psi_init
 - Parameters, [133](#)
- get_PsiNameFile
 - Parameters, [134](#)
- get_PsiNameFileS
 - Parameters, [134](#)
- get_Rbound
 - Parameters, [134](#)
- get_Sf1
 - Choice_friction, [36](#)
 - Friction, [75](#)
- get_Sf2
 - Choice_friction, [36](#)
 - Friction, [75](#)
- get_T
 - Parameters, [135](#)
- get_Tbound
 - Parameters, [136](#)
- get_Vin
 - Choice_infiltration, [38](#)
 - Infiltration, [91](#)
- get_amortENO
 - Parameters, [124](#)
- get_bottom_imp_discharge
 - Parameters, [125](#)
- get_bottom_imp_h
 - Parameters, [125](#)
- get_cfl
 - Choice_flux, [33](#)
 - Flux, [63](#)
- get_cffix
 - Parameters, [125](#)
- get_choice_dt_specific_points
 - Parameters, [125](#)
- get_choice_specific_point
 - Parameters, [125](#)
- get_dt_specific_points
 - Parameters, [126](#)
- get_dtfix
 - Parameters, [126](#)
- get_dtheta_coef
 - Parameters, [126](#)
- get_dtheta_init
 - Parameters, [126](#)
- get_dthetaNameFile
 - Parameters, [126](#)
- get_dthetaNameFileS
 - Parameters, [126](#)
- get_dx
 - Parameters, [127](#)
- get_dy
 - Parameters, [127](#)
- get_f1
 - Choice_flux, [33](#)
 - Flux, [63](#)
- get_f2
 - Choice_flux, [34](#)
 - Flux, [63](#)
- get_f3
 - Choice_flux, [34](#)
 - Flux, [63](#)
- get_flux
 - Parameters, [127](#)
- get_fric
 - Parameters, [127](#)
- get_fric_init
 - Parameters, [127](#)
- get_friccoef
 - Parameters, [127](#)
- get_frictionNameFile
 - Parameters, [128](#)
- get_frictionNameFileS
 - Parameters, [128](#)

- Parameters, 128
- get_hbound
 - Boundary_condition, 27
 - Choice_condition, 31
- get_hhydro_l
 - Hydrostatic, 89
- get_hhydro_r
 - Hydrostatic, 89
- get_hmod
 - Choice_infiltration, 38
 - Infiltration, 90
- get_huv
 - Parameters, 128
- get_huvNameFile
 - Parameters, 128
- get_huvNameFileS
 - Parameters, 128
- get_imax_coef
 - Parameters, 128
- get_imax_init
 - Parameters, 129
- get_imaxNameFile
 - Parameters, 129
- get_imaxNameFileS
 - Parameters, 129
- get_inf
 - Parameters, 129
- get_l
 - Parameters, 131
- get_left_imp_discharge
 - Parameters, 131
- get_left_imp_h
 - Parameters, 131
- get_lim
 - Parameters, 131
- get_list_pointNameFile
 - Parameters, 132
- get_list_pointNameFileS
 - Parameters, 132
- get_modifENO
 - Parameters, 132
- get_nbtimes
 - Parameters, 132
- get_order
 - Parameters, 133
- get_output
 - Parameters, 133
- get_outputDirectory
 - Parameters, 133
- get_path_input_directory
 - Parameters, 133
- get_q1mod
 - Choice_friction, 36
- Friction, 74
- get_q2mod
 - Choice_friction, 36
 - Friction, 75
- get_rain
 - Parameters, 134
- get_rainNameFile
 - Parameters, 134
- get_rainNameFileS
 - Parameters, 134
- get_rec
 - Choice_limiter, 41
 - Limiter, 95
 - Parameters, 135
- get_right_imp_discharge
 - Parameters, 135
- get_right_imp_h
 - Parameters, 135
- get_scheme_type
 - Parameters, 135
- get_suffix
 - Parameters, 135
- get_times_files_Bbound
 - Parameters, 136
- get_times_files_Lbound
 - Parameters, 136
- get_times_files_Rbound
 - Parameters, 136
- get_times_files_Tbound
 - Parameters, 136
- get_top_imp_discharge
 - Parameters, 136
- get_top_imp_h
 - Parameters, 137
- get_topo
 - Parameters, 137
- get_topographyNameFile
 - Parameters, 137
- get_topographyNameFileS
 - Parameters, 137
- get_type_Bbound
 - Parameters, 137
- get_type_Lbound
 - Parameters, 137
- get_type_Rbound
 - Parameters, 138
- get_type_Tbound
 - Parameters, 138
- get_unormbound
 - Boundary_condition, 27
 - Choice_condition, 31
- get_utanbound
 - Boundary_condition, 27

- Choice_condition, [31](#)
- get_x_coord
 - Parameters, [138](#)
- get_y_coord
 - Parameters, [138](#)
- get_zcrust_coef
 - Parameters, [138](#)
- get_zcrust_init
 - Parameters, [138](#)
- get_zcrustNameFile
 - Parameters, [139](#)
- get_zcrustNameFileS
 - Parameters, [139](#)
- getValue
 - Parser, [150](#)
- getValueOfPolynomial
 - Bc_imp_discharge, [18](#)
- getValueOfDerivativeOfPolynomial
 - Bc_imp_discharge, [17](#)
- Gnuplot, [76](#)
 - ~Gnuplot, [77](#)
 - Gnuplot, [77](#)
 - write, [77](#)
- GreenAmpt, [78](#)
 - ~GreenAmpt, [78](#)
 - calcul, [79](#)
 - capacity, [80](#)
 - GreenAmpt, [78](#)
- h
 - Scheme, [166](#)
- h1l
 - Scheme, [166](#)
- h1left
 - Scheme, [166](#)
- h1r
 - Scheme, [166](#)
- h1right
 - Scheme, [166](#)
- h2l
 - Scheme, [166](#)
- h2left
 - Scheme, [167](#)
- h2r
 - Scheme, [167](#)
- h2right
 - Scheme, [167](#)
- HE_CA
 - misc.hpp, [211](#)
- hbound
 - Boundary_condition, [28](#)
- headerformat
 - xyz2asc.c, [272](#)
- Headers/libboundaryconditions/bc_imp_discharge.↔
hpp, [183](#)
- Headers/libboundaryconditions/bc_imp_height.hpp,
[184](#)
- Headers/libboundaryconditions/bc_neumann.hpp, [184](#)
- Headers/libboundaryconditions/bc_periodic.hpp, [185](#)
- Headers/libboundaryconditions/bc_wall.hpp, [186](#)
- Headers/libboundaryconditions/boundary_condition.↔
hpp, [186](#)
- Headers/libboundaryconditions/choice_condition.hpp,
[187](#)
- Headers/libflux/choice_flux.hpp, [188](#)
- Headers/libflux/f_hll.hpp, [189](#)
- Headers/libflux/f_hll2.hpp, [190](#)
- Headers/libflux/f_hllc.hpp, [190](#)
- Headers/libflux/f_hllc2.hpp, [191](#)
- Headers/libflux/f_rusanov.hpp, [192](#)
- Headers/libflux/flux.hpp, [192](#)
- Headers/libfrictions/choice_friction.hpp, [193](#)
- Headers/libfrictions/fr_darcy_weisbach.hpp, [194](#)
- Headers/libfrictions/fr_laminar.hpp, [195](#)
- Headers/libfrictions/fr_manning.hpp, [195](#)
- Headers/libfrictions/friction.hpp, [196](#)
- Headers/libfrictions/no_friction.hpp, [196](#)
- Headers/libinitializations/choice_init_huv.hpp, [197](#)
- Headers/libinitializations/choice_init_topo.hpp, [198](#)
- Headers/libinitializations/huv_generated.hpp, [199](#)
- Headers/libinitializations/huv_generated_radial_↔
dam_dry.hpp, [200](#)
- Headers/libinitializations/huv_generated_radial_↔
dam_wet.hpp, [200](#)
- Headers/libinitializations/huv_generated_thacker.hpp,
[201](#)
- Headers/libinitializations/huv_read.hpp, [202](#)
- Headers/libinitializations/initialization_huv.hpp, [202](#)
- Headers/libinitializations/initialization_topo.hpp, [203](#)
- Headers/libinitializations/topo_generated_flat.hpp, [204](#)
- Headers/libinitializations/topo_generated_thacker.hpp,
[204](#)
- Headers/libinitializations/topo_read.hpp, [205](#)
- Headers/liblimitations/choice_limiter.hpp, [206](#)
- Headers/liblimitations/limiter.hpp, [207](#)
- Headers/liblimitations/minmod.hpp, [207](#)
- Headers/liblimitations/vanalbada.hpp, [208](#)
- Headers/liblimitations/vanleer.hpp, [209](#)
- Headers/libparameters/misc.hpp, [209](#)
- Headers/libparameters/parameters.hpp, [212](#)
- Headers/libparser/parser.hpp, [213](#)
- Headers/librain_infiltration/choice_infiltration.hpp, [213](#)
- Headers/librain_infiltration/choice_rain.hpp, [214](#)
- Headers/librain_infiltration/greenampt.hpp, [215](#)
- Headers/librain_infiltration/infiltration.hpp, [216](#)
- Headers/librain_infiltration/no_infiltration.hpp, [216](#)

- Headers/librain_infiltration/no_rain.hpp, 217
- Headers/librain_infiltration/rain.hpp, 218
- Headers/librain_infiltration/rain_generated.hpp, 218
- Headers/librain_infiltration/rain_read.hpp, 219
- Headers/libreconstructions/choice_reconstruction.↵
hpp, 219
- Headers/libreconstructions/eno.hpp, 220
- Headers/libreconstructions/eno_mod.hpp, 221
- Headers/libreconstructions/hydrostatic.hpp, 222
- Headers/libreconstructions/muscl.hpp, 222
- Headers/libreconstructions/reconstruction.hpp, 223
- Headers/libsave/choice_output.hpp, 224
- Headers/libsave/choice_save_specific_points.hpp,
224
- Headers/libsave/gnuplot.hpp, 225
- Headers/libsave/no_evolution_file.hpp, 226
- Headers/libsave/no_save.hpp, 227
- Headers/libsave/one_point.hpp, 227
- Headers/libsave/output.hpp, 228
- Headers/libsave/save_specific_points.hpp, 228
- Headers/libsave/several_points.hpp, 229
- Headers/libsave/vtk_out.hpp, 230
- Headers/libschemas/choice_scheme.hpp, 230
- Headers/libschemas/order1.hpp, 231
- Headers/libschemas/order2.hpp, 232
- Headers/libschemas/scheme.hpp, 233
- height_Vinf_tot
Scheme, 167
- height_of_tot
Scheme, 167
- hl_rec
Hydrostatic, 89
- hmod
Infiltration, 91
- hr_rec
Hydrostatic, 89
- hs
Scheme, 167
- huv_NF
Parameters, 143
- Huv_generated, 81
 - ~Huv_generated, 81
 - Huv_generated, 81
 - initialization, 81
- Huv_generated_Radial_Dam_dry, 82
 - ~Huv_generated_Radial_Dam_dry, 83
 - Huv_generated_Radial_Dam_dry, 82
 - initialization, 83
- Huv_generated_Radial_Dam_wet, 83
 - ~Huv_generated_Radial_Dam_wet, 84
 - Huv_generated_Radial_Dam_wet, 84
 - initialization, 84
- Huv_generated_Thacker, 84
 - ~Huv_generated_Thacker, 85
 - Huv_generated_Thacker, 85
 - initialization, 85
- huv_init
Parameters, 143
- huv_namefile
Parameters, 143
- Huv_read, 85
 - ~Huv_read, 86
 - Huv_read, 86
 - initialization, 86
- Hydrostatic, 87
 - ~Hydrostatic, 87
 - calcul, 88
 - get_hhydro_l, 89
 - get_hhydro_r, 89
 - hl_rec, 89
 - hr_rec, 89
 - Hydrostatic, 87
- IE_CA
misc.hpp, 211
- imax_NF
Parameters, 143
- imax_coef
Parameters, 143
- imax_init
Parameters, 143
- imax_namefile
Parameters, 143
- inf
Parameters, 143
- Infiltration, 89
 - ~Infiltration, 90
 - calcul, 90
 - DX, 91
 - DY, 91
 - get_Vin, 91
 - get_hmod, 90
 - hmod, 91
 - Infiltration, 90
 - NXCELL, 91
 - NYCELL, 91
 - Vin, 91
- initial
 - Choice_output, 44
 - Output, 114
- initialization
 - Choice_init_huv, 39
 - Choice_init_topo, 40
 - Huv_generated, 81
 - Huv_generated_Radial_Dam_dry, 83
 - Huv_generated_Radial_Dam_wet, 84
 - Huv_generated_Thacker, 85

- Huv_read, 86
- Initialization_huv, 92
- Initialization_topo, 94
- Topo_generated_Thacker, 177
- Topo_generated_flat, 176
- Topo_read, 178
- Initialization_huv, 92
 - ~Initialization_huv, 92
 - DX, 93
 - DY, 93
 - initialization, 92
 - Initialization_huv, 92
 - NXCELL, 93
 - NYCELL, 93
- Initialization_topo, 93
 - ~Initialization_topo, 94
 - DX, 94
 - DY, 94
 - initialization, 94
 - Initialization_topo, 94
 - NXCELL, 94
 - NYCELL, 94
- is_Bbound_changed
 - Scheme, 167
- is_Lbound_changed
 - Scheme, 167
- is_Rbound_changed
 - Scheme, 167
- is_Tbound_changed
 - Scheme, 167
- is_coord_in_file_valid
 - Parameters, 139
- Kc_NF
 - Parameters, 144
- Kc_coef
 - Parameters, 143
- Kc_init
 - Parameters, 143
- Kc_namefile
 - Parameters, 143
- Ks_NF
 - Parameters, 144
- Ks_coef
 - Parameters, 144
- Ks_init
 - Parameters, 144
- Ks_namefile
 - Parameters, 144
- L
 - Parameters, 144
- I
 - Parameters, 144
- L_IMP_H
 - Scheme, 168
- L_IMP_Q
 - Scheme, 168
- L_choice_bound
 - Scheme, 167
- LENGTH_OF_LINE
 - asc2xyz.c, 270
 - cropxyz.c, 273
 - xyz2asc.c, 271
- Lbound
 - Parameters, 144
 - Scheme, 168
- Lbound_type
 - Parameters, 144
 - Scheme, 168
- left_imp_discharge
 - Parameters, 144
- left_imp_h
 - Parameters, 144
- left_times_files
 - Parameters, 145
 - Scheme, 168
- lim
 - Parameters, 145
- Limiter, 95
 - ~Limiter, 95
 - calcul, 95
 - get_rec, 95
 - Limiter, 95
 - rec, 96
- limiter
 - Reconstruction, 155
- list_point_NF
 - Parameters, 145
- list_point_namefile
 - Parameters, 145
- list_points_x
 - Parameters, 145
- list_points_y
 - Parameters, 145
- MAX_CFL_X
 - misc.hpp, 211
- MAX_CFL_Y
 - misc.hpp, 211
- MAX_ITER
 - misc.hpp, 211
- MAX_SCAL
 - misc.hpp, 211
- MUSCL, 97
 - ~MUSCL, 98
 - calcul, 98
 - MUSCL, 98

- main
 - asc2xyz.c, 270
 - cropxyz.c, 273
 - FullSWOF_2D.cpp, 234
 - xyz2asc.c, 272
- maincalclflux
 - Scheme, 162
- maincalcscheme
 - Scheme, 163
- max
 - misc.hpp, 211
- min
 - misc.hpp, 211
- Minmod, 96
 - ~Minmod, 96
 - calcul, 97
 - Minmod, 96
- misc.hpp
 - CONST_CFL_X, 211
 - CONST_CFL_Y, 211
 - EPSILON, 211
 - GRAV, 211
 - GRAV_DEM, 211
 - HE_CA, 211
 - IE_CA, 211
 - MAX_CFL_X, 211
 - MAX_CFL_Y, 211
 - MAX_ITER, 211
 - MAX_SCAL, 211
 - max, 211
 - min, 211
 - NB_CHAR, 211
 - RATIO_CLOSE_CELL, 211
 - SCALAR, 212
 - TAB, 212
 - VE_CA, 212
 - VECT, 212
 - VERSION, 212
 - ZERO, 212
- modifENO
 - Parameters, 145
- n
 - Scheme, 168
- NB_CHAR
 - misc.hpp, 211
- NBTIMES
 - Scheme, 168
- NXCELL
 - Boundary_condition, 28
 - Friction, 75
 - Infiltration, 91
 - Initialization_huv, 93
 - Initialization_topo, 94
 - Output, 116
 - Rain, 151
 - Reconstruction, 155
 - Scheme, 168
- NYCELL
 - Boundary_condition, 28
 - Friction, 75
 - Infiltration, 91
 - Initialization_huv, 93
 - Initialization_topo, 94
 - Output, 116
 - Rain, 151
 - Reconstruction, 156
 - Scheme, 168
- namefile_Bound_flux
 - Output, 116
- namefile_Bound_flux_BT
 - Output, 116
- namefile_Bound_flux_LR
 - Output, 116
- namefile_check_volume
 - Output, 116
- namefile_final
 - Output, 116
- namefile_init
 - Output, 116
- namefile_res
 - Output, 116
- nbtimes
 - Parameters, 145
- newtonSolver
 - Bc_imp_discharge, 18
- No_Evolution_File, 99
 - ~No_Evolution_File, 101
 - No_Evolution_File, 100
 - write, 101
- No_Friction, 101
 - ~No_Friction, 102
 - calcul, 102
 - calculSf, 102
 - No_Friction, 102
- No_Infiltration, 103
 - ~No_Infiltration, 104
 - calcul, 104
 - No_Infiltration, 103
- No_Rain, 104
 - ~No_Rain, 105
 - No_Rain, 105
 - rain_func, 105
- No_save, 105
 - ~No_save, 106
 - No_save, 106
 - save, 106

- Nxcell
 - Parameters, 145
- Nycell
 - Parameters, 145
- ORDER
 - Scheme, 168
- OUTDATA_TYPE
 - asc2xyz.c, 270
 - xyz2asc.c, 272
- One_point, 107
 - ~One_point, 107
 - One_point, 107
 - save, 108
- order
 - Parameters, 145
- Order1, 108
 - ~Order1, 110
 - calcul, 110
 - Order1, 109
- Order2, 110
 - ~Order2, 111
 - calcul, 111
 - Order2, 111
- out
 - Scheme, 168
- out_specific_points
 - Scheme, 169
- Output, 111
 - ~Output, 113
 - boundaries_flux, 113
 - boundaries_flux_BT, 113
 - boundaries_flux_LR, 114
 - check_vol, 114
 - DX, 115
 - DY, 115
 - final, 114
 - initial, 114
 - NXCELL, 116
 - NYCELL, 116
 - namefile_Bound_flux, 116
 - namefile_Bound_flux_BT, 116
 - namefile_Bound_flux_LR, 116
 - namefile_check_volume, 116
 - namefile_final, 116
 - namefile_init, 116
 - namefile_res, 116
 - Output, 113
 - outputDirectory, 116
 - result, 115
 - write, 115
- output_directory
 - Parameters, 146
- output_format
 - Parameters, 146
- outputDirectory
 - Output, 116
- p_bottom_times_files
 - Scheme, 169
- p_left_times_files
 - Scheme, 169
- p_right_times_files
 - Scheme, 169
- p_top_times_files
 - Scheme, 169
- Parameters, 117
 - ~Parameters, 123
 - amortENO, 141
 - Bbound, 141
 - Bbound_type, 141
 - bottom_imp_discharge, 141
 - bottom_imp_h, 141
 - bottom_times_files, 141
 - cfl_fix, 141
 - Choice_dt_specific_points, 141
 - Choice_points, 141
 - dt_fix, 141
 - dt_specific_points, 141
 - dtheta_NF, 142
 - dtheta_coef, 141
 - dtheta_init, 142
 - dtheta_namefile, 142
 - dx, 142
 - dy, 142
 - fill_array, 123
 - fill_array_bc_inhomogeneous, 124
 - flux, 142
 - fric, 142
 - fric_NF, 142
 - fric_init, 142
 - fric_namefile, 142
 - friccoef, 142
 - get_Bbound, 125
 - get_Kc_coef, 129
 - get_Kc_init, 129
 - get_KcNameFile, 130
 - get_KcNameFileS, 130
 - get_Ks_coef, 130
 - get_Ks_init, 130
 - get_KsNameFile, 130
 - get_KsNameFileS, 130
 - get_L, 131
 - get_Lbound, 131
 - get_Nxcell, 132
 - get_Nycell, 132
 - get_Psi_coef, 133
 - get_Psi_init, 133

get_PsiNameFile, 134
 get_PsiNameFileS, 134
 get_Rbound, 134
 get_T, 135
 get_Tbound, 136
 get_amortENO, 124
 get_bottom_imp_discharge, 125
 get_bottom_imp_h, 125
 get_cflfix, 125
 get_choice_dt_specific_points, 125
 get_choice_specific_point, 125
 get_dt_specific_points, 126
 get_dtfix, 126
 get_dtheta_coef, 126
 get_dtheta_init, 126
 get_dthetaNameFile, 126
 get_dthetaNameFileS, 126
 get_dx, 127
 get_dy, 127
 get_flux, 127
 get_fric, 127
 get_fric_init, 127
 get_friccoef, 127
 get_frictionNameFile, 128
 get_frictionNameFileS, 128
 get_huv, 128
 get_huvNameFile, 128
 get_huvNameFileS, 128
 get_imax_coef, 128
 get_imax_init, 129
 get_imaxNameFile, 129
 get_imaxNameFileS, 129
 get_inf, 129
 get_l, 131
 get_left_imp_discharge, 131
 get_left_imp_h, 131
 get_lim, 131
 get_list_pointNameFile, 132
 get_list_pointNameFileS, 132
 get_modifENO, 132
 get_nbtimes, 132
 get_order, 133
 get_output, 133
 get_outputDirectory, 133
 get_path_input_directory, 133
 get_rain, 134
 get_rainNameFile, 134
 get_rainNameFileS, 134
 get_rec, 135
 get_right_imp_discharge, 135
 get_right_imp_h, 135
 get_scheme_type, 135
 get_suffix, 135
 get_times_files_Bbound, 136
 get_times_files_Lbound, 136
 get_times_files_Rbound, 136
 get_times_files_Tbound, 136
 get_top_imp_discharge, 136
 get_top_imp_h, 137
 get_topo, 137
 get_topographyNameFile, 137
 get_topographyNameFileS, 137
 get_type_Bbound, 137
 get_type_Lbound, 137
 get_type_Rbound, 138
 get_type_Tbound, 138
 get_x_coord, 138
 get_y_coord, 138
 get_zcrust_coef, 138
 get_zcrust_init, 138
 get_zcrustNameFile, 139
 get_zcrustNameFileS, 139
 huv_NF, 143
 huv_init, 143
 huv_namefile, 143
 imax_NF, 143
 imax_coef, 143
 imax_init, 143
 imax_namefile, 143
 inf, 143
 is_coord_in_file_valid, 139
 Kc_NF, 144
 Kc_coef, 143
 Kc_init, 143
 Kc_namefile, 143
 Ks_NF, 144
 Ks_coef, 144
 Ks_init, 144
 Ks_namefile, 144
 L, 144
 l, 144
 Lbound, 144
 Lbound_type, 144
 left_imp_discharge, 144
 left_imp_h, 144
 left_times_files, 145
 lim, 145
 list_point_NF, 145
 list_point_namefile, 145
 list_points_x, 145
 list_points_y, 145
 modifENO, 145
 nbtimes, 145
 Nxcell, 145
 Nycell, 145
 order, 145

- output_directory, 146
- output_format, 146
- Parameters, 123
- path_input_directory, 146
- Psi_NF, 146
- Psi_coef, 146
- Psi_init, 146
- Psi_namefile, 146
- rain, 146
- rain_NF, 146
- rain_namefile, 146
- Rbound, 146
- Rbound_type, 147
- rec, 147
- right_imp_discharge, 147
- right_imp_h, 147
- right_times_files, 147
- scheme_type, 147
- setparameters, 139
- suffix_outputs, 147
- T, 147
- Tbound, 147
- Tbound_type, 147
- top_imp_discharge, 147
- top_imp_h, 148
- top_times_files, 148
- topo, 148
- topo_NF, 148
- topography_namefile, 148
- verif_file_bc_inhomogeneous, 140
- x_coord, 148
- y_coord, 148
- zcrust_NF, 148
- zcrust_coef, 148
- zcrust_init, 148
- zcrust_namefile, 148
- Parser, 149
 - ~Parser, 149
 - getValue, 150
 - Parser, 149
- path_input_directory
 - Parameters, 146
- Psi_NF
 - Parameters, 146
- Psi_coef
 - Parameters, 146
- Psi_init
 - Parameters, 146
- Psi_namefile
 - Parameters, 146
- q1
 - Scheme, 169
- q1mod
 - Friction, 76
- q2
 - Scheme, 169
- q2mod
 - Friction, 76
- qs1
 - Scheme, 169
- qs2
 - Scheme, 169
- R_IMP_H
 - Scheme, 169
- R_IMP_Q
 - Scheme, 170
- R_choice_bound
 - Scheme, 169
- RATIO_CLOSE_CELL
 - misc.hpp, 211
- Rain, 150
 - ~Rain, 151
 - DX, 151
 - DY, 151
 - NXCELL, 151
 - NYCELL, 151
 - Rain, 151
 - rain_func, 151
 - Scheme, 170
- rain
 - Parameters, 146
- rain_NF
 - Parameters, 146
- rain_func
 - Choice_rain, 45
 - No_Rain, 105
 - Rain, 151
 - Rain_generated, 152
 - Rain_read, 154
- Rain_generated, 152
 - ~Rain_generated, 152
- rain_func, 152
 - Rain_generated, 152
- rain_namefile
 - Parameters, 146
- Rain_read, 153
 - ~Rain_read, 153
- rain_func, 154
 - Rain_read, 153
- Rbound
 - Parameters, 146
 - Scheme, 170
- Rbound_type
 - Parameters, 147
 - Scheme, 170
- rec

- Limiter, [96](#)
 - Parameters, [147](#)
- Reconstruction, [154](#)
 - ~Reconstruction, [155](#)
 - calcul, [155](#)
 - delta_z1, [155](#)
 - delta_z2, [155](#)
 - limiter, [155](#)
 - NXCELL, [155](#)
 - NYCELL, [156](#)
 - Reconstruction, [155](#)
 - z1l, [156](#)
 - z1r, [156](#)
 - z2l, [156](#)
 - z2r, [156](#)
- result
 - Choice_output, [44](#)
 - Output, [115](#)
- right_imp_discharge
 - Parameters, [147](#)
- right_imp_h
 - Parameters, [147](#)
- right_times_files
 - Parameters, [147](#)
 - Scheme, [170](#)
- row
 - Save_specific_points, [157](#)
- SCALAR
 - misc.hpp, [212](#)
- SCHEME_TYPE
 - Scheme, [170](#)
- save
 - Choice_save_specific_points, [48](#)
 - No_save, [106](#)
 - One_point, [108](#)
 - Save_specific_points, [157](#)
 - Several_points, [174](#)
- Save_specific_points, [156](#)
 - ~Save_specific_points, [157](#)
 - column, [157](#)
 - DT_SPECIFIC_POINTS, [157](#)
 - DX, [157](#)
 - DY, [157](#)
 - row, [157](#)
 - save, [157](#)
 - Save_specific_points, [157](#)
 - T_output, [158](#)
- Scheme, [158](#)
 - ~Scheme, [161](#)
 - allocation, [161](#)
 - B_IMP_H, [163](#)
 - B_IMP_Q, [163](#)
 - B_choice_bound, [163](#)
 - Bbound, [164](#)
 - Bbound_type, [164](#)
 - bottom_times_files, [164](#)
 - boundary, [161](#)
 - CFL_FIX, [164](#)
 - calcul, [162](#)
 - cpu_time, [164](#)
 - cur_time, [164](#)
 - DT_FIX, [165](#)
 - DX, [165](#)
 - DY, [165](#)
 - deallocation, [162](#)
 - delz1, [164](#)
 - delz2, [164](#)
 - delzc1, [164](#)
 - delzc2, [164](#)
 - dt1, [164](#)
 - dt_first, [165](#)
 - dt_max, [165](#)
 - dt_output, [165](#)
 - end, [165](#)
 - f1, [165](#)
 - f2, [165](#)
 - f3, [165](#)
 - FRICCOEF, [166](#)
 - Fr, [165](#)
 - froude_number, [162](#)
 - fs2d_par, [166](#)
 - g1, [166](#)
 - g2, [166](#)
 - g3, [166](#)
 - h, [166](#)
 - h1l, [166](#)
 - h1left, [166](#)
 - h1r, [166](#)
 - h1right, [166](#)
 - h2l, [166](#)
 - h2left, [167](#)
 - h2r, [167](#)
 - h2right, [167](#)
 - height_Vinf_tot, [167](#)
 - height_of_tot, [167](#)
 - hs, [167](#)
 - is_Bbound_changed, [167](#)
 - is_Lbound_changed, [167](#)
 - is_Rbound_changed, [167](#)
 - is_Tbound_changed, [167](#)
 - L_IMP_H, [168](#)
 - L_IMP_Q, [168](#)
 - L_choice_bound, [167](#)
 - Lbound, [168](#)
 - Lbound_type, [168](#)
 - left_times_files, [168](#)

- maincalcflux, 162
- maincalcscheme, 163
- n, 168
- NBTIMES, 168
- NXCELL, 168
- NYCELL, 168
- ORDER, 168
- out, 168
- out_specific_points, 169
- p_bottom_times_files, 169
- p_left_times_files, 169
- p_right_times_files, 169
- p_top_times_files, 169
- q1, 169
- q2, 169
- qs1, 169
- qs2, 169
- R_IMP_H, 169
- R_IMP_Q, 170
- R_choice_bound, 169
- Rain, 170
- Rbound, 170
- Rbound_type, 170
- right_times_files, 170
- SCHEME_TYPE, 170
- Scheme, 161
- start, 170
- T, 170
- T_IMP_H, 170
- T_IMP_Q, 170
- T_choice_bound, 170
- T_output, 171
- Tbound, 171
- Tbound_type, 171
- timecomputation, 171
- top_times_files, 171
- Total_volume_outflow, 171
- tx, 171
- ty, 171
- u, 171
- u1l, 171
- u1r, 171
- u2l, 172
- u2r, 172
- us, 172
- v, 172
- v1l, 172
- v1r, 172
- v2l, 172
- v2r, 172
- verif, 172
- Vin_tot, 172
- Vol_inf_tot_cumul, 172
- Vol_of_tot, 173
- Volrain_Tot, 173
- vs, 173
- z, 173
- scheme_type
 - Parameters, 147
- set_tx
 - Choice_flux, 34
 - Flux, 63
- setChoice
 - Choice_condition, 31
- setXY
 - Choice_condition, 32
- setparameters
 - Parameters, 139
- Several_points, 173
 - ~Several_points, 174
 - save, 174
 - Several_points, 174
- Sf1
 - Friction, 76
- Sf2
 - Friction, 76
- Sources/FullSWOF_2D.cpp, 233
- Sources/libboundaryconditions/bc_imp_discharge.↔
cpp, 234
- Sources/libboundaryconditions/bc_imp_height.cpp,
235
- Sources/libboundaryconditions/bc_neumann.cpp, 235
- Sources/libboundaryconditions/bc_periodic.cpp, 236
- Sources/libboundaryconditions/bc_wall.cpp, 237
- Sources/libboundaryconditions/boundary_condition.↔
cpp, 237
- Sources/libboundaryconditions/choice_condition.cpp,
238
- Sources/libflux/choice_flux.cpp, 238
- Sources/libflux/f_hll.cpp, 239
- Sources/libflux/f_hll2.cpp, 239
- Sources/libflux/f_hllc.cpp, 240
- Sources/libflux/f_hllc2.cpp, 240
- Sources/libflux/f_rusanov.cpp, 240
- Sources/libflux/flux.cpp, 241
- Sources/libfrictions/choice_friction.cpp, 241
- Sources/libfrictions/fr_darcy_weisbach.cpp, 242
- Sources/libfrictions/fr_laminar.cpp, 243
- Sources/libfrictions/fr_manning.cpp, 243
- Sources/libfrictions/friction.cpp, 244
- Sources/libfrictions/no_friction.cpp, 244
- Sources/libinitializations/choice_init_huv.cpp, 245
- Sources/libinitializations/choice_init_topo.cpp, 245
- Sources/libinitializations/huv_generated.cpp, 246
- Sources/libinitializations/huv_generated_radial_dam↔
_dry.cpp, 246

- Sources/libinitializations/huv_generated_radial_dam_wet.cpp, 247
- Sources/libinitializations/huv_generated_thacker.cpp, 247
- Sources/libinitializations/huv_read.cpp, 248
- Sources/libinitializations/initialization_huv.cpp, 248
- Sources/libinitializations/initialization_topo.cpp, 249
- Sources/libinitializations/topo_generated_flat.cpp, 249
- Sources/libinitializations/topo_generated_thacker.cpp, 250
- Sources/libinitializations/topo_read.cpp, 250
- Sources/liblimitations/choice_limiter.cpp, 251
- Sources/liblimitations/limiter.cpp, 251
- Sources/liblimitations/minmod.cpp, 252
- Sources/liblimitations/vanalbada.cpp, 252
- Sources/liblimitations/vanleer.cpp, 253
- Sources/libparameters/parameters.cpp, 253
- Sources/libparser/parser.cpp, 254
- Sources/librain_infiltration/choice_infiltration.cpp, 254
- Sources/librain_infiltration/choice_rain.cpp, 255
- Sources/librain_infiltration/greenampt.cpp, 255
- Sources/librain_infiltration/infiltration.cpp, 256
- Sources/librain_infiltration/no_infiltration.cpp, 256
- Sources/librain_infiltration/no_rain.cpp, 257
- Sources/librain_infiltration/rain.cpp, 257
- Sources/librain_infiltration/rain_generated.cpp, 258
- Sources/librain_infiltration/rain_read.cpp, 258
- Sources/libreconstructions/choice_reconstruction.cpp, 259
- Sources/libreconstructions/eno.cpp, 259
- Sources/libreconstructions/eno_mod.cpp, 260
- Sources/libreconstructions/hydrostatic.cpp, 260
- Sources/libreconstructions/muscl.cpp, 261
- Sources/libreconstructions/reconstruction.cpp, 261
- Sources/libsave/choice_output.cpp, 262
- Sources/libsave/choice_save_specific_points.cpp, 262
- Sources/libsave/gnuplot.cpp, 263
- Sources/libsave/no_evolution_file.cpp, 263
- Sources/libsave/no_save.cpp, 264
- Sources/libsave/one_point.cpp, 264
- Sources/libsave/output.cpp, 265
- Sources/libsave/save_specific_points.cpp, 265
- Sources/libsave/several_points.cpp, 266
- Sources/libsave/vtk_out.cpp, 266
- Sources/libschemas/choice_scheme.cpp, 267
- Sources/libschemas/order1.cpp, 267
- Sources/libschemas/order2.cpp, 268
- Sources/libschemas/scheme.cpp, 268
- start
 - Scheme, 170
- stricomp
 - asc2xyz.c, 270
 - cropxyz.c, 273
 - xyz2asc.c, 272
- suffix_outputs
 - Parameters, 147
- T
 - Parameters, 147
 - Scheme, 170
- T_IMP_H
 - Scheme, 170
- T_IMP_Q
 - Scheme, 170
- T_choice_bound
 - Scheme, 170
- T_output
 - Save_specific_points, 158
 - Scheme, 171
- TAB
 - misc.hpp, 212
- TRUE
 - asc2xyz.c, 270
 - xyz2asc.c, 271
- Tbound
 - Parameters, 147
 - Scheme, 171
- Tbound_type
 - Parameters, 147
 - Scheme, 171
- timecomputation
 - Scheme, 171
- Tools/ConvertFormat/asc2xyz.c, 269
- Tools/ConvertFormat/xyz2asc.c, 270
- Tools/ExtractWindow/cropxyz.c, 272
- top_imp_discharge
 - Parameters, 147
- top_imp_h
 - Parameters, 148
- top_times_files
 - Parameters, 148
 - Scheme, 171
- topo
 - Parameters, 148
- topo_NF
 - Parameters, 148
- Topo_generated_Thacker, 176
 - ~Topo_generated_Thacker, 177
 - initialization, 177
 - Topo_generated_Thacker, 176
- Topo_generated_flat, 175
 - ~Topo_generated_flat, 175
 - initialization, 176
 - Topo_generated_flat, 175
- Topo_read, 177
 - ~Topo_read, 178
 - initialization, 178

- Topo_read, 178
- topography_namefile
 - Parameters, 148
- Total_volume_outflow
 - Scheme, 171
- tx
 - Flux, 64
 - Scheme, 171
- ty
 - Scheme, 171
- u
 - Scheme, 171
- u1l
 - Scheme, 171
- u1r
 - Scheme, 171
- u2l
 - Scheme, 172
- u2r
 - Scheme, 172
- unormbound
 - Boundary_condition, 28
- unormfix
 - Boundary_condition, 28
- us
 - Scheme, 172
- utanbound
 - Boundary_condition, 28
- v
 - Scheme, 172
- v1l
 - Scheme, 172
- v1r
 - Scheme, 172
- v2l
 - Scheme, 172
- v2r
 - Scheme, 172
- VE_CA
 - misc.hpp, 212
- VECT
 - misc.hpp, 212
- VERSION
 - misc.hpp, 212
- VanAlbada, 178
 - ~VanAlbada, 179
 - calcul, 179
 - VanAlbada, 179
- VanLeer, 180
 - ~VanLeer, 180
 - calcul, 180
 - VanLeer, 180
- verif
 - Scheme, 172
- verif_file_bc_inhomogeneous
 - Parameters, 140
- Vin
 - Infiltration, 91
- Vin_tot
 - Scheme, 172
- Vol_inf_tot_cumul
 - Scheme, 172
- Vol_of_tot
 - Scheme, 173
- Volrain_Tot
 - Scheme, 173
- vs
 - Scheme, 173
- Vtk_Out, 181
 - ~Vtk_Out, 182
 - Vtk_Out, 181
 - write, 182
- write
 - Choice_output, 44
 - Gnuplot, 77
 - No_Evolution_File, 101
 - Output, 115
 - Vtk_Out, 182
- x_coord
 - Parameters, 148
- xyz2asc.c
 - ASCHEADER, 271
 - ASCHEADER_NBCHAR, 271
 - ASCHEADER_NBLINE, 271
 - FALSE, 271
 - headerformat, 272
 - LENGTH_OF_LINE, 271
 - main, 272
 - OUTDATA_TYPE, 272
 - stricmp, 272
 - TRUE, 271
- y_coord
 - Parameters, 148
- z
 - Scheme, 173
- z1l
 - Reconstruction, 156
- z1r
 - Reconstruction, 156
- z2l
 - Reconstruction, 156
- z2r
 - Reconstruction, 156

ZERO

- misc.hpp, [212](#)

zcrust_NF

- Parameters, [148](#)

zcrust_coef

- Parameters, [148](#)

zcrust_init

- Parameters, [148](#)

zcrust_namefile

- Parameters, [148](#)