Documentation of FullSWOF_1D

v1.00.02

(2013-05-12)

Generated by Doxygen 1.8.2
on Sun May 12 2013 00:36:57

Chapter 1

Todo List

Class Boundary_condition

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

Member Choice_condition::calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)

Add time dependance in the boundary conditions.

Member Choice_friction::set_c (SCALAR)

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

Member ENO::ENO (Parameters &, VECT)

Exception should be treated.

Member ENO_mod::ENO_mod (Parameters &, VECT)

Exception should be treated.

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

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

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

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

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

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

Member Friction::set_c (SCALAR)

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

Member Gnuplot::Gnuplot (Parameters &)

Exception should be treated.

Member Hu_read::initialization (VECT &, VECT &)

Exception should be treated.

Member Initialization_topo::Initialization_topo (Parameters &)

Exception should be treated.

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

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

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

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

Member Order2::Order2 (Parameters &)

Exception should be treated.

Member Output::initial (VECT, VECT, VECT)

Exception should be treated.

Member Output::Output (Parameters &)

Exception should be treated.

Member Output::result (SCALAR, clock_t, SCALAR)

Exception should be treated.

Member Output::save (VECT, VECT, VECT)

Exception should be treated.

Member Parameters::setparameters (const char *)

Exception should be treated.

Member Parser::GetValue (const char *)

Exception should be treated.

Member Parser::Parser (const char *)

Exception should be treated.

Member Reconstruction::Reconstruction (Parameters &, VECT)

Exception should be treated.

Member Scheme::allocation ()

Exception should be treated.

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

Improve the treatment of numerical errors.

Member Topo_read::initialization (VECT &)

Exception should be treated.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

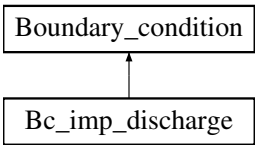
Class Documentation

5.1 Bc_imp_discharge Class Reference

Imposed discharge.

`#include <bc_imp_discharge.hpp>`

Inheritance diagram for Bc_imp_discharge:



Public Member Functions

- Bc_imp_discharge (Parameters &, VECT &, int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)
  Calculates the boundary condition.
- SCALAR NewtonSolver (const SCALAR, const SCALAR, const SCALAR, const SCALAR, int) const
  Solves the equation with Newton iterative method.
- SCALAR ValPoly (const SCALAR, const SCALAR, const SCALAR, const SCALAR, int) const
  Gives the value of the function that must vanish.
- SCALAR ValDerPoly (const SCALAR, const SCALAR, const SCALAR, int) const
  Gives the value of the derivative of the function that must vanish.
- virtual ~Bc_imp_discharge ()
  Destructor.

Additional Inherited Members

5.1.1 Detailed Description

Imposed discharge.

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

Definition at line 72 of file bc_imp_discharge.hpp.

5.1.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 60 of file bc_imp_discharge.cpp.

5.1.2.2 Bc_imp_discharge::~Bc_imp_discharge ( ) [virtual]

Destructor.

Definition at line 203 of file bc_imp_discharge.cpp.

5.1.3 Member Function Documentation

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

Calculates the boundary condition.

Two cases are considered: subcritical and supercritical flows.

Parameters

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

Warning

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

Modifies

boundary_condition::hbound water height on the fictive cell.
boundary_condition::ubound velocity on the fictive cell.

Implements Boundary_condition.

Definition at line 86 of file bc_imp_discharge.cpp.

5.1.3.2 SCALAR Bc_imp_discharge::NewtonSolver (const SCALAR *HIN,* const SCALAR *UIN,* const SCALAR *QFIX,* const SCALAR *H_INIT,* int *normal*) const

Solves the equation with Newton iterative method.

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

Parameters

in	*HIN*	water height of the first cell inside the domain.
in	*UIN*	velocity of the first cell inside the domain.
in	*QFIX*	fixed (imposed) value of the discharge.
in	*H_INIT*	initialisation of the Newton solver.
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary.

Warning

Warning: Newton bc did not converge.

Returns

h: water height that satifies Riemann invariants.

Definition at line 135 of file bc_imp_discharge.cpp.

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

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

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

Parameters

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

Returns

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

Definition at line 187 of file bc_imp_discharge.cpp.

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

Gives the value of the function that must vanish.

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

Parameters

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

Returns

The value of the polynomial function.

Definition at line 171 of file bc_imp_discharge.cpp.

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

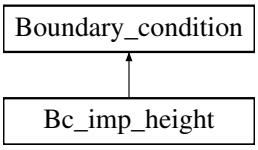
- Headers/libboundaryconditions/bc_imp_discharge.hpp
- Sources/libboundaryconditions/bc_imp_discharge.cpp

5.2 Bc_imp_height Class Reference

Imposed water height.

`#include <bc_imp_height.hpp>`

Inheritance diagram for Bc_imp_height:



Public Member Functions

- Bc_imp_height (Parameters &, VECT &, int)

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)

  Calculates the boundary condition.
- virtual ~Bc_imp_height ()

  Destructor.

Additional Inherited Members

5.2.1 Detailed Description

Imposed water height.

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

Definition at line 76 of file bc_imp_height.hpp.

5.2.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 63 of file bc_imp_height.cpp.

5.2.2.2 Bc_imp_height::~Bc_imp_height ( ) [virtual]

Destructor.

Definition at line 135 of file bc_imp_height.cpp.

5.2.3 Member Function Documentation

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

Calculates the boundary condition.

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

Parameters

in	*hin*	water height of the first cell inside the domain.
in	*uin*	velocity of the first cell inside the domain.
in	*hfix*	fixed (imposed) value of the water height.
in	*qfix*	fixed (imposed) value of the discharge.
in	*hin_oppbound*	value of the water height of the first cell inside the domain at the opposite bound (unused).
in	*uin_oppbound*	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	*time*	current time (unused).
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary.

Modifies

boundary_condition::hbound water height on the fictive cell.
boundary_condition::ubound velocity on the fictive cell.

Implements Boundary_condition.

Definition at line 85 of file bc_imp_height.cpp.

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

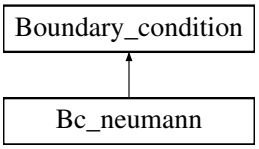
- Headers/libboundaryconditions/bc_imp_height.hpp
- Sources/libboundaryconditions/bc_imp_height.cpp

5.3 Bc_neumann Class Reference

Neumann condition.

`#include <bc_neumann.hpp>`

Inheritance diagram for Bc_neumann:



Public Member Functions

- Bc_neumann (Parameters &, VECT &, int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)
  Calculates the boundary condition.
- virtual ~Bc_neumann ()
  Destructor.

Additional Inherited Members

5.3.1 Detailed Description

Neumann condition.

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

Definition at line 71 of file bc_neumann.hpp.

5.3.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 60 of file bc_neumann.cpp.

5.3.2.2 Bc_neumann::~Bc_neumann ( ) [virtual]

Destructor.

Definition at line 104 of file bc_neumann.cpp.

5.3.3 Member Function Documentation

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

Calculates the boundary condition.

Two cases are considered: subcritical and supercritical flows.

Parameters

in	*hin*	water height of the first cell inside the domain.
in	*uin*	velocity of the first cell inside the domain.
in	*hfix*	fixed (imposed) value of the water height (unused).
in	*qfix*	fixed (imposed) value of the discharge (unused).
in	*hin_oppbound*	value of the water height of the first cell inside the domain at the opposite bound (unused).
in	*uin_oppbound*	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	*time*	current time (unused).
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary (unused).

Modifies

boundary_condition::hbound water height on the fictive cell.
boundary_condition::ubound velocity on the fictive cell.

Implements Boundary_condition.

Definition at line 82 of file bc_neumann.cpp.

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

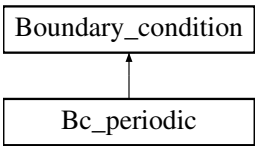
- Headers/libboundaryconditions/bc_neumann.hpp
- Sources/libboundaryconditions/bc_neumann.cpp

5.4 Bc_periodic Class Reference

Periodic condition.

`#include <bc_periodic.hpp>`

Inheritance diagram for Bc_periodic:



Public Member Functions

- Bc_periodic (Parameters &, VECT &, int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)

Calculates boundary condition.

- virtual ∼Bc_periodic ()

  Destructor.

Additional Inherited Members

5.4.1 Detailed Description

Periodic condition.

Class that computes the periodic boundary condition

Definition at line 71 of file bc_periodic.hpp.

5.4.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 59 of file bc_periodic.cpp.

5.4.2.2 Bc_periodic::∼Bc_periodic ( ) [virtual]

Destructor.

Definition at line 104 of file bc_periodic.cpp.

5.4.3 Member Function Documentation

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

Calculates boundary condition.

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

Parameters

in	*hin*	water height of the first cell inside the domain (unused).
in	*uin*	velocity of the first cell inside the domain (unused).
in	*hfix*	fixed (imposed) value of the water height (unused).
in	*qfix*	fixed (imposed) value of the discharge (unused).
in	*hin_oppbound*	value of the water height of the first cell inside the domain at the opposite bound.
in	*uin_oppbound*	value of the velocity of the first cell inside the domain at the opposite bound.
in	*time*	current time (unused).
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary (unused).

Modifies

boundary_condition::hbound water height on the fictive cell.
boundary_condition::ubound velocity on the fictive cell.

Implements Boundary_condition.

Definition at line 82 of file bc_periodic.cpp.

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

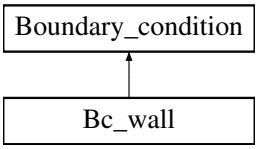
- Headers/libboundaryconditions/bc_periodic.hpp
- Sources/libboundaryconditions/bc_periodic.cpp

5.5 Bc_wall Class Reference

Wall condition.

`#include <bc_wall.hpp>`

Inheritance diagram for Bc_wall:



Public Member Functions

- Bc_wall (Parameters &, VECT &, int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)
  Calculates the boundary condition.
- virtual ~Bc_wall ()
  Destructor.

Additional Inherited Members

5.5.1 Detailed Description

Wall condition.

Class that computes the wall boundary condition.

Definition at line 71 of file bc_wall.hpp.

5.5.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

Definition at line 59 of file bc_wall.cpp.

5.5.2.2 Bc_wall::~Bc_wall ( ) [virtual]

Destructor.

Definition at line 103 of file bc_wall.cpp.

5.5.3 Member Function Documentation

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

Calculates the boundary condition.

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

Parameters

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

Modifies

boundary_condition::hbound water height on the fictive cell.
boundary_condition::ubound velocity on the fictive cell.

Implements Boundary_condition.

Definition at line 80 of file bc_wall.cpp.

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

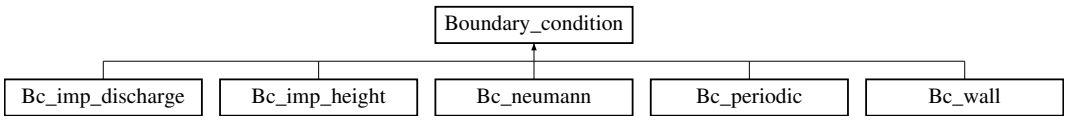
- Headers/libboundaryconditions/bc_wall.hpp
- Sources/libboundaryconditions/bc_wall.cpp

5.6 Boundary_condition Class Reference

Boundary condition.

`#include <boundary_condition.hpp>`

Inheritance diagram for Boundary_condition:



Public Member Functions

- Boundary_condition (Parameters &)

  Constructor.
- virtual void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)=0

  Function to be specified in each boundary condition.
- SCALAR get_hbound () const

  Gives the water height on the fictive cell.
- SCALAR get_ubound () const

  Gives the velocity of the flow on the fictive cell.
- virtual ∼Boundary_condition ()

  Destructor.

Protected Attributes

- SCALAR hbound
- SCALAR ubound
- SCALAR ufix
- const int NXCELL

5.6.1 Detailed Description

Boundary condition.

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

Todo 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 71 of file boundary_condition.hpp.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Boundary_condition::Boundary_condition (Parameters & *par*)

Constructor.

Defines the number of cells.

Parameters

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

Definition at line 58 of file boundary_condition.cpp.

5.6.2.2 Boundary_condition::~Boundary_condition ( ) [virtual]

Destructor.

Definition at line 87 of file boundary_condition.cpp.

5.6.3 Member Function Documentation

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

Function to be specified in each boundary condition.

Implemented in Bc_imp_height, Bc_imp_discharge, Bc_neumann, Bc_periodic, and Bc_wall.

5.6.3.2 SCALAR Boundary_condition::get_hbound ( ) const

Gives the water height on the fictive cell.

Returns

Boundary_condition::hbound water height on the fictive cell.

Definition at line 67 of file boundary_condition.cpp.

5.6.3.3 SCALAR Boundary_condition::get_ubound ( ) const

Gives the velocity of the flow on the fictive cell.

Returns

Boundary_condition::ubound velocity on the fictive cell.

Definition at line 77 of file boundary_condition.cpp.

5.6.4 Member Data Documentation

5.6.4.1 SCALAR Boundary_condition::hbound [protected]

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

Definition at line 93 of file boundary_condition.hpp.

5.6.4.2 const int Boundary_condition::NXCELL [protected]

Number of cells (in space).

Definition at line 99 of file boundary_condition.hpp.

5.6.4.3 SCALAR Boundary_condition::ubound [protected]

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

Definition at line 95 of file boundary_condition.hpp.

5.6.4.4 SCALAR Boundary_condition::ufix [protected]

Imposed value of the velocity from Parameters::qfix and Parameters::hfix.

Definition at line 97 of file boundary_condition.hpp.

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

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

5.7 Choice_condition Class Reference

Choice of boundary condition.

`#include <choice_condition.hpp>`

Public Member Functions

- Choice_condition (int, Parameters &, VECT &, int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)
  Calculates the boundary condition.
- SCALAR get_hbound ()
  Gives the water height on the fictive cell.
- SCALAR get_ubound ()
  Gives the velocity of the flow on the fictive cell.
- virtual ~Choice_condition ()
  Destructor.

5.7.1 Detailed Description

Choice of boundary condition.

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

Definition at line 90 of file choice_condition.hpp.

5.7.2 Constructor & Destructor Documentation

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

Constructor.

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

Parameters

in	*choice*	integer that correspond to the chosen boundary condition.
in	*par*	parameter, contains all the values from the parameters file.
in	*z*	vector that represents the topography.
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary.

Definition at line 59 of file choice_condition.cpp.

5.7.2.2 Choice_condition::~Choice_condition ( ) [virtual]

Destructor.

Definition at line 130 of file choice_condition.cpp.

5.7.3 Member Function Documentation

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

Calculates the boundary condition.

Calls the calculation of the boundary condition.

Parameters

in	*hin*	water height of the first cell inside the domain.
in	*uin*	velocity of the first cell inside the domain.
in	*hfix*	fixed (imposed) value of the water height.
in	*qfix*	fixed (imposed) value of the discharge.
in	*hin_oppbound*	value of the water height of the first cell inside the domain at the opposite bound.
in	*uin_oppbound*	value of the velocity of the first cell inside the domain at the opposite bound.
in	*time*	current time (unused, for the moment).
in	*normal*	integer to specify whether it is the left (-1) or the right (1) boundary.

Todo Add time dependance in the boundary conditions.

Definition at line 89 of file choice_condition.cpp.

5.7.3.2 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 108 of file choice_condition.cpp.

5.7.3.3 SCALAR Choice_condition::get_ubound ( )

Gives the velocity of the flow on the fictive cell.

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

Returns

Boundary_condition::ubound velocity on the fictive cell for the chosen boundary condition.

Definition at line 119 of file choice_condition.cpp.

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

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

5.8 Choice_flux Class Reference

Choice of numerical flux.

`#include <choice_flux.hpp>`

Public Member Functions

- Choice_flux (int)

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR)

  Calculates the numerical flux.
- void set_tx (SCALAR)

  Sets the variable Flux::tx.
- SCALAR get_f1 ()

  Gives the first component of the numerical flux.
- SCALAR get_f2 ()

  Gives the second component of the numerical flux.
- SCALAR get_cfl ()

  Gives the CFL value.
- virtual ~Choice_flux ()

  Destructor.

5.8.1 Detailed Description

Choice of numerical flux.

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

Definition at line 92 of file choice_flux.hpp.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Choice_flux::Choice_flux (int *choice*)

Constructor.

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

Parameters

in	*choice*	integer that correspond to the chosen numerical flux.

Definition at line 59 of file choice_flux.cpp.

5.8.2.2 Choice_flux::~Choice_flux ( ) [virtual]

Destructor.

Definition at line 144 of file choice_flux.cpp.

5.8.3 Member Function Documentation

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

Calculates the numerical flux.

Calls the calculation of the numerical flux.

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Definition at line 86 of file choice_flux.cpp.

5.8.3.2 SCALAR Choice_flux::get_cfl ( )

Gives the CFL value.

Calls the function to get the value of the CFL.

Returns

Flux::cfl value of the CFL.

Definition at line 133 of file choice_flux.cpp.

5.8.3.3 SCALAR Choice_flux::get_f1 ( )

Gives the first component of the numerical flux.

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

Returns

Flux::f1 first componant of the numerical flux.

Definition at line 111 of file choice_flux.cpp.

5.8.3.4 SCALAR Choice_flux::get_f2 ( )

Gives the second component of the numerical flux.

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

Returns

Flux::f2 second componant of the numerical flux.

Definition at line 122 of file choice_flux.cpp.

5.8.3.5 void Choice_flux::set_tx (SCALAR *tx*)

Sets the variable Flux::tx.

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

Parameters

in	*tx*	value of dt/dx.

Definition at line 100 of file choice_flux.cpp.

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

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

5.9 Choice_friction Class Reference

Choice of friction law.

`#include <choice_friction.hpp>`

Public Member Functions

- Choice_friction (int)
  Constructor.
- void calc (SCALAR, SCALAR, SCALAR)
  Calculates the friction term.
- SCALAR get_qmod ()
  Gives the discharge modified by the friction term.
- void set_c (SCALAR)
  Sets the variable Friction::c.
- void set_dt (SCALAR)

Sets the variable Friction::dt.

- virtual ∼Choice_friction ()

  Destructor.

5.9.1 Detailed Description

Choice of friction law.

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

Definition at line 82 of file choice_friction.hpp.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Choice_friction::Choice_friction (int *choice*)

Constructor.

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

Parameters

in	*choice*	integer that correspond to the chosen friction law.

Definition at line 59 of file choice_friction.cpp.

5.9.2.2 Choice_friction::∼Choice_friction ( ) [virtual]

Destructor.

Definition at line 131 of file choice_friction.cpp.

5.9.3 Member Function Documentation

5.9.3.1 void Choice_friction::calc (SCALAR *uold,* SCALAR *hnew,* SCALAR *qnew*)

Calculates the friction term.

Calls the calculation of the friction law.

Parameters

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

Note

The friction only affects the discharge.

Definition at line 83 of file choice_friction.cpp.

5.9.3.2 SCALAR Choice_friction::get_qmod ( )

Gives the discharge modified by the friction term.

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

Returns

Friction::qmod discharge modified by the friction term.

Definition at line 97 of file choice_friction.cpp.

5.9.3.3 void Choice_friction::set_c (SCALAR *c*)

Sets the variable Friction::c.

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

Parameters

in	*c*	value of the friction coefficient.

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

Definition at line 108 of file choice_friction.cpp.

5.9.3.4 void Choice_friction::set_dt (SCALAR *dt*)

Sets the variable Friction::dt.

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

Parameters

in	*dt*	value of the time step.

Definition at line 120 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_init_hu Class Reference

Choice of initialization for h and u.

`#include <choice_init_hu.hpp>`

Public Member Functions

- Choice_init_hu (Parameters &)

Constructor.

- void initialization (VECT &, VECT &)

  Performs the initialization.

- virtual ~Choice_init_hu ()

  Destructor.

5.10.1 Detailed Description

Choice of initialization for h and u.

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

Definition at line 96 of file choice_init_hu.hpp.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Choice_init_hu::Choice_init_hu (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 59 of file choice_init_hu.cpp.

5.10.2.2 Choice_init_hu::~Choice_init_hu ( ) [virtual]

Destructor.

Definition at line 103 of file choice_init_hu.cpp.

5.10.3 Member Function Documentation

5.10.3.1 void Choice_init_hu::initialization (VECT & *h*, VECT & *u*)

Performs the initialization.

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

Parameters

in	*h*	water height.
in	*u*	velocity.

Definition at line 91 of file choice_init_hu.cpp.

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

- Headers/libinitializations/choice_init_hu.hpp
- Sources/libinitializations/choice_init_hu.cpp

5.11 Choice_init_topo Class Reference

Choice of initialization for the topography.

`#include <choice_init_topo.hpp>`

Public Member Functions

- Choice_init_topo (Parameters &)
  Constructor.
- void initialization (VECT &)
  Performs the initialization.
- virtual ~Choice_init_topo ()
  Destructor.

5.11.1 Detailed Description

Choice of initialization for the topography.

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

Definition at line 87 of file choice_init_topo.hpp.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Choice_init_topo::Choice_init_topo (Parameters & *par*)

Constructor.

Defines the initialization of the topography 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_init_topo.cpp.

5.11.2.2 Choice_init_topo::~Choice_init_topo ( ) [virtual]

Destructor.

Definition at line 97 of file choice_init_topo.cpp.

5.11.3 Member Function Documentation

5.11.3.1 void Choice_init_topo::initialization (VECT & *z*)

Performs the initialization.

Calls the initialization of the topography.

Parameters

in	*z*	topography.

Definition at line 86 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

5.12 Choice_limiter Class Reference

Choice of slope limiter.

`#include <choice_limiter.hpp>`

Public Member Functions

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

5.12.1 Detailed Description

Choice of slope limiter.

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

Definition at line 83 of file choice_limiter.hpp.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Choice_limiter::Choice_limiter (int *choice*)

Constructor.

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

Parameters

in	*choice*	integer that correspond to the chosen slope limiter.

Definition at line 59 of file choice_limiter.cpp.

5.12.2.2 Choice_limiter::~Choice_limiter ( ) [virtual]

Destructor.

Definition at line 103 of file choice_limiter.cpp.

5.12.3 Member Function Documentation

5.12.3.1 void Choice_limiter::calc (SCALAR *a*, SCALAR *b*)

Calculates the slope limiter.

Calls the calculation of the slope limiter.

Parameters

in	*a*	slope on the left of the cell.
in	*b*	slope on the right of the cell.

Definition at line 80 of file choice_limiter.cpp.

5.12.3.2 SCALAR Choice_limiter::get_rec ( ) const

Gives the reconstructed value.

Calls the function to get the reconstructed value.

Returns

Limiter::rec reconstructed value for the chosen slope limiter.

Definition at line 92 of file choice_limiter.cpp.

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

- Headers/liblimitations/choice_limiter.hpp
- Sources/liblimitations/choice_limiter.cpp

5.13 Choice_output Class Reference

Choice of output format.

`#include <choice_output.hpp>`

Public Member Functions

- Choice_output (Parameters &)
  Constructor.
- void write (VECT, VECT, VECT, int)
  Save the current time.
- void initial (VECT, VECT, VECT)
  Saves the initial time.

- void bound_flux (int, SCALAR, SCALAR, SCALAR, SCALAR)

  Saves the flux and the cumulative flux on the boundaries.
- void result (SCALAR, clock_t, SCALAR)

  Saves global values.
- void save (VECT, VECT, VECT)

  Saves the final time.
- virtual ~Choice_output ()

  Destructor.

5.13.1 Detailed Description

Choice of output format.

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

Definition at line 74 of file choice_output.hpp.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Choice_output::Choice_output (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 58 of file choice_output.cpp.

5.13.2.2 Choice_output::~Choice_output ( ) [virtual]

Destructor.

Definition at line 141 of file choice_output.cpp.

5.13.3 Member Function Documentation

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

Saves the flux and the cumulative flux on the boundaries.

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

Parameters

in	*tps*	current time.
in	*f_l*	flux on the left boundary.
in	*f_r*	flux on the right boundary.
in	*cum_f_l*	cumulative flux on the left boundary.
in	*cum_f_r*	cumulative flux on the right boundary.

Definition at line 100 of file choice_output.cpp.

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

Saves the initial time.

Calls the saving of the inital time.

Parameters

in	*h*	water height.
in	*u*	velocity.
in	*z*	topography.

Definition at line 87 of file choice_output.cpp.

5.13.3.3 void Choice_output::result (SCALAR *t*, clock_t *cpu*, SCALAR *froude*)

Saves global values.

Calls the saving of the global values.

Parameters

in	*t*	elapsed time.
in	*cpu*	CPU time.
in	*froude*	mean Froude number (in space) at the final time.

Definition at line 115 of file choice_output.cpp.

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

Saves the final time.

Calls the saving of the final time.

Parameters

in	*z*	topography.
in	*h*	water height.
in	*q*	discharge.

Definition at line 128 of file choice_output.cpp.

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

Save the current time.

Calls the saving of the current time.

Parameters

in	*h*	water height.

in	*q*	discharge.
in	*z*	topography.
in	*n*	index of the current time.

Definition at line 73 of file choice_output.cpp.

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

- Headers/libsave/choice_output.hpp
- Sources/libsave/choice_output.cpp

5.14 Choice_reconstruction Class Reference

Choice of reconstruction.

`#include <choice_reconstruction.hpp>`

Public Member Functions

- Choice_reconstruction (Parameters &, VECT)

  Constructor.
- void calc (VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &)
- virtual ~Choice_reconstruction ()

  Destructor.

5.14.1 Detailed Description

Choice of reconstruction.

Class that calls the reconstruction chosen in the parameters file.

Definition at line 82 of file choice_reconstruction.hpp.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Choice_reconstruction::Choice_reconstruction (Parameters & *par,* VECT *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*	vector that represents the topography.

Definition at line 59 of file choice_reconstruction.cpp.

5.14.2.2 Choice_reconstruction::~Choice_reconstruction ( ) [virtual]

Destructor.

Definition at line 100 of file choice_reconstruction.cpp.

5.14.3 Member Function Documentation

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

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

Definition at line 81 of file choice_reconstruction.cpp.

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

- Headers/libreconstructions/choice_reconstruction.hpp
- Sources/libreconstructions/choice_reconstruction.cpp

5.15 Choice_scheme Class Reference

Choice of numerical scheme.

`#include <choice_scheme.hpp>`

Public Member Functions

- Choice_scheme (Parameters &)
  Constructor.
- void calc ()
  Performs the scheme.
- virtual ~Choice_scheme ()
  Destructor.

5.15.1 Detailed Description

Choice of numerical scheme.

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

Definition at line 78 of file choice_scheme.hpp.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 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 59 of file choice_scheme.cpp.

5.15.2.2 Choice_scheme::~Choice_scheme ( ) [virtual]

Destructor.

Definition at line 88 of file choice_scheme.cpp.

5.15.3 Member Function Documentation

5.15.3.1 void Choice_scheme::calc ( )

Performs the scheme.

Calls the computation of the solution.

Definition at line 77 of file choice_scheme.cpp.

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

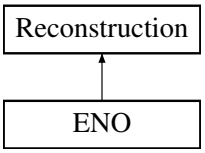
- Headers/libschemes/choice_scheme.hpp
- Sources/libschemes/choice_scheme.cpp

5.16 ENO Class Reference

ENO recontruction

`#include <eno.hpp>`

Inheritance diagram for ENO:



Public Member Functions

- ENO (Parameters &, VECT)
  Constructor.
- void calc (VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &)

Calculates the reconstruction in space.

- ∼ENO ()

  Destructor.

Additional Inherited Members

5.16.1 Detailed Description

ENO recontruction

Class that computes ENO reconstruction in space.

Definition at line 69 of file eno.hpp.

5.16.2 Constructor & Destructor Documentation

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

Constructor.

Initializations.

Parameters

in	*par*	parameter, contains all the values from the parameters file.
in	*z*	topography.

Warning

Problem: allocation of som_z failed.

Note

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

Todo Exception should be treated.

Definition at line 59 of file eno.cpp.

5.16.2.2 ENO::∼ENO ( )

Destructor.

Definition at line 199 of file eno.cpp.

5.16.3 Member Function Documentation

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

Calculates the reconstruction in space.

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

Parameters

in	*h*	water height.
in	*u*	velocity of the flow.
in	*z*	topography.
out	*hr*	reconstructed water height on the right of the cell.
out	*hl*	reconstructed water height on the left of the cell.
out	*delz*	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	*ur*	reconstructed velocity on the right of the cell.
out	*ul*	reconstructed velocity on the left of the cell.
out	*dzi*	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Implements Reconstruction.

Definition at line 83 of file eno.cpp.

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

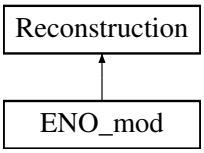
- Headers/libreconstructions/eno.hpp
- Sources/libreconstructions/eno.cpp

5.17 ENO_mod Class Reference

Modified ENO recontruction.

`#include <eno_mod.hpp>`

Inheritance diagram for ENO_mod:



Public Member Functions

- ENO_mod (Parameters &, VECT)
  Constructor.
- void calc (VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &)
  Calculates the reconstruction in space.
- ∼ENO_mod ()
  Destructor.

Additional Inherited Members

5.17.1 Detailed Description

Modified ENO recontruction.

Class that computes the modified ENO reconstruction in space.

Definition at line 69 of file eno_mod.hpp.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 ENO_mod::ENO_mod (Parameters & *par,* VECT *z*)

Constructor.

Initializations.

Parameters

in	*par*	parameter, contains all the values from the parameters file.
in	*z*	topography.

Warning

Problem: allocation of som_z failed.

Note

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

Todo Exception should be treated.

Definition at line 58 of file eno_mod.cpp.

5.17.2.2 ENO_mod::~ENO_mod ( )

Destructor.

Definition at line 214 of file eno_mod.cpp.

5.17.3 Member Function Documentation

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

Calculates the reconstruction in space.

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

Parameters

in	*h*	water height.
in	*u*	velocity of the flow.
in	*z*	topography.
out	*hr*	reconstructed water height on the right of the cell.
out	*hl*	reconstructed water height on the left of the cell.
out	*delz*	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	*ur*	reconstructed velocity on the right of the cell.
out	*ul*	reconstructed velocity on the left of the cell.
out	*dzi*	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Implements Reconstruction.

Definition at line 82 of file eno_mod.cpp.

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

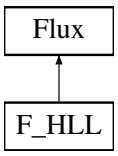
- Headers/libreconstructions/eno_mod.hpp
- Sources/libreconstructions/eno_mod.cpp

5.18 F_HLL Class Reference

HLL flux.

`#include <f_hll.hpp>`

Inheritance diagram for F_HLL:



Public Member Functions

- F_HLL ()

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR)

  Calculates the numerical flux.
- virtual ~F_HLL ()

  Destructor.

Additional Inherited Members

5.18.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux.

Definition at line 69 of file f_hll.hpp.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 F_HLL::F_HLL ( )

Constructor.

Definition at line 58 of file f_hll.cpp.

5.18.2.2 F_HLL::~F_HLL ( ) [virtual]

Destructor.

Definition at line 120 of file f_hll.cpp.

5.18.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

with

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

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

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Modifies

flux::f1 first componant of the numerical flux.
flux::f2 second componant of the numerical flux.
flux::cfl value of the CFL.

Note

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

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

Implements Flux.

Definition at line 61 of file f_hll.cpp.

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

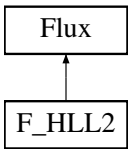
- Headers/libflux/f_hll.hpp
- Sources/libflux/f_hll.cpp

5.19 F_HLL2 Class Reference

HLL flux.

`#include <f_hll2.hpp>`

Inheritance diagram for F_HLL2:



Public Member Functions

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

Additional Inherited Members

5.19.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux with a reduced formulation.

Definition at line 70 of file f_hll2.hpp.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 F_HLL2::F_HLL2 ( )

Constructor.

Definition at line 58 of file f_hll2.cpp.

5.19.2.2 F_HLL2::~F_HLL2 ( ) [virtual]

Destructor.

Definition at line 106 of file f_hll2.cpp.

5.19.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

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

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Modifies

flux::f1 first componant of the numerical flux.
flux::f2 second componant of the numerical flux.
flux::cfl value of the CFL.

Implements Flux.

Definition at line 61 of file f_hll2.cpp.

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

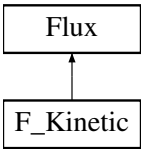
- Headers/libflux/f_hll2.hpp
- Sources/libflux/f_hll2.cpp

5.20 F_Kinetic Class Reference

Kinetic flux.

`#include <f_kinetic.hpp>`

Inheritance diagram for F_Kinetic:



Public Member Functions

- F_Kinetic ()

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR, SCALAR)

Calculates the numerical flux.

- virtual ∼F_Kinetic ()

  Destructor.

Additional Inherited Members

5.20.1 Detailed Description

Kinetic flux.

Class that computes the kinetic numerical flux.

Definition at line 70 of file f_kinetic.hpp.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 F_Kinetic::F_Kinetic ( )

Constructor.

Definition at line 58 of file f_kinetic.cpp.

5.20.2.2 F_Kinetic::∼F_Kinetic ( ) [virtual]

Destructor.

Definition at line 123 of file f_kinetic.cpp.

5.20.3 Member Function Documentation

5.20.3.1 void F_Kinetic::calc (SCALAR *h_L*, SCALAR *u_L*, SCALAR *h_R*, SCALAR *u_R*) [virtual]

Calculates the numerical flux.

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

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

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

where

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

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

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Modifies

flux::f1 first componant of the numerical flux.
flux::f2 second componant of the numerical flux.
flux::cfl value of the CFL.

Implements Flux.

Definition at line 64 of file f_kinetic.cpp.

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

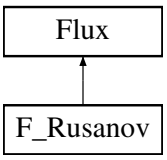
- Headers/libflux/f_kinetic.hpp
- Sources/libflux/f_kinetic.cpp

5.21 F_Rusanov Class Reference

Rusanov flux.

`#include <f_rusanov.hpp>`

Inheritance diagram for F_Rusanov:



Public Member Functions

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

Additional Inherited Members

5.21.1 Detailed Description

Rusanov flux.

Class that computes Rusanov numerical flux.

Definition at line 69 of file f_rusanov.hpp.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 F_Rusanov::F_Rusanov ( )

Constructor.

Definition at line 58 of file f_rusanov.cpp.

5.21.2.2 F_Rusanov::~F_Rusanov ( ) [virtual]

Destructor.

Definition at line 97 of file f_rusanov.cpp.

5.21.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

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

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Modifies

flux::f1 first componant of the numerical flux.
flux::f2 second componant of the numerical flux.
flux::cfl value of the CFL.

Implements Flux.

Definition at line 61 of file f_rusanov.cpp.

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

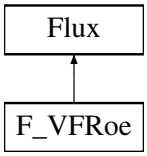
- Headers/libflux/f_rusanov.hpp
- Sources/libflux/f_rusanov.cpp

5.22 F_VFRoe Class Reference

VFRoe flux.

`#include <f_vfroe.hpp>`

Inheritance diagram for F_VFRoe:



Public Member Functions

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

Additional Inherited Members

5.22.1 Detailed Description

VFRoe flux.

Class that computes the VFRoe-ncv numerical flux.

Definition at line 70 of file f_vfroe.hpp.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 F_VFRoe::F_VFRoe ( )

Constructor.

Definition at line 58 of file f_vfroe.cpp.

5.22.2.2 F_VFRoe::~F_VFRoe ( ) [virtual]

Destructor.

Definition at line 149 of file f_vfroe.cpp.

5.22.3 Member Function Documentation

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

Calculates the numerical flux.

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

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

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

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

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

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

Parameters

in	*h_L*	water height at the left of the interface where the flux is calculated.
in	*u_L*	velocity at the left of the interface where the flux is calculated.
in	*h_R*	water height at the right of the interface where the flux is calculated.
in	*u_R*	velocity at the right of the interface where the flux is calculated.

Modifies

flux::f1 first componant of the numerical flux.
flux::f2 second componant of the numerical flux.
flux::cfl value of the CFL.

Implements Flux.

Definition at line 74 of file f_vfroe.cpp.

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

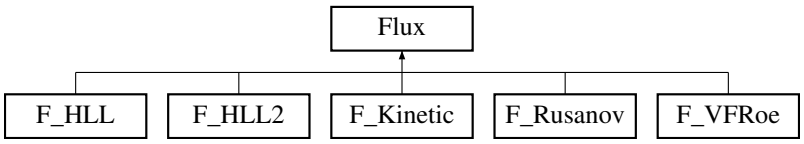
- Headers/libflux/f_vfroe.hpp
- Sources/libflux/f_vfroe.cpp

5.23 Flux Class Reference

Numerical flux.

`#include <flux.hpp>`

Inheritance diagram for Flux:



Public Member Functions

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

- virtual ~Flux ()

  Destructor.

Protected Attributes

- SCALAR f1
- SCALAR f2
- SCALAR cfl
- SCALAR tx

5.23.1 Detailed Description

Numerical flux.

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

Definition at line 71 of file flux.hpp.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 Flux::Flux ( )

Constructor.

Definition at line 58 of file flux.cpp.

5.23.2.2 Flux::~Flux ( ) [virtual]

Destructor.

Definition at line 106 of file flux.cpp.

5.23.3 Member Function Documentation

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

Function to be specified in each numerical flux.

Implemented in F_HLL2, F_Kinetic, F_VFRoe, F_HLL, and F_Rusanov.

5.23.3.2 SCALAR Flux::get_cfl ( ) const

Gives the CFL value.

Returns

Flux::cfl value of the CFL.

Definition at line 96 of file flux.cpp.

5.23.3.3 SCALAR Flux::get_f1 ( ) const

Gives the first component of the numerical flux.

Returns

Flux::f1 first componant of the numerical flux.

Definition at line 76 of file flux.cpp.

5.23.3.4 SCALAR Flux::get_f2 ( ) const

Gives the second component of the numerical flux.

Returns

Flux::f2 second componant of the numerical flux.

Definition at line 86 of file flux.cpp.

5.23.3.5 void Flux::set_tx (SCALAR *tx*)

Sets the variable Flux::tx.

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

Parameters

in	*tx*	value of dt/dx.

Definition at line 65 of file flux.cpp.

5.23.4 Member Data Documentation

5.23.4.1 SCALAR Flux::cfl [protected]

CFL value.

Definition at line 103 of file flux.hpp.

5.23.4.2 SCALAR Flux::f1 [protected]

First component of the numerical flux.

Definition at line 99 of file flux.hpp.

5.23.4.3 SCALAR Flux::f2 [protected]

Second component of the numerical flux.

Definition at line 101 of file flux.hpp.

5.23.4.4 SCALAR Flux::tx [protected]

Value of dt/dx.

Definition at line 105 of file flux.hpp.

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

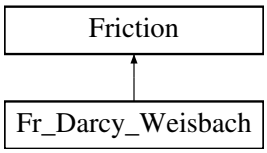
- Headers/libflux/flux.hpp
- Sources/libflux/flux.cpp

5.24 Fr_Darcy_Weisbach Class Reference

Darcy-Weisbach law.

`#include <fr_darcy_weisbach.hpp>`

Inheritance diagram for Fr_Darcy_Weisbach:



Public Member Functions

- Fr_Darcy_Weisbach ()

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR)

  Calculates the friction term.
- virtual ~Fr_Darcy_Weisbach ()

  Destructor.

Additional Inherited Members

5.24.1 Detailed Description

Darcy-Weisbach law.

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

Definition at line 70 of file fr_darcy_weisbach.hpp.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Fr_Darcy_Weisbach::Fr_Darcy_Weisbach ( )

Constructor.

Definition at line 58 of file fr_darcy_weisbach.cpp.

5.24.2.2 Fr_Darcy_Weisbach::~Fr_Darcy_Weisbach ( ) [virtual]

Destructor.

Definition at line 81 of file fr_darcy_weisbach.cpp.

5.24.3 Member Function Documentation

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

Calculates the friction term.

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

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

where f is the friction coefficient.

Parameters

in	*uold*	velocity at the previous time (n if you are calculating the $n+1$th time step), denoted by u_i^n in the above formula.
in	*hnew*	water height after the Shallow-Water cmoputation (without friction), denoted by h_i^{n+1} in the above formula.
in	*qnew*	discharge after the Shallow-Water computation (without friction), denoted by q_i^* in the above formula.

Modifies

friction::qmod discharge modified by the friction term.

Note

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

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

Implements Friction.

Definition at line 61 of file fr_darcy_weisbach.cpp.

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

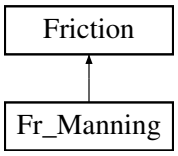
- Headers/libfrictions/fr_darcy_weisbach.hpp
- Sources/libfrictions/fr_darcy_weisbach.cpp

5.25 Fr_Manning Class Reference

Manning law.

`#include <fr_manning.hpp>`

Inheritance diagram for Fr_Manning:



Public Member Functions

- Fr_Manning ()

  Constructor.
- void calc (SCALAR, SCALAR, SCALAR)

  Calculates the Manning friction term.
- virtual ~Fr_Manning ()

  Destructor.

Additional Inherited Members

5.25.1 Detailed Description

Manning law.

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

Definition at line 71 of file fr_manning.hpp.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 Fr_Manning::Fr_Manning ( )

Constructor.

Definition at line 58 of file fr_manning.cpp.

5.25.2.2 Fr_Manning::~Fr_Manning ( ) [virtual]

Destructor.

Definition at line 81 of file fr_manning.cpp.

5.25.3 Member Function Documentation

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

Calculates the Manning friction term.

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

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

where K is the friction coefficient.

Parameters

in	*uold*	velocity at the previous time (n if you are calculating the $n+1$th time step), denoted by u_i^n in the above formula.
in	*hnew*	water height after the Shallow-Water cmoputation (without friction), denoted by h_i^{n+1} in the above formula.
in	*qnew*	discharge after the Shallow-Water computation (without friction), denoted by q_i^* in the above formula.

Modifies

friction::qmod discharge modified by the friction term.

Note

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

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

Implements Friction.

Definition at line 61 of file fr_manning.cpp.

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

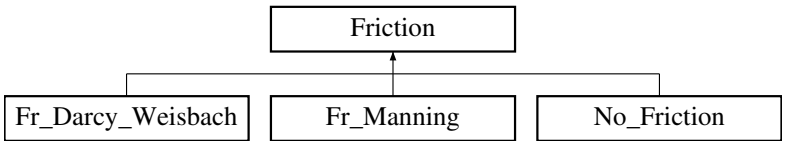
- Headers/libfrictions/fr_manning.hpp
- Sources/libfrictions/fr_manning.cpp

5.26 Friction Class Reference

Friction law

`#include <friction.hpp>`

Inheritance diagram for Friction:



Public Member Functions

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

Sets the time step Friction::dt.

- virtual ∼Friction ()

  Destructor.

Protected Attributes

- SCALAR qmod
- SCALAR c
- SCALAR dt

5.26.1 Detailed Description

Friction law

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

Definition at line 71 of file friction.hpp.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 Friction::Friction ( )

Constructor.

Definition at line 59 of file friction.cpp.

5.26.2.2 Friction::∼Friction ( ) [virtual]

Destructor.

Definition at line 96 of file friction.cpp.

5.26.3 Member Function Documentation

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

Function to be specified in each friction law.

Implemented in Fr_Manning, Fr_Darcy_Weisbach, and No_Friction.

5.26.3.2 SCALAR Friction::get_qmod ( ) const

Gives the discharge modified by the friction term.

Returns

Friction::qmod discharge modified by the friction term.

Definition at line 62 of file friction.cpp.

5.26.3.3 void Friction::set_c (SCALAR *c*)

Sets the friction coefficient Friction::c.

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

Parameters

in	*c*	value of the friction coefficient.

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

Definition at line 72 of file friction.cpp.

5.26.3.4 void Friction::set_dt (SCALAR *dt*)

Sets the time step Friction::dt.

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

Parameters

in	*dt*	value of the time step.

Definition at line 84 of file friction.cpp.

5.26.4 Member Data Documentation

5.26.4.1 SCALAR Friction::c [protected]

Friction coefficient.

Definition at line 98 of file friction.hpp.

5.26.4.2 SCALAR Friction::dt [protected]

Time step.

Definition at line 100 of file friction.hpp.

5.26.4.3 SCALAR Friction::qmod [protected]

Discharge modified by the friction term.

Definition at line 96 of file friction.hpp.

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

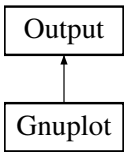
- Headers/libfrictions/friction.hpp
- Sources/libfrictions/friction.cpp

5.27 Gnuplot Class Reference

Gnuplot output

`#include <gnuplot.hpp>`

Inheritance diagram for Gnuplot:



Public Member Functions

- Gnuplot (Parameters &)

  Constructor.
- void write (VECT, VECT, VECT, int)

  Saves one time step.
- virtual ∼Gnuplot ()

  Destructor.

Additional Inherited Members

5.27.1 Detailed Description

Gnuplot output

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

Definition at line 70 of file gnuplot.hpp.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 Gnuplot::Gnuplot (Parameters & *par*)

Constructor.

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

Parameters

in	*par*	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.

Todo Exception should be treated.

Definition at line 58 of file gnuplot.cpp.

5.27.2.2 Gnuplot::~Gnuplot ( ) [virtual]

Destructor.

Definition at line 123 of file gnuplot.cpp.

5.27.3 Member Function Documentation

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

Saves one time step.

Writes the values of Scheme::h, Scheme::u (=q/h), Scheme::z, Scheme::q, Scheme::h+ Scheme::z (free surface), and the Froude number $\frac{q}{h\sqrt{gh}}$ at the current time in huz_evolution.dat.

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

Parameters

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

Implements Output.

Definition at line 92 of file gnuplot.cpp.

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

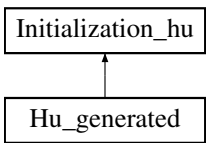
- Headers/libsave/gnuplot.hpp
- Sources/libsave/gnuplot.cpp

5.28 Hu_generated Class Reference

No water configuration.

`#include <hu_generated.hpp>`

Inheritance diagram for Hu_generated:



Public Member Functions

- Hu_generated (Parameters &)

  Constructor.
- void initialization (VECT &, VECT &)

  Performs the initialization.
- virtual ~Hu_generated ()

  Destructor.

Additional Inherited Members

5.28.1 Detailed Description

No water configuration.

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

Definition at line 72 of file hu_generated.hpp.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 Hu_generated::Hu_generated (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file hu_generated.cpp.

5.28.2.2 Hu_generated::~Hu_generated ( ) [virtual]

Destructor.

Definition at line 66 of file hu_generated.cpp.

5.28.3 Member Function Documentation

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

Performs the initialization.

Initializes the water height and the velocity at 0.

Parameters

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

Implements Initialization_hu.

Definition at line 69 of file hu_generated.cpp.

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

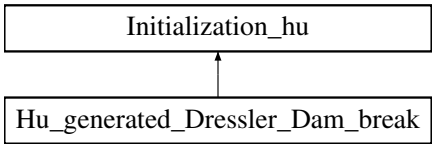
- Headers/libinitializations/hu_generated.hpp
- Sources/libinitializations/hu_generated.cpp

5.29 Hu_generated_Dressler_Dam_break Class Reference

Dressler dam break configuration.

`#include <hu_generated_dressler_dam_break.hpp>`

Inheritance diagram for Hu_generated_Dressler_Dam_break:



Public Member Functions

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

Additional Inherited Members

5.29.1 Detailed Description

Dressler dam break configuration.

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

Definition at line 71 of file hu_generated_dressler_dam_break.hpp.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 Hu_generated_Dressler_Dam_break::Hu_generated_Dressler_Dam_break (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 58 of file hu_generated_dressler_dam_break.cpp.

5.29.2.2 Hu_generated_Dressler_Dam_break::~Hu_generated_Dressler_Dam_break ( ) [virtual]

Destructor.

Definition at line 73 of file hu_generated_dressler_dam_break.cpp.

5.29.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Implements Initialization_hu.

Definition at line 76 of file hu_generated_dressler_dam_break.cpp.

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

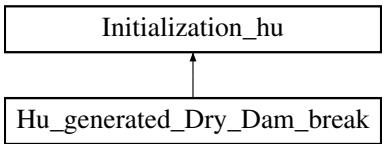
- Headers/libinitializations/hu_generated_dressler_dam_break.hpp
- Sources/libinitializations/hu_generated_dressler_dam_break.cpp

5.30 Hu_generated_Dry_Dam_break Class Reference

Dry dam break configuration.

`#include <hu_generated_dry_dam_break.hpp>`

Inheritance diagram for Hu_generated_Dry_Dam_break:



Public Member Functions

- Hu_generated_Dry_Dam_break (Parameters &)

  Constructor.
- void initialization (VECT &, VECT &)

  Performs the initialization.
- virtual ~Hu_generated_Dry_Dam_break ()

  Destructor.

Additional Inherited Members

5.30.1 Detailed Description

Dry dam break configuration.

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

Definition at line 71 of file hu_generated_dry_dam_break.hpp.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 Hu_generated_Dry_Dam_break::Hu_generated_Dry_Dam_break (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 59 of file hu_generated_dry_dam_break.cpp.

5.30.2.2 Hu_generated_Dry_Dam_break::~Hu_generated_Dry_Dam_break ( ) [virtual]

Destructor.

Definition at line 74 of file hu_generated_dry_dam_break.cpp.

5.30.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Implements Initialization_hu.

Definition at line 77 of file hu_generated_dry_dam_break.cpp.

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

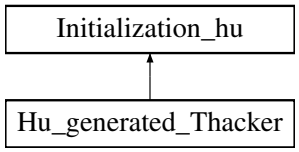
- Headers/libinitializations/hu_generated_dry_dam_break.hpp
- Sources/libinitializations/hu_generated_dry_dam_break.cpp

5.31 Hu_generated_Thacker Class Reference

Thacker configuration.

`#include <hu_generated_thacker.hpp>`

Inheritance diagram for Hu_generated_Thacker:



Public Member Functions

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

Additional Inherited Members

5.31.1 Detailed Description

Thacker configuration.

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

Definition at line 71 of file hu_generated_thacker.hpp.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 Hu_generated_Thacker::Hu_generated_Thacker (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 58 of file hu_generated_thacker.cpp.

5.31.2.2 Hu_generated_Thacker::~Hu_generated_Thacker ( ) [virtual]

Destructor.

Definition at line 75 of file hu_generated_thacker.cpp.

5.31.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Implements Initialization_hu.

Definition at line 77 of file hu_generated_thacker.cpp.

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

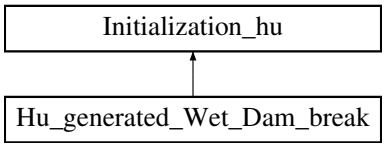
- Headers/libinitializations/hu_generated_thacker.hpp
- Sources/libinitializations/hu_generated_thacker.cpp

5.32 Hu_generated_Wet_Dam_break Class Reference

Wet dam break configuration.

`#include <hu_generated_wet_dam_break.hpp>`

Inheritance diagram for Hu_generated_Wet_Dam_break:



Public Member Functions

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

Additional Inherited Members

5.32.1 Detailed Description

Wet dam break configuration.

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

Definition at line 70 of file hu_generated_wet_dam_break.hpp.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 Hu_generated_Wet_Dam_break::Hu_generated_Wet_Dam_break (Parameters & *par*)

Constructor.

Defines the position of the dam (half of the domain), the water height before the dam (5 millimeters), the water height after the dam (1 millimeter) and the velocity (0 m/s), see Goutal and Maurel [1997].

Parameters

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

Definition at line 58 of file hu_generated_wet_dam_break.cpp.

5.32.2.2 Hu_generated_Wet_Dam_break::~Hu_generated_Wet_Dam_break ( ) [virtual]

Destructor.

Definition at line 73 of file hu_generated_wet_dam_break.cpp.

5.32.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Implements Initialization_hu.

Definition at line 75 of file hu_generated_wet_dam_break.cpp.

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

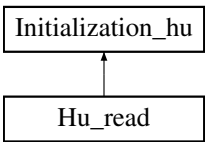
- Headers/libinitializations/hu_generated_wet_dam_break.hpp
- Sources/libinitializations/hu_generated_wet_dam_break.cpp

5.33 Hu_read Class Reference

File configuration.

`#include <hu_read.hpp>`

Inheritance diagram for Hu_read:



Public Member Functions

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

Additional Inherited Members

5.33.1 Detailed Description

File configuration.

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

Definition at line 71 of file hu_read.hpp.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 Hu_read::Hu_read (Parameters & *par*)

Constructor.

Defines the name of the file for the initialization.

Parameters

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

Definition at line 59 of file hu_read.cpp.

5.33.2.2 Hu_read::~Hu_read ( ) [virtual]

Destructor.

Definition at line 70 of file hu_read.cpp.

5.33.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

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

Warning

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

Note

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

Todo Exception should be treated.

Implements Initialization_hu.

Definition at line 73 of file hu_read.cpp.

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

- Headers/libinitializations/hu_read.hpp
- Sources/libinitializations/hu_read.cpp

5.34 Hydrostatic Class Reference

Hydrostatic reconstruction

`#include <hydrostatic.hpp>`

Public Member Functions

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

Protected Attributes

- SCALAR hl_rec
- SCALAR hr_rec

5.34.1 Detailed Description

Hydrostatic reconstruction

Class that computes the hydrostatic reconstruction.

Definition at line 68 of file hydrostatic.hpp.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 Hydrostatic::Hydrostatic ( )

Constructor.

Definition at line 57 of file hydrostatic.cpp.

5.34.2.2 Hydrostatic::~Hydrostatic ( ) [virtual]

Destructor.

Definition at line 97 of file hydrostatic.cpp.

5.34.3 Member Function Documentation

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

Calculates the hydrostatic reconstruction.

See Audusse et al. [2004] for more details.

Parameters

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

Modifies

Hydrostatic::hl_rec, set to $(hl\_0 - \max(0, dz))_+$.
Hydrostatic::hr_rec, set to $(hr\_0 - \max(0, -dz))_+$.

Definition at line 60 of file hydrostatic.cpp.

5.34.3.2 SCALAR Hydrostatic::get_hhydro_l ( ) const

Gives the reconstructed water height on the left.

Returns

Hydrostatic::hl_rec Hydrostatic reconstruction on the left.

Definition at line 77 of file hydrostatic.cpp.

5.34.3.3 SCALAR Hydrostatic::get_hhydro_r ( ) const

Gives the reconstructed water height on the right.

Returns

Hydrostatic::hr_rec Hydrostatic reconstruction on the right.

Definition at line 87 of file hydrostatic.cpp.

5.34.4 Member Data Documentation

5.34.4.1 SCALAR Hydrostatic::hl_rec [protected]

Hydrostatic reconstruction on the left

Definition at line 88 of file hydrostatic.hpp.

5.34.4.2 SCALAR Hydrostatic::hr_rec [protected]

Hydrostatic reconstruction on the right

Definition at line 90 of file hydrostatic.hpp.

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

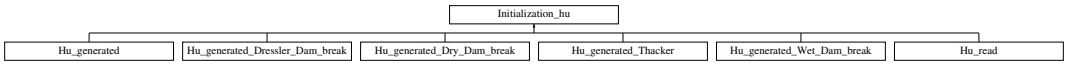
- Headers/libreconstructions/hydrostatic.hpp
- Sources/libreconstructions/hydrostatic.cpp

5.35 Initialization_hu Class Reference

Initialization of h and u.

`#include <initialization_hu.hpp>`

Inheritance diagram for Initialization_hu:



Public Member Functions

- Initialization_hu (Parameters &)

  Constructor.
- virtual void initialization (VECT &, VECT &)=0

  Function to be specified in each initialization.
- virtual ~Initialization_hu ()

  Destructor.

Protected Attributes

- const int NXCELL
- const int M
- const SCALAR DX

5.35.1 Detailed Description

Initialization of h and u.

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

Definition at line 69 of file initialization_hu.hpp.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 Initialization_hu::Initialization_hu (Parameters & *par*)

Constructor.

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

Parameters

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

Definition at line 58 of file initialization_hu.cpp.

5.35.2.2 Initialization_hu::~Initialization_hu ( ) [virtual]

Destructor.

Definition at line 70 of file initialization_hu.cpp.

5.35.3 Member Function Documentation

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

Function to be specified in each initialization.

Implemented in Hu_generated, Hu_generated_Dressler_Dam_break, Hu_generated_Dry_Dam_break, Hu_generated_Thacker, Hu_read, and Hu_generated_Wet_Dam_break.

5.35.4 Member Data Documentation

5.35.4.1 const SCALAR Initialization_hu::DX [protected]

Space step.

Definition at line 88 of file initialization_hu.hpp.

5.35.4.2 const int Initialization_hu::M [protected]

Number of time steps.

Definition at line 86 of file initialization_hu.hpp.

5.35.4.3 const int Initialization_hu::NXCELL [protected]

Number of cells in space.

Definition at line 84 of file initialization_hu.hpp.

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

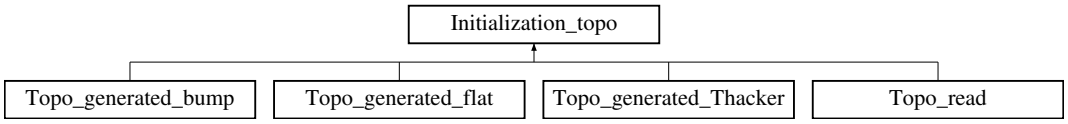
- Headers/libinitializations/initialization_hu.hpp
- Sources/libinitializations/initialization_hu.cpp

5.36 Initialization_topo Class Reference

Initialization of z.

`#include <initialization_topo.hpp>`

Inheritance diagram for Initialization_topo:



Public Member Functions

- Initialization_topo (Parameters &)

  Constructor.
- virtual void initialization (VECT &)=0

  Function to be specified in each initialization.
- virtual ~Initialization_topo ()

  Destructor.

Protected Attributes

- VECT z
- const int NXCELL
- const int M
- const SCALAR DX

5.36.1 Detailed Description

Initialization of z.

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

Definition at line 69 of file initialization_topo.hpp.

5.36.2 Constructor & Destructor Documentation

5.36.2.1 Initialization_topo::Initialization_topo (Parameters & *par*)

Constructor.

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

Parameters

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

Warning

Problem: allocation of z failed.

Note

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

Todo Exception should be treated.

Definition at line 58 of file initialization_topo.cpp.

5.36.2.2 Initialization_topo::~Initialization_topo ( ) [virtual]

Destructor.

Definition at line 78 of file initialization_topo.cpp.

5.36.3 Member Function Documentation

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

Function to be specified in each initialization.

Implemented in Topo_read, Topo_generated_flat, Topo_generated_Thacker, and Topo_generated_bump.

5.36.4 Member Data Documentation

5.36.4.1 const SCALAR Initialization_topo::DX [protected]

Space step.

Definition at line 91 of file initialization_topo.hpp.

5.36.4.2 const int Initialization_topo::M [protected]

Number of time steps.

Definition at line 89 of file initialization_topo.hpp.

5.36.4.3 const int Initialization_topo::NXCELL [protected]

Number of cells in space.

Definition at line 87 of file initialization_topo.hpp.

5.36.4.4 VECT Initialization_topo::z [protected]

Definition at line 85 of file initialization_topo.hpp.

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

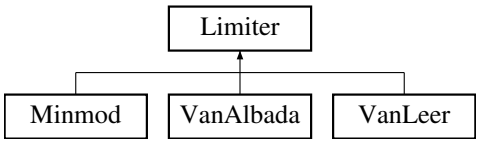
- Headers/libinitializations/initialization_topo.hpp
- Sources/libinitializations/initialization_topo.cpp

5.37 Limiter Class Reference

Slope limiter.

`#include <limiter.hpp>`

Inheritance diagram for Limiter:



Public Member Functions

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

Protected Attributes

- SCALAR rec
  Reconstructed value.

5.37.1 Detailed Description

Slope limiter.

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

Definition at line 69 of file limiter.hpp.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 Limiter::Limiter ( )

Constructor.

Definition at line 58 of file limiter.cpp.

5.37.2.2 Limiter::~Limiter ( ) [virtual]

Destructor.

Definition at line 72 of file limiter.cpp.

5.37.3 Member Function Documentation

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

Function to be specified in each slope limiter.

Implemented in VanLeer, Minmod, and VanAlbada.

5.37.3.2 SCALAR Limiter::get_rec ( ) const

Gives the reconstructed value.

Returns

Limiter::rec reconstructed value.

Definition at line 62 of file limiter.cpp.

5.37.4 Member Data Documentation

5.37.4.1 SCALAR Limiter::rec [protected]

Reconstructed value.

Definition at line 88 of file limiter.hpp.

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

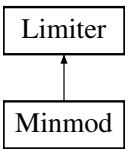
- Headers/liblimitations/limiter.hpp
- Sources/liblimitations/limiter.cpp

5.38 Minmod Class Reference

Minmod slope limiter

`#include <minmod.hpp>`

Inheritance diagram for Minmod:



Public Member Functions

- Minmod ()

  Constructor.
- void calc (SCALAR, SCALAR)

  Calculates the value of the slope limiter.
- virtual ∼Minmod ()

  Destructor.

Additional Inherited Members

5.38.1 Detailed Description

Minmod slope limiter

Class that calculates the minmod slope limiter.

Definition at line 69 of file minmod.hpp.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 Minmod::Minmod ( )

Constructor.

Definition at line 58 of file minmod.cpp.

5.38.2.2 Minmod::∼Minmod ( ) [virtual]

Destructor.

Definition at line 85 of file minmod.cpp.

5.38.3 Member Function Documentation

5.38.3.1 void Minmod::calc (SCALAR *a*, SCALAR *b*) [virtual]

Calculates the value of the slope limiter.

Minmod function:

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

Parameters

in	*a*	slope on the left of the cell.
in	*b*	slope on the right of the cell.

Modifies

Limiter::rec recontructed value.

Implements Limiter.

Definition at line 61 of file minmod.cpp.

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

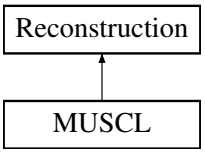
- Headers/liblimitations/minmod.hpp
- Sources/liblimitations/minmod.cpp

5.39 MUSCL Class Reference

MUSCL recontruction

`#include <muscl.hpp>`

Inheritance diagram for MUSCL:



Public Member Functions

- MUSCL (Parameters &, VECT)

  Constructor.
- void calc (VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &)

  Calculates the reconstruction in space.
- ~MUSCL ()

  Destructor.

Additional Inherited Members

5.39.1 Detailed Description

MUSCL recontruction

Class that computes MUSCL reconstruction in space.

Definition at line 70 of file muscl.hpp.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 MUSCL::MUSCL (Parameters & *par,* VECT *z*)

Constructor.

Definition at line 58 of file muscl.cpp.

5.39.2.2 MUSCL::~MUSCL ( )

Destructor.

Definition at line 131 of file muscl.cpp.

5.39.3 Member Function Documentation

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

Calculates the reconstruction in space.

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

Parameters

in	*h*	water height.
in	*u*	velocity of the flow.
in	*z*	topography.
out	*hr*	reconstructed water height on the right of the cell.
out	*hl*	reconstructed water height on the left of the cell.
out	*delz*	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	*ur*	reconstructed velocity on the right of the cell.
out	*ul*	reconstructed velocity on the left of the cell.
out	*dzi*	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Note

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

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

Implements Reconstruction.

Definition at line 61 of file muscl.cpp.

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

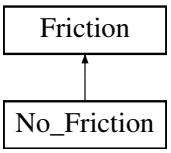
- Headers/libreconstructions/muscl.hpp
- Sources/libreconstructions/muscl.cpp

5.40 No_Friction Class Reference

No friction.

`#include <no_friction.hpp>`

Inheritance diagram for No_Friction:



Public Member Functions

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

Additional Inherited Members

5.40.1 Detailed Description

No friction.

Does no computation.

Definition at line 69 of file no_friction.hpp.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 No_Friction::No_Friction ( )

Constructor.

Definition at line 57 of file no_friction.cpp.

5.40.2.2 No_Friction::~No_Friction ( ) [virtual]

Destructor.

Definition at line 73 of file no_friction.cpp.

5.40.3 Member Function Documentation

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

Does no calculation.

No computation (no friction).

Modifies

friction::qmod discharge modified by the friction term.

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

Implements Friction.

Definition at line 60 of file no_friction.cpp.

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

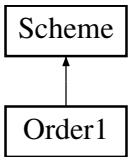
- Headers/libfrictions/no_friction.hpp
- Sources/libfrictions/no_friction.cpp

5.41 Order1 Class Reference

Order 1 scheme.

`#include <order1.hpp>`

Inheritance diagram for Order1:



Public Member Functions

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

Additional Inherited Members

5.41.1 Detailed Description

Order 1 scheme.

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

Definition at line 69 of file order1.hpp.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 Order1::Order1 (Parameters & *par*)

Constructor.

Definition at line 58 of file order1.cpp.

5.41.2.2 Order1::~Order1 ( ) [virtual]

Destructor.

Definition at line 146 of file order1.cpp.

5.41.3 Member Function Documentation

5.41.3.1 void Order1::calc ( ) [virtual]

Performs the numerical scheme.

Performs the first order numerical scheme.

Note

In DEBUG mode, the programme will save another file with boundary fluxes.

Implements Scheme.

Definition at line 64 of file order1.cpp.

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

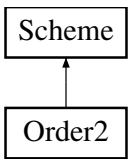
- Headers/libschemes/order1.hpp
- Sources/libschemes/order1.cpp

5.42 Order2 Class Reference

Order 2 scheme.

`#include <order2.hpp>`

Inheritance diagram for Order2:



Public Member Functions

- Order2 (Parameters &)

  Constructor.
- void calc ()

  Performs the numerical scheme.
- virtual ~Order2 ()

  Destructor.

Additional Inherited Members

5.42.1 Detailed Description

Order 2 scheme.

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

Definition at line 69 of file order2.hpp.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 Order2::Order2 (Parameters & *par*)

Constructor.

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

Parameters

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

Warning

Problem: allocation of ∗sa failed.

Note

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

Todo Exception should be treated.

Definition at line 59 of file order2.cpp.

5.42.2.2 Order2::∼Order2 ( ) [virtual]

Destructor.

Definition at line 188 of file order2.cpp.

5.42.3 Member Function Documentation

5.42.3.1 void Order2::calc ( ) [virtual]

Performs the numerical scheme.

Performs the second order numerical scheme.

Note

In DEBUG mode, the programme will save another file with boundary fluxes.

Implements Scheme.

Definition at line 101 of file order2.cpp.

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

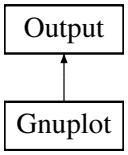
- Headers/libschemes/order2.hpp
- Sources/libschemes/order2.cpp

5.43 Output Class Reference

Output format

`#include <output.hpp>`

Inheritance diagram for Output:



Public Member Functions

- Output (Parameters &)

  Constructor.
- virtual void write (VECT, VECT, VECT, int)=0

  Function to be specified in each output format.
- void initial (VECT, VECT, VECT)

  Saves the initial time.
- void result (SCALAR, clock_t, SCALAR)

  Saves global values.
- void bound_flux (int, SCALAR, SCALAR, SCALAR, SCALAR)

  Saves the flux and the cumulative flux on the boundaries.
- void save (VECT, VECT, VECT)

  Saves the final time.
- virtual ~Output ()

  Destructor.

Protected Attributes

- const int NXCELL
- const int M
- const SCALAR DX
- const SCALAR DT
- const int NBTIMES
- int nbt
- string outputDirectory
- string namefile_init
- string namefile_end
- string namefile_res
- string namefile_flux

5.43.1 Detailed Description

Output format

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

Definition at line 68 of file output.hpp.

5.43.2 Constructor & Destructor Documentation

5.43.2.1 Output::Output (Parameters & *par*)

Constructor.

Defines the names of the outputs.

If run in DEBUG mode, writes the header of the file 'boundary_flux.dat'.

Parameters

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

Warning

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

Note

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

Todo Exception should be treated.

Definition at line 57 of file output.cpp.

5.43.2.2 Output::~Output ( ) [virtual]

Destructor.

Definition at line 100 of file output.cpp.

5.43.3 Member Function Documentation

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

Saves the flux and the cumulative flux on the boundaries.

Parameters

in	*tps*	current time.
in	*f_l*	flux on the left boundary.
in	*f_r*	flux on the right boundary.
in	*cum_f_l*	cumulative flux on the left boundary.
in	*cum_f_r*	cumulative flux on the right boundary.

Definition at line 132 of file output.cpp.

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

Saves the initial time.

Parameters

in	*h*	water height.
in	*u*	velocity.
in	*z*	topography.

Warning

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

Note

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

Todo Exception should be treated.

Definition at line 103 of file output.cpp.

5.43.3.3 void Output::result (SCALAR *t*, clock_t *cpu*, SCALAR *froude*)

Saves global values.

Parameters

in	*t*	elapsed time.
in	*cpu*	CPU time.
in	*froude*	mean Froude number (in space) at the final time.

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.

Todo Exception should be treated.

Definition at line 183 of file output.cpp.

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

Saves the final time.

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

Parameters

in	*z*	topography.
in	*h*	water height.
in	*q*	discharge.

Warning

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

Note

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

Todo Exception should be treated.

Definition at line 148 of file output.cpp.

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

Function to be specified in each output format.

Implemented in Gnuplot.

5.43.4 Member Data Documentation

5.43.4.1 const SCALAR Output::DT [protected]

Time step.

Definition at line 102 of file output.hpp.

5.43.4.2 const SCALAR Output::DX [protected]

Space step.

Definition at line 100 of file output.hpp.

5.43.4.3 const int Output::M [protected]

Number of time steps.

Definition at line 98 of file output.hpp.

5.43.4.4 string Output::namefile_end [protected]

Name of the file where the final time is saved.

Definition at line 112 of file output.hpp.

5.43.4.5 string Output::namefile_flux [protected]

Name of the file where the boundary fluxes are saved.

Definition at line 116 of file output.hpp.

5.43.4.6 string Output::namefile_init [protected]

Name of the file where the initialization is saved.

Definition at line 110 of file output.hpp.

5.43.4.7 string Output::namefile_res [protected]

Name of the file where the global results are saved.

Definition at line 114 of file output.hpp.

5.43.4.8 int Output::nbt [protected]

Number of times steps between two savings.

Definition at line 106 of file output.hpp.

5.43.4.9 const int Output::NBTIMES [protected]

Number of times saved.

Definition at line 104 of file output.hpp.

5.43.4.10 const int Output::NXCELL [protected]

Number of cells in space.

Definition at line 96 of file output.hpp.

5.43.4.11 string Output::outputDirectory [protected]

Name of the output directory.

Definition at line 108 of file output.hpp.

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

- Headers/libsave/output.hpp
- Sources/libsave/output.cpp

5.44 Parameters Class Reference

Gets parameters.

`#include <parameters.hpp>`

Public Member Functions

- Parameters ()
  Constructor.
- virtual ~Parameters ()
  Destructor.

- void setparameters (const char *)

  Sets the parameters.
- int get_Nxcell () const

  Gives the number of cells in space.
- int get_m () const

  Gives the number of time steps to attain the final time.
- int get_nbtimes () const

  Gives the number of times saved.
- SCALAR get_dx () const

  Gives the space step.
- SCALAR get_dt () const

  Gives the time step.
- SCALAR get_tx () const

  Gives the value of dt/dx.
- int get_Lbound () const

  Gives the value corresponding to the left boundary condition.
- SCALAR get_L_imp_q () const

  Gives the value of the imposed discharge in left bc.
- SCALAR get_L_imp_h () const

  Gives the value of the imposed water height in left bc.
- int get_Rbound () const

  Gives the value corresponding to the right boundary condition.
- SCALAR get_R_imp_q () const

  Gives the value of the imposed discharge in right bc.
- SCALAR get_R_imp_h () const

  Gives the value of the imposed water height in right bc.
- int get_flux () const

  Gives the value corresponding to the flux.
- int get_order () const

  Gives the order of the scheme.
- SCALAR get_cfl () const

  Gives the cfl of the scheme.
- int get_rec () const

  Gives the value corresponding to the reconstruction.
- int get_fric () const

  Gives the value corresponding to the friction law.
- SCALAR get_friccoef () const

  Gives the value of the friction coefficient.
- int get_lim () const

  Gives the value corresponding to the limiter.
- int get_topo () const

  Gives the value corresponding to the topography.
- int get_hu () const

  Gives the value corresponding to the initialization of h and u.
- SCALAR get_amortENO () const

  Gives the value of the amortENO parameter.
- SCALAR get_modifENO () const

Gives the value of the modifENO parameter.

- string get_topographyNameFile () const

  Gives the topography path + Input directory.

- string get_huNameFile (void) const

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

- string get_topographyNameFileS () const

  Gives the topography namefile (inside the Input directory)

- string get_huNameFileS (void) const

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

- string get_outputDirectory (void) const

  Gives the output directory with the suffix.

- string get_suffix (void) const

  Gives the suffix.

Protected Attributes

- int Nxcell
- int m
- int nbtimes
- SCALAR dx
- SCALAR dt
- SCALAR tx
- SCALAR L
- SCALAR T
- SCALAR cfl
- int Lbound
- int Rbound
- SCALAR L_imp_q
- SCALAR L_imp_h
- SCALAR R_imp_q
- SCALAR R_imp_h
- int flux
- int order
- int rec
- int fric
- int lim
- int topo
- int hu_init
- SCALAR amortENO
- SCALAR modifENO
- SCALAR friccoef
- string topography_namefile
- string topo_NF
- string hu_namefile
- string hu_NF
- string output_directory
- string suffix_o

5.44.1 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 70 of file parameters.hpp.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 Parameters::Parameters ( )

Constructor.

Definition at line 62 of file parameters.cpp.

5.44.2.2 Parameters::~Parameters ( ) [virtual]

Destructor.

Definition at line 662 of file parameters.cpp.

5.44.3 Member Function Documentation

5.44.3.1 SCALAR Parameters::get_amortENO ( ) const

Gives the value of the amortENO parameter.

Returns

The value of the amortENO parameter Parameters::amortENO.

Definition at line 855 of file parameters.cpp.

5.44.3.2 SCALAR Parameters::get_cfl ( ) const

Gives the cfl of the scheme.

Returns

The cfl of the scheme Parameters::cfl.

Definition at line 805 of file parameters.cpp.

5.44.3.3 SCALAR Parameters::get_dt ( ) const

Gives the time step.

Returns

The time step Parameters::dt.

Definition at line 705 of file parameters.cpp.

5.44.3.4 SCALAR Parameters::get_dx ( ) const

Gives the space step.

Returns

The space step Parameters::dx.

Definition at line 685 of file parameters.cpp.

5.44.3.5 int Parameters::get_flux ( ) const

Gives the value corresponding to the flux.

Returns

The value corresponding to the flux Parameters::flux.

Definition at line 785 of file parameters.cpp.

5.44.3.6 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 825 of file parameters.cpp.

5.44.3.7 SCALAR Parameters::get_friccoef ( ) const

Gives the value of the friction coefficient.

Returns

The value of the friction coefficient Parameters::friccoef.

Definition at line 835 of file parameters.cpp.

5.44.3.8 int Parameters::get_hu ( ) const

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

Returns

The value corresponding to the initialization of h and u Parameters::hu_init.

Definition at line 925 of file parameters.cpp.

5.44.3.9 string Parameters::get_huNameFile (void) const

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

Returns

The h and u path for the initialization + Input directory Parameters::hu_namefile.

Definition at line 905 of file parameters.cpp.

5.44.3.10 string Parameters::get_huNameFileS (void) const

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

Returns

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

Definition at line 915 of file parameters.cpp.

5.44.3.11 SCALAR Parameters::get_L_imp_h ( ) const

Gives the value of the imposed water height in left bc.

Returns

The value of the imposed water height in the left boundary condition Parameters::L_imp_h.

Definition at line 745 of file parameters.cpp.

5.44.3.12 SCALAR Parameters::get_L_imp_q ( ) const

Gives the value of the imposed discharge in left bc.

Returns

The value of the imposed discharge in the left boundary condition Parameters::L_imp_q.

Definition at line 735 of file parameters.cpp.

5.44.3.13 int Parameters::get_Lbound ( ) const

Gives the value corresponding to the left boundary condition.

Returns

The value corresponding to the left boundary condition Parameters::Lbound.

Definition at line 725 of file parameters.cpp.

5.44.3.14 int Parameters::get_lim ( ) const

Gives the value corresponding to the limiter.

Returns

The value corresponding to the limiter Parameters::lim.

Definition at line 845 of file parameters.cpp.

5.44.3.15 int Parameters::get_m ( ) const

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

Returns

The number of time steps to attain the final time Parameters::m.

Definition at line 675 of file parameters.cpp.

5.44.3.16 SCALAR Parameters::get_modifENO ( ) const

Gives the value of the modifENO parameter.

Returns

The value of the modifENO parameter Parameters::modifENO.

Definition at line 865 of file parameters.cpp.

5.44.3.17 int Parameters::get_nbtimes ( ) const

Gives the number of times saved.

Returns

The number of times saved Parameters::nbtimes.

Definition at line 695 of file parameters.cpp.

5.44.3.18 int Parameters::get_Nxcell ( ) const

Gives the number of cells in space.

Returns

The number of cells in space Parameters::Nxcell.

Definition at line 665 of file parameters.cpp.

5.44.3.19 int Parameters::get_order ( ) const

Gives the order of the scheme.

Returns

The order of the scheme Parameters::order.

Definition at line 795 of file parameters.cpp.

5.44.3.20 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 935 of file parameters.cpp.

5.44.3.21 SCALAR Parameters::get_R_imp_h ( ) const

Gives the value of the imposed water height in right bc.

Returns

The value of the imposed water height in the right boundary condition Parameters::R_imp_h.

Definition at line 775 of file parameters.cpp.

5.44.3.22 SCALAR Parameters::get_R_imp_q ( ) const

Gives the value of the imposed discharge in right bc.

Returns

The value of the imposed discharge in the right boundary condition Parameters::R_imp_q.

Definition at line 765 of file parameters.cpp.

5.44.3.23 int Parameters::get_Rbound ( ) const

Gives the value corresponding to the right boundary condition.

Returns

The value corresponding to the right boundary condition Parameters::Rbound.

Definition at line 755 of file parameters.cpp.

5.44.3.24 int Parameters::get_rec ( ) const

Gives the value corresponding to the reconstruction.

Returns

The value corresponding to the reconstruction Parameters::rec.

Definition at line 815 of file parameters.cpp.

5.44.3.25 string Parameters::get_suffix (void) const

Gives the suffix.

Returns

The suffix (for the output directory) Parameters::suffix_o.

Definition at line 945 of file parameters.cpp.

5.44.3.26 int Parameters::get_topo ( ) const

Gives the value corresponding to the topography.

Returns

The value corresponding to the topography Parameters::topo.

Definition at line 895 of file parameters.cpp.

5.44.3.27 string Parameters::get_topographyNameFile (void) const

Gives the topography path + Input directory.

Returns

The topography path + Input directory Parameters::topography_namefile.

Definition at line 875 of file parameters.cpp.

5.44.3.28 string Parameters::get_topographyNameFileS (void) const

Gives the topography namefile (inside the Input directory)

Returns

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

Definition at line 885 of file parameters.cpp.

5.44.3.29 SCALAR Parameters::get_tx ( ) const

Gives the value of dt/dx.

Returns

The value of dt/dx Parameters::tx.

Definition at line 715 of file parameters.cpp.

5.44.3.30 void Parameters::setparameters (const char * *FILENAME*)

Sets the parameters.

Gets all the parameters from the file FILENAME, check and affect them. The values used by FullSWOF_1D are saved in the file parameters.dat. These values are also printed in the terminal when the code is run.

Parameters

in	*FILENAME*	name of the parameters file.

Warning

parameters.txt: ERROR: ***.
parameters.txt: WARNING: ***.
ERROR: the *** file *** does not exists in the directory Inputs.
Impossible to open the *** file. Verify if the directory *** exists.

Note

If a values 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.

Todo Exception should be treated.

Definition at line 66 of file parameters.cpp.

5.44.4 Member Data Documentation

5.44.4.1 SCALAR Parameters::amortENO [protected]

Parameter for eno.

Definition at line 217 of file parameters.hpp.

5.44.4.2 SCALAR Parameters::cfl [protected]

CFL value.

Definition at line 189 of file parameters.hpp.

5.44.4.3 SCALAR Parameters::dt [protected]

Time step.

Definition at line 181 of file parameters.hpp.

5.44.4.4 SCALAR Parameters::dx [protected]

Space step.

Definition at line 179 of file parameters.hpp.

5.44.4.5 int Parameters::flux [protected]

Numerical flux.

Definition at line 203 of file parameters.hpp.

5.44.4.6 int Parameters::fric [protected]

Friction.

Definition at line 209 of file parameters.hpp.

5.44.4.7 SCALAR Parameters::friccoef [protected]

Friction coefficient.

Definition at line 221 of file parameters.hpp.

5.44.4.8 int Parameters::hu_init [protected]

Type of initial conditions for h and u.

Definition at line 215 of file parameters.hpp.

5.44.4.9 string Parameters::hu_namefile [protected]

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

Definition at line 227 of file parameters.hpp.

5.44.4.10 string Parameters::hu_NF [protected]

Name of the file for the initialization without 'Inpouts'.

Definition at line 229 of file parameters.hpp.

5.44.4.11 SCALAR Parameters::L [protected]

Length of the domain.

Definition at line 185 of file parameters.hpp.

5.44.4.12 SCALAR Parameters::L_imp_h [protected]

Imposed water height on the left boundary.

Definition at line 197 of file parameters.hpp.

5.44.4.13 SCALAR Parameters::L_imp_q [protected]

Imposed discharge on the left boundary.

Definition at line 195 of file parameters.hpp.

5.44.4.14 int Parameters::Lbound [protected]

Left boundary condition.

Definition at line 191 of file parameters.hpp.

5.44.4.15 int Parameters::lim [protected]

Slope limiter.

Definition at line 211 of file parameters.hpp.

5.44.4.16 int Parameters::m [protected]

Number of time steps.

Definition at line 175 of file parameters.hpp.

5.44.4.17 SCALAR Parameters::modifENO [protected]

Parameter for eno_modif.

Definition at line 219 of file parameters.hpp.

5.44.4.18 int Parameters::nbtimes [protected]

Number of times saved.

Definition at line 177 of file parameters.hpp.

5.44.4.19 int Parameters::Nxcell [protected]

Number of cells in space.

Definition at line 173 of file parameters.hpp.

5.44.4.20 int Parameters::order [protected]

Order of the numerical scheme.

Definition at line 205 of file parameters.hpp.

5.44.4.21 string Parameters::output_directory [protected]

Name of the output directory Outputs+suffix.

Definition at line 231 of file parameters.hpp.

5.44.4.22 SCALAR Parameters::R_imp_h [protected]

Imposed water height on the right boundary.

Definition at line 201 of file parameters.hpp.

5.44.4.23 SCALAR Parameters::R_imp_q [protected]

Imposed discharge on the right boundary.

Definition at line 199 of file parameters.hpp.

5.44.4.24 int Parameters::Rbound [protected]

Right boundary condition.

Definition at line 193 of file parameters.hpp.

5.44.4.25 int Parameters::rec [protected]

Reconstruction.

Definition at line 207 of file parameters.hpp.

5.44.4.26 string Parameters::suffix_o [protected]

Suffix for the output directory.

Definition at line 233 of file parameters.hpp.

5.44.4.27 SCALAR Parameters::T [protected]

Final time.

Definition at line 187 of file parameters.hpp.

5.44.4.28 int Parameters::topo [protected]

Type of topography.

Definition at line 213 of file parameters.hpp.

5.44.4.29 string Parameters::topo_NF [protected]

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

Definition at line 225 of file parameters.hpp.

5.44.4.30 string Parameters::topography_namefile [protected]

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

Definition at line 223 of file parameters.hpp.

5.44.4.31 SCALAR Parameters::tx [protected]

Ratio dt/dx.

Definition at line 183 of file parameters.hpp.

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

- Headers/libparameters/parameters.hpp
- Sources/libparameters/parameters.cpp

5.45 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.

5.45.1 Detailed Description

Parser to read the entries

Class that reads the input file writen as description <variable>:: value # comment and keep the values after the "::" ignoring the comments that begin with a "#".

Definition at line 79 of file parser.hpp.

5.45.2 Constructor & Destructor Documentation

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

Constructor.

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

Parameters

in	*FILENAME*	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.

Todo Exception should be treated.

Definition at line 57 of file parser.cpp.

5.45.2.2 Parser::~Parser ( ) [virtual]

Destructor.

Definition at line 170 of file parser.cpp.

5.45.3 Member Function Documentation

5.45.3.1 string Parser::GetValue (const char * *TAG*)

Returns the value of the variable.

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

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.

Todo Exception should be treated.

For each variable, returns the value of the variable writen in the input tabular Inputs: TAG, the name of the variable with delimiters Outputs: value,

Definition at line 114 of file parser.cpp.

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

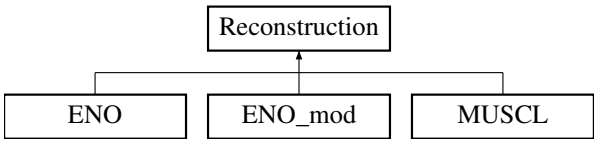
- Headers/libparser/parser.hpp
- Sources/libparser/parser.cpp

5.46 Reconstruction Class Reference

Reconstruction of the variables

`#include <reconstruction.hpp>`

Inheritance diagram for Reconstruction:



Public Member Functions

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

Protected Attributes

- const int NXCELL
  Number of cells in space.
- VECT zr
  Reconstructed topography on the right boundary.
- VECT zl
  Reconstructed topography on the left boundary.
- VECT delta0_z
  Difference between the values of the topography on two adjacent cells (on the right)
- Choice_limiter ∗ limiter
  Slope limiter.

5.46.1 Detailed Description

Reconstruction of the variables

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

Definition at line 73 of file reconstruction.hpp.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 Reconstruction::Reconstruction (Parameters & *par,* VECT *z*)

Constructor.

Defines the number of cells, the slope limiter, and initializes Reconstruction::zl, Reconstruction::zr, Reconstruction::delta0_z.

Parameters

in	*par*	parameter, contains all the values from the parameters file.
in	*z*	topography.

Warning

Problem: allocation of $***$ failed.

Note

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

Todo Exception should be treated.

Definition at line 58 of file reconstruction.cpp.

5.46.2.2 Reconstruction::~Reconstruction ( ) [virtual]

Destructor.

Definition at line 98 of file reconstruction.cpp.

5.46.3 Member Function Documentation

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

Function to be specified in each reconstruction.

Implemented in ENO, MUSCL, and ENO_mod.

5.46.4 Member Data Documentation

5.46.4.1 VECT Reconstruction::delta0_z [protected]

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

Definition at line 95 of file reconstruction.hpp.

5.46.4.2 Choice_limiter* Reconstruction::limiter [protected]

Slope limiter.

Definition at line 97 of file reconstruction.hpp.

5.46.4.3 const int Reconstruction::NXCELL [protected]

Number of cells in space.

Definition at line 89 of file reconstruction.hpp.

5.46.4.4 VECT Reconstruction::zl [protected]

Reconstructed topography on the left boundary.

Definition at line 93 of file reconstruction.hpp.

5.46.4.5 VECT Reconstruction::zr [protected]

Reconstructed topography on the right boundary.

Definition at line 91 of file reconstruction.hpp.

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

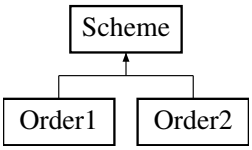
- Headers/libreconstructions/reconstruction.hpp
- Sources/libreconstructions/reconstruction.cpp

5.47 Scheme Class Reference

Numerical scheme.

`#include <scheme.hpp>`

Inheritance diagram for Scheme:



Public Member Functions

- Scheme (Parameters &)

  Constructor.
- virtual void calc ()=0

  Function to be specified in each numerical scheme.
- void allocation ()
- void deallocation ()
- void maincalc (VECT, VECT, VECT &, VECT &, VECT &, SCALAR &, SCALAR &)

  Main calculation of the scheme.
- SCALAR froude_number (VECT, VECT)

  Computation of the mean value of the Froude number.
- void boundary (VECT &, VECT &, const SCALAR, const SCALAR, const SCALAR, const SCALAR, SCALAR, const int)

  Calls the boundary conditions and affects the boundary values.
- virtual ~Scheme ()

  Destructor.

Protected Attributes

- const int NXCELL
- const int M
- const SCALAR DT
- const SCALAR TX
- const int NBTIMES
- const SCALAR CFL
- const SCALAR L_IMP_Q
- const SCALAR L_IMP_H
- const SCALAR R_IMP_Q
- const SCALAR R_IMP_H
- int nbt
- VECT h
- VECT u
- VECT q
- VECT z
- VECT delta_z
- VECT dzi
- VECT f1
- VECT f2
- VECT hs
- VECT us
- VECT qs
- VECT hl
- VECT hr
- VECT ul
- VECT ur
- VECT hleft
- VECT hright
- SCALAR flux_l
- SCALAR flux_r
- SCALAR cum_flux_l
- SCALAR cum_flux_r
- SCALAR Fr
- int time_initial
- time_t start
- time_t end
- SCALAR elapsed_time
- clock_t cpu_time
- Hydrostatic rec_hydro
- Choice_condition * Lbound
- Choice_condition * Rbound
- Choice_flux * flux_num
- Choice_friction * fric
- Choice_init_topo * topo
- Choice_init_hu * hu_init
- Choice_output * out

5.47.1 Detailed Description

Numerical scheme.

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

Definition at line 97 of file scheme.hpp.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 Scheme::Scheme (Parameters & *par*)

Constructor.

Initializations and allocations.

Parameters

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

Definition at line 58 of file scheme.cpp.

5.47.2.2 Scheme::~Scheme ( ) [virtual]

Destructor.

Definition at line 221 of file scheme.cpp.

5.47.3 Member Function Documentation

5.47.3.1 void Scheme::allocation ( )

Allocation of spatialized variables

Warning

Problem: allocation of $***$ failed.

Note

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

Todo Exception should be treated.

Definition at line 229 of file scheme.cpp.

5.47.3.2 void Scheme::boundary (VECT & *h_tmp*, VECT & *u_tmp*, const SCALAR *L_IMP_Q*, const SCALAR *L_IMP_H*, const SCALAR *R_IMP_Q*, const SCALAR *R_IMP_H*, SCALAR *temps_tmp*, const int *nodex*)

Calls the boundary conditions and affects the boundary values.

Parameters

in,out	*h_tmp*	water height.
in,out	*u_tmp*	velocity.
in	*L_IMP_Q*	left imposed discharge.
in	*L_IMP_H*	left imposed water height.
in	*R_IMP_Q*	right imposed discharge.
in	*R_IMP_H*	right imposed water height.
in	*temps_tmp*	current time.
in	*nodex*	number of space cells.

Definition at line 165 of file scheme.cpp.

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

Function to be specified in each numerical scheme.

Implemented in Order1, and Order2.

5.47.3.4 void Scheme::deallocation ( )

Deallocation of variables

Definition at line 344 of file scheme.cpp.

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

Computation of the mean value of the Froude number.

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

Parameters

in	*h*	water height.
in	*u*	velocity.

Returns

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

Definition at line 190 of file scheme.cpp.

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

Main calculation of the scheme.

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

Parameters

in	*he*	water height.
in	*ve*	velocity.

out	*hes*	water height after one step of the scheme.
out	*ves*	velocity after one step of the scheme.
out	*qes*	discharge after one step of the scheme.
out	*flux_l*	Flux on the left.
out	*flux_r*	Flux on the right.

Todo Improve the treatment of numerical errors.

Definition at line 106 of file scheme.cpp.

5.47.4 Member Data Documentation

5.47.4.1 const SCALAR Scheme::CFL [protected]

CFL value.

Definition at line 138 of file scheme.hpp.

5.47.4.2 clock_t Scheme::cpu_time [protected]

CPU time.

Definition at line 202 of file scheme.hpp.

5.47.4.3 SCALAR Scheme::cum_flux_l [protected]

Cumulative flux on the left.

Definition at line 188 of file scheme.hpp.

5.47.4.4 SCALAR Scheme::cum_flux_r [protected]

Cumulative flux on the right.

Definition at line 190 of file scheme.hpp.

5.47.4.5 VECT Scheme::delta_z [protected]

Variations of the topography.

Definition at line 158 of file scheme.hpp.

5.47.4.6 const SCALAR Scheme::DT [protected]

Time step.

Definition at line 132 of file scheme.hpp.

5.47.4.7 VECT Scheme::dzi [protected]

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

Definition at line 160 of file scheme.hpp.

5.47.4.8 SCALAR Scheme::elapsed_time [protected]

Duration of the computation.

Definition at line 200 of file scheme.hpp.

5.47.4.9 time_t Scheme::end [protected]

End of timer.

Definition at line 198 of file scheme.hpp.

5.47.4.10 VECT Scheme::f1 [protected]

First component of the numerical flux.

Definition at line 162 of file scheme.hpp.

5.47.4.11 VECT Scheme::f2 [protected]

Second component of the numerical flux.

Definition at line 164 of file scheme.hpp.

5.47.4.12 SCALAR Scheme::flux_l [protected]

Flux on the left.

Definition at line 184 of file scheme.hpp.

5.47.4.13 Choice_flux* Scheme::flux_num [protected]

Definition at line 207 of file scheme.hpp.

5.47.4.14 SCALAR Scheme::flux_r [protected]

Flux on the right.

Definition at line 186 of file scheme.hpp.

5.47.4.15 SCALAR Scheme::Fr [protected]

Mean Froude number.

Definition at line 192 of file scheme.hpp.

5.47.4.16 Choice_friction* Scheme::fric [protected]

Definition at line 208 of file scheme.hpp.

5.47.4.17 VECT Scheme::h [protected]

Water height.

Definition at line 150 of file scheme.hpp.

5.47.4.18 VECT Scheme::hl [protected]

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

Definition at line 172 of file scheme.hpp.

5.47.4.19 VECT Scheme::hleft [protected]

Hydrostatic reconstruction on the left.

Definition at line 180 of file scheme.hpp.

5.47.4.20 VECT Scheme::hr [protected]

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

Definition at line 174 of file scheme.hpp.

5.47.4.21 VECT Scheme::hright [protected]

Hydrostatic reconstruction on the right.

Definition at line 182 of file scheme.hpp.

5.47.4.22 VECT Scheme::hs [protected]

Water height after one step of the scheme.

Definition at line 166 of file scheme.hpp.

5.47.4.23 Choice_init_hu* Scheme::hu_init [protected]

Definition at line 210 of file scheme.hpp.

5.47.4.24 const SCALAR Scheme::L_IMP_H [protected]

Imposed water height on the left boundary.

Definition at line 142 of file scheme.hpp.

5.47.4.25 const SCALAR Scheme::L_IMP_Q [protected]

Imposed discharge on the left boundary.

Definition at line 140 of file scheme.hpp.

5.47.4.26 Choice_condition* Scheme::Lbound [protected]

Definition at line 205 of file scheme.hpp.

5.47.4.27 const int Scheme::M [protected]

Number of time steps.

Definition at line 130 of file scheme.hpp.

5.47.4.28 int Scheme::nbt [protected]

Number of time steps between two savings.

Definition at line 148 of file scheme.hpp.

5.47.4.29 const int Scheme::NBTIMES [protected]

Number of times saved.

Definition at line 136 of file scheme.hpp.

5.47.4.30 const int Scheme::NXCELL [protected]

Number of cells in space.

Definition at line 128 of file scheme.hpp.

5.47.4.31 Choice_output* Scheme::out [protected]

Definition at line 211 of file scheme.hpp.

5.47.4.32 VECT Scheme::q [protected]

Discharge.

Definition at line 154 of file scheme.hpp.

5.47.4.33 VECT Scheme::qs [protected]

Discharge after one step of the scheme.

Definition at line 170 of file scheme.hpp.

5.47.4.34 const SCALAR Scheme::R_IMP_H [protected]

Imposed water height on the right boundary.

Definition at line 146 of file scheme.hpp.

5.47.4.35 const SCALAR Scheme::R_IMP_Q [protected]

Imposed discharge on the right boundary.

Definition at line 144 of file scheme.hpp.

5.47.4.36 Choice_condition* Scheme::Rbound [protected]

Definition at line 206 of file scheme.hpp.

5.47.4.37 Hydrostatic Scheme::rec_hydro [protected]

Definition at line 204 of file scheme.hpp.

5.47.4.38 time_t Scheme::start [protected]

Beginning of timer.

Definition at line 196 of file scheme.hpp.

5.47.4.39 int Scheme::time_initial [protected]

Initial time: the initial time is time_initial * dt.

Definition at line 194 of file scheme.hpp.

5.47.4.40 Choice_init_topo* Scheme::topo [protected]

Definition at line 209 of file scheme.hpp.

5.47.4.41 const SCALAR Scheme::TX [protected]

Ratio dt/dx.

Definition at line 134 of file scheme.hpp.

5.47.4.42 VECT Scheme::u [protected]

Velocity.

Definition at line 152 of file scheme.hpp.

5.47.4.43 VECT Scheme::ul [protected]

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

Definition at line 176 of file scheme.hpp.

5.47.4.44 VECT Scheme::ur [protected]

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

Definition at line 178 of file scheme.hpp.

5.47.4.45 VECT Scheme::us [protected]

Velocity after one step of the scheme.

Definition at line 168 of file scheme.hpp.

5.47.4.46 VECT Scheme::z [protected]

Topography.

Definition at line 156 of file scheme.hpp.

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

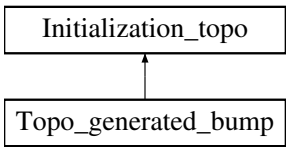
- Headers/libschemes/scheme.hpp
- Sources/libschemes/scheme.cpp

5.48 Topo_generated_bump Class Reference

Bump configuration.

`#include <topo_generated_bump.hpp>`

Inheritance diagram for Topo_generated_bump:



Public Member Functions

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

Additional Inherited Members

5.48.1 Detailed Description

Bump configuration.

Class that initializes a bump topography.

Definition at line 69 of file topo_generated_bump.hpp.

5.48.2 Constructor & Destructor Documentation

5.48.2.1 Topo_generated_bump::Topo_generated_bump (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 58 of file topo_generated_bump.cpp.

5.48.2.2 Topo_generated_bump::~Topo_generated_bump ( ) [virtual]

Destructor.

Definition at line 65 of file topo_generated_bump.cpp.

5.48.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

in,out	*z*	topography.

Implements Initialization_topo.

Definition at line 68 of file topo_generated_bump.cpp.

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

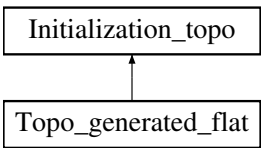
- Headers/libinitializations/topo_generated_bump.hpp
- Sources/libinitializations/topo_generated_bump.cpp

5.49 Topo_generated_flat Class Reference

Flat configuration.

`#include <topo_generated_flat.hpp>`

Inheritance diagram for Topo_generated_flat:



Public Member Functions

- Topo_generated_flat (Parameters &)

  Constructor.
- void initialization (VECT &)

  Performs the initialization.
- virtual ~Topo_generated_flat ()

  Destructor.

Additional Inherited Members

5.49.1 Detailed Description

Flat configuration.

Class that initializes a flat topography, with value 0.

Definition at line 70 of file topo_generated_flat.hpp.

5.49.2 Constructor & Destructor Documentation

5.49.2.1 Topo_generated_flat::Topo_generated_flat (Parameters & *par*)

Constructor.

Parameters

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

Definition at line 59 of file topo_generated_flat.cpp.

5.49.2.2 Topo_generated_flat::~Topo_generated_flat ( ) [virtual]

Destructor.

Definition at line 67 of file topo_generated_flat.cpp.

5.49.3 Member Function Documentation

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

Performs the initialization.

Initializes the topography to 0.

Parameters

in,out	*z*	topography.

Implements Initialization_topo.

Definition at line 70 of file topo_generated_flat.cpp.

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

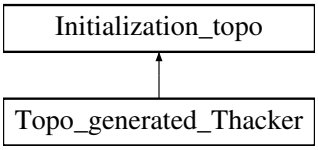
- Headers/libinitializations/topo_generated_flat.hpp
- Sources/libinitializations/topo_generated_flat.cpp

5.50 Topo_generated_Thacker Class Reference

Thacker configuration.

`#include <topo_generated_thacker.hpp>`

Inheritance diagram for Topo_generated_Thacker:



Public Member Functions

- Topo_generated_Thacker (Parameters &)
  Constructor.
- void initialization (VECT &)
  Performs the initialization.
- virtual ~Topo_generated_Thacker ()
  Destructor.

Additional Inherited Members

5.50.1 Detailed Description

Thacker configuration.

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

Definition at line 70 of file topo_generated_thacker.hpp.

5.50.2 Constructor & Destructor Documentation

5.50.2.1 Topo_generated_Thacker::Topo_generated_Thacker (Parameters & *par*)

Constructor.

Defines the parameter h0 and a of the parabola.

Parameters

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

Definition at line 58 of file topo_generated_thacker.cpp.

5.50.2.2 Topo_generated_Thacker::~Topo_generated_Thacker ( ) [virtual]

Destructor.

Definition at line 70 of file topo_generated_thacker.cpp.

5.50.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

in,out	*z*	topography.

Implements Initialization_topo.

Definition at line 73 of file topo_generated_thacker.cpp.

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

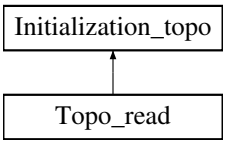
- Headers/libinitializations/topo_generated_thacker.hpp
- Sources/libinitializations/topo_generated_thacker.cpp

5.51 Topo_read Class Reference

File configuration.

`#include <topo_read.hpp>`

Inheritance diagram for Topo_read:



Public Member Functions

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

Additional Inherited Members

5.51.1 Detailed Description

File configuration.

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

Definition at line 70 of file topo_read.hpp.

5.51.2 Constructor & Destructor Documentation

5.51.2.1 Topo_read::Topo_read (Parameters & *par*)

Constructor.

Defines the name of the file for the initialization.

Parameters

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

Definition at line 59 of file topo_read.cpp.

5.51.2.2 Topo_read::~Topo_read ( ) [virtual]

Destructor.

Definition at line 70 of file topo_read.cpp.

5.51.3 Member Function Documentation

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

Performs the initialization.

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

Parameters

in,out	*z*	topography.

Warning

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

Note

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

Todo Exception should be treated.

Implements Initialization_topo.

Definition at line 74 of file topo_read.cpp.

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

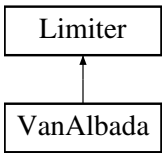
- Headers/libinitializations/topo_read.hpp
- Sources/libinitializations/topo_read.cpp

5.52 VanAlbada Class Reference

Van Albada slope limiter.

`#include <vanalbada.hpp>`

Inheritance diagram for VanAlbada:



Public Member Functions

- VanAlbada ()

  Constructor.
- void calc (SCALAR, SCALAR)

  Calculates the value of the slope limiter.
- virtual ~VanAlbada ()

  Destructor.

Additional Inherited Members

5.52.1 Detailed Description

Van Albada slope limiter.

Class that calculates Van Albada slope limiter.

Definition at line 69 of file vanalbada.hpp.

5.52.2 Constructor & Destructor Documentation

5.52.2.1 VanAlbada::VanAlbada ( )

Constructor.

Definition at line 58 of file vanalbada.cpp.

5.52.2.2 VanAlbada::~VanAlbada ( ) [virtual]

Destructor.

Definition at line 83 of file vanalbada.cpp.

5.52.3 Member Function Documentation

5.52.3.1 void VanAlbada::calc (SCALAR *a*, SCALAR *b*) [virtual]

Calculates the value of the slope limiter.

Van Albada function:

$$\text{VA}(x,y) = \begin{cases} 0 & \text{if } \text{sign}(x) \neq \text{sign}(y), \\ \dfrac{x(y^2+\varepsilon)+y(x^2+\varepsilon)}{x^2+y^2+2\varepsilon} & \text{else,} \end{cases}$$

with $0 \leq \varepsilon \ll 1$.

Parameters

in	*a*	slope on the left of the cell.
in	*b*	slope on the right of the cell.

Modifies

Limiter::rec recontructed value.

Implements Limiter.

Definition at line 61 of file vanalbada.cpp.

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

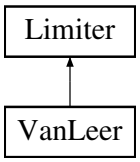
- Headers/liblimitations/vanalbada.hpp
- Sources/liblimitations/vanalbada.cpp

5.53 VanLeer Class Reference

Van Leer slope limiter.

`#include <vanleer.hpp>`

Inheritance diagram for VanLeer:



Public Member Functions

- VanLeer ()
  Constructor.
- void calc (SCALAR, SCALAR)

Calculates the value of the slope limiter.

- virtual ~VanLeer ()

  Destructor.

Additional Inherited Members

5.53.1 Detailed Description

Van Leer slope limiter.

Class that calculates Van Leer slope limiter.

Definition at line 69 of file vanleer.hpp.

5.53.2 Constructor & Destructor Documentation

5.53.2.1 VanLeer::VanLeer ( )

Constructor.

Definition at line 59 of file vanleer.cpp.

5.53.2.2 VanLeer::~VanLeer ( ) [virtual]

Destructor.

Definition at line 83 of file vanleer.cpp.

5.53.3 Member Function Documentation

5.53.3.1 void VanLeer::calc (SCALAR *a*, SCALAR *b*) [virtual]

Calculates the value of the slope limiter.

Van Leer function:

$$\mathrm{VL}(x,y) = \begin{cases} 0 & \text{if } xy \leq 0, \\ \dfrac{2xy}{x+y} & \text{else.} \end{cases}$$

Parameters

in	*a*	slope on the left of the cell.
in	*b*	slope on the right of the cell.

Modifies

Limiter::rec recontructed value.

Implements Limiter.

Definition at line 62 of file vanleer.cpp.

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

- Headers/liblimitations/vanleer.hpp

- Sources/liblimitations/vanleer.cpp

Chapter 6

File Documentation

6.1 Headers/libboundaryconditions/bc_imp_discharge.hpp File Reference

Imposed discharge.

```
#include "boundary_condition.hpp"
```

Classes

- class Bc_imp_discharge

  Imposed discharge.

Macros

- #define BC_IMP_DISCHARGE_HPP

6.1.1 Detailed Description

Imposed discharge.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file bc_imp_discharge.hpp.

6.1.2 Macro Definition Documentation

6.1.2.1 #define BC_IMP_DISCHARGE_HPP

Definition at line 63 of file bc_imp_discharge.hpp.

6.2 Headers/libboundaryconditions/bc_imp_height.hpp File Reference

Imposed water height.

```
#include "boundary_condition.hpp"
```

Classes

- class Bc_imp_height

  Imposed water height.

6.2.1 Detailed Description

Imposed water height.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: imposed water height (and discharge if necessary), based on the modified method of characteristics and Riemann invariants.

See Also

Olivier Delestre Ph.D thesis Annexe A Delestre [2010] http://tel.archives-ouvertes.fr/tel-00587197

Copyright

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

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

Definition in file bc_imp_height.hpp.

6.3 Headers/libboundaryconditions/bc_neumann.hpp File Reference

Neumann condition.

```
#include "boundary_condition.hpp"
```

Classes

- class Bc_neumann
  Neumann condition.

6.3.1 Detailed Description

Neumann condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file bc_neumann.hpp.

6.4 Headers/libboundaryconditions/bc_periodic.hpp File Reference

Periodic condition.

```
#include "boundary_condition.hpp"
```

Classes

- class Bc_periodic

  Periodic condition.

6.4.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: periodic condition.

Copyright

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

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

Definition in file bc_periodic.hpp.

6.5 Headers/libboundaryconditions/bc_wall.hpp File Reference

Wall condition.

```
#include "boundary_condition.hpp"
```

Classes

- class Bc_wall

  Wall condition.

6.5.1 Detailed Description

Wall condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: wall condition (the discharge at the boundary is null).

Copyright

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

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

Definition in file bc_wall.hpp.

6.6 Headers/libboundaryconditions/boundary_condition.hpp File Reference

Boundary condition.

```
#include "parameters.hpp"
```

Classes

- class Boundary_condition

  Boundary condition.

6.6.1 Detailed Description

Boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Common part for all the boundary conditions.

Copyright

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

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

Definition in file boundary_condition.hpp.

6.7 Headers/libboundaryconditions/choice_condition.hpp File Reference

Choice of boundary condition.

```
#include "boundary_condition.hpp"
#include "bc_neumann.hpp"
#include "bc_imp_height.hpp"
#include "bc_imp_discharge.hpp"
#include "bc_wall.hpp"
#include "bc_periodic.hpp"
```

Classes

- class Choice_condition

  Choice of boundary condition.

Macros

- #define CHOICE_CONDITION_HPP

6.7.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen boundary condition.

Copyright

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

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

Definition in file choice_condition.hpp.

6.7.2 Macro Definition Documentation

6.7.2.1 #define CHOICE_CONDITION_HPP

Definition at line 82 of file choice_condition.hpp.

6.8 Headers/libflux/choice_flux.hpp File Reference

Choice of numerical flux.

```
#include "flux.hpp"
#include "f_rusanov.hpp"
#include "f_hll.hpp"
#include "f_hll2.hpp"
#include "f_kinetic.hpp"
#include "f_vfroe.hpp"
```

Classes

- class Choice_flux

  Choice of numerical flux.

Macros

- #define CHOICE_FLUX_HPP

6.8.1 Detailed Description

Choice of numerical flux.

Author

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

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen numerical flux.

Copyright

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

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

Definition in file choice_flux.hpp.

6.8.2 Macro Definition Documentation

6.8.2.1 #define CHOICE_FLUX_HPP

Definition at line 83 of file choice_flux.hpp.

6.9 Headers/libflux/f_hll.hpp File Reference

HLL flux.

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

Classes

- class F_HLL

  HLL flux.

6.9.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical flux: Harten, Lax, van Leer formulation.

Copyright

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

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

Definition in file f_hll.hpp.

6.10 Headers/libflux/f_hll2.hpp File Reference

HLL flux.

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

Classes

- class F_HLL2
  HLL flux.

6.10.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file f_hll2.hpp.

6.11 Headers/libflux/f_kinetic.hpp File Reference

Kinetic flux.

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

Classes

- class F_Kinetic
  Kinetic flux.

6.11.1 Detailed Description

Kinetic flux.

Author

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

Version

1.00.00

Date

2012-04-25

Numerical flux: kinetic formulation.

Copyright

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

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

Definition in file f_kinetic.hpp.

6.12 Headers/libflux/f_rusanov.hpp File Reference

Rusanov flux.

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

Classes

- class F_Rusanov
  Rusanov flux.

6.12.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical flux: Rusanov formulation.

Copyright

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

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

Definition in file f_rusanov.hpp.

6.13 Headers/libflux/f_vfroe.hpp File Reference

VFRoe flux.

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

Classes

- class F_VFRoe
  VFRoe flux.

6.13.1 Detailed Description

VFRoe flux.

Author

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

Version

1.00.00

Date

2012-04-25

Numerical flux: VFRoe-ncv formulation.

Copyright

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

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

Definition in file f_vfroe.hpp.

6.14 Headers/libflux/flux.hpp File Reference

Numerical flux.

```
#include "parameters.hpp"
```

Classes

- class Flux

  Numerical flux.

6.14.1 Detailed Description

Numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the numerical fluxes.

Copyright

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

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

Definition in file flux.hpp.

6.15 Headers/libfrictions/choice_friction.hpp File Reference

Choice of friction law.

```
#include "friction.hpp"
#include "no_friction.hpp"
#include "fr_manning.hpp"
#include "fr_darcy_weisbach.hpp"
```

Classes

- class Choice_friction

  Choice of friction law.

Macros

- #define CHOICE_FRICTION_HPP

6.15.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

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

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

Definition in file choice_friction.hpp.

6.15.2 Macro Definition Documentation

6.15.2.1 #define CHOICE_FRICTION_HPP

Definition at line 74 of file choice_friction.hpp.

6.16 Headers/libfrictions/fr_darcy_weisbach.hpp File Reference

Darcy-Weisbach law.

```
#include "friction.hpp"
```

Classes

- class Fr_Darcy_Weisbach
  Darcy-Weisbach law.

6.16.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Friction law: Darcy-Weisbach.

Copyright

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

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

Definition in file fr_darcy_weisbach.hpp.

6.17 Headers/libfrictions/fr_manning.hpp File Reference

Manning law.

```
#include "friction.hpp"
```

Classes

- class Fr_Manning
  Manning law.

6.17.1 Detailed Description

Manning law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Friction law: Manning.

Copyright

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

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

Definition in file fr_manning.hpp.

6.18 Headers/libfrictions/friction.hpp File Reference

Friction law

```
#include "parameters.hpp"
```

Classes

- class Friction
  Friction law

6.18.1 Detailed Description

Friction law

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the friction laws.

Copyright

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

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

Definition in file friction.hpp.

6.19 Headers/libfrictions/no_friction.hpp File Reference

No friction.

```
#include "friction.hpp"
```

Classes

- class No_Friction

  No friction.

6.19.1 Detailed Description

No friction.

Author

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

Version

1.00.00

Date

2012-04-25

Friction law: does no computation.

Copyright

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

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

Definition in file no_friction.hpp.

6.20 Headers/libinitializations/choice_init_hu.hpp File Reference

Choice of initialization for h and u.

```
#include "initialization_hu.hpp"
#include "hu_read.hpp"
#include "hu_generated.hpp"
#include "hu_generated_wet_dam_break.hpp"
#include "hu_generated_dry_dam_break.hpp"
#include "hu_generated_dressler_dam_break.hpp"
#include "hu_generated_thacker.hpp"
```

Classes

- class Choice_init_hu

  Choice of initialization for h and u.

Macros

- #define CHOICE_INIT_HU_HPP

6.20.1 Detailed Description

Choice of initialization for h and u.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen initialization of the water height and of the velocity.

Copyright

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

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

Definition in file choice_init_hu.hpp.

6.20.2 Macro Definition Documentation

6.20.2.1 #define CHOICE_INIT_HU_HPP

Definition at line 87 of file choice_init_hu.hpp.

6.21 Headers/libinitializations/choice_init_topo.hpp File Reference

Choice of initialization for the topography.

```
#include "initialization_topo.hpp"
#include "topo_read.hpp"
#include "topo_generated_flat.hpp"
#include "topo_generated_thacker.hpp"
#include "topo_generated_bump.hpp"
```

Classes

- class Choice_init_topo
  Choice of initialization for the topography.

Macros

- #define CHOICE_INIT_TOPO_HPP

6.21.1 Detailed Description

Choice of initialization for the topography.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

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

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

Definition in file choice_init_topo.hpp.

6.21.2 Macro Definition Documentation

6.21.2.1 #define CHOICE_INIT_TOPO_HPP

Definition at line 78 of file choice_init_topo.hpp.

6.22 Headers/libinitializations/hu_generated.hpp File Reference

No water configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_generated

  No water configuration.

6.22.1 Detailed Description

No water configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a wet domain.

Copyright

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

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

Definition in file hu_generated.hpp.

6.23 Headers/libinitializations/hu_generated_dressler_dam_break.hpp File Reference

Dressler dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_generated_Dressler_Dam_break
  Dressler dam break configuration.

6.23.1 Detailed Description

Dressler dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break as studied by Dressler (with friction).

Copyright

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

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

Definition in file hu_generated_dressler_dam_break.hpp.

6.24 Headers/libinitializations/hu_generated_dry_dam_break.hpp File Reference

Dry dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_generated_Dry_Dam_break
  Dry dam break configuration.

6.24.1 Detailed Description

Dry dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break on a dry domain.

Copyright

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

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

Definition in file hu_generated_dry_dam_break.hpp.

6.25 Headers/libinitializations/hu_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_generated_Thacker
  Thacker configuration.

6.25.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of Thacker's benchmark.

Copyright

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

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

Definition in file hu_generated_thacker.hpp.

6.26 Headers/libinitializations/hu_generated_wet_dam_break.hpp File Reference

Wet dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_generated_Wet_Dam_break

  Wet dam break configuration.

6.26.1 Detailed Description

Wet dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break on a wet domain.

Copyright

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

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

Definition in file hu_generated_wet_dam_break.hpp.

6.27 Headers/libinitializations/hu_read.hpp File Reference

File configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class Hu_read

  File configuration.

6.27.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: the values are read in a file.

Copyright

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

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

Definition in file hu_read.hpp.

6.28 Headers/libinitializations/initialization_hu.hpp File Reference

Initialization of h and u

```
#include "parameters.hpp"
```

Classes

- class Initialization_hu
  Initialization of h and u.

6.28.1 Detailed Description

Initialization of h and u

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the initialization of the water height and of the velocity.

Copyright

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

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

Definition in file initialization_hu.hpp.

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

Version

1.00.00

Date

2012-04-25

Common part for all the initialization of the topography.

Copyright

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

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

Definition in file initialization_topo.hpp.

6.30 Headers/libinitializations/topo_generated_bump.hpp File Reference

Bump configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class Topo_generated_bump

  Bump configuration.

6.30.1 Detailed Description

Bump configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: the topography is a bump.

Copyright

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

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

Definition in file topo_generated_bump.hpp.

6.31 Headers/libinitializations/topo_generated_flat.hpp File Reference

Flat configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class Topo_generated_flat

  Flat configuration.

6.31.1 Detailed Description

Flat configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: the topography is flat, its value is 0.

Copyright

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

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

Definition in file topo_generated_flat.hpp.

6.32 Headers/libinitializations/topo_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class Topo_generated_Thacker
  Thacker configuration.

6.32.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: parabolic topography for Thacker's Benchmark.

Copyright

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

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

Definition in file topo_generated_thacker.hpp.

6.33 Headers/libinitializations/topo_read.hpp File Reference

File configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class Topo_read
  File configuration.

6.33.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: the values are read in a file.

Copyright

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

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

Definition in file topo_read.hpp.

6.34 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

6.34.1 Detailed Description

Choice of slope limiter.

Author

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

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

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

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

Definition in file choice_limiter.hpp.

6.34.2 Macro Definition Documentation

6.34.2.1 #define CHOICE_LIMITER_HPP

Definition at line 75 of file choice_limiter.hpp.

6.35 Headers/liblimitations/limiter.hpp File Reference

Slope limiter.

```
#include "parameters.hpp"
```

Classes

- class Limiter

  Slope limiter.

6.35.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the slope limiters.

Copyright

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

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

Definition in file limiter.hpp.

6.36 Headers/liblimitations/minmod.hpp File Reference

Minmod limiter

```
#include "limiter.hpp"
```

Classes

- class Minmod

  Minmod slope limiter

6.36.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: minmod.

Copyright

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

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

Definition in file minmod.hpp.

6.37 Headers/liblimitations/vanalbada.hpp File Reference

Van Albada limiter.

```
#include "limiter.hpp"
```

Classes

- class VanAlbada
  Van Albada slope limiter.

6.37.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: Van Albada.

Copyright

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

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

Definition in file vanalbada.hpp.

6.38 Headers/liblimitations/vanleer.hpp File Reference

Van Leer limiter.

```
#include "limiter.hpp"
```

Classes

- class VanLeer

  Van Leer slope limiter.

6.38.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: Van Leer.

Copyright

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

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

Definition in file vanleer.hpp.

6.39 Headers/libparameters/misc.hpp File Reference

Definitions.

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <stdio.h>
#include <fstream>
#include <cmath>
#include <cstring>
#include <cstdio>
#include <iomanip>
#include <stdlib.h>
#include <vector>
#include <unistd.h>
#include <time.h>
#include <sstream>
#include <cfloat>
```

Macros

- #define MAX(a, b) (a>=b?a:b)
- #define MIN(a, b) (a<=b?a:b)
- #define GRAV 9.81
- #define GRAV_DEM 4.905
- #define NB_CHAR 256
- #define ZERO 0.
- #define HE_CA 1.e-10
- #define VE_CA 1.e-10
- #define EPSILON 1.e-13
- #define RATIO_CLOSE_CELL 1.e-3
- #define VERSION "FullSWOF_1D version 1.00.02, 2013-05-12"
- #define MAX_SCAL DBL_MAX

Typedefs

- typedef double SCALAR
- typedef vector< SCALAR > VECT

6.39.1 Detailed Description

Definitions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2013)

Version

1.00.02

Date

2013-05-12

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)

Definition in file misc.hpp.

6.39.2 Macro Definition Documentation

6.39.2.1 #define EPSILON 1.e-13

Definition at line 82 of file misc.hpp.

6.39.2.2 #define GRAV 9.81

Definition at line 75 of file misc.hpp.

6.39.2.3 #define GRAV_DEM 4.905

Definition at line 76 of file misc.hpp.

6.39.2.4 #define HE_CA 1.e-10

Definition at line 80 of file misc.hpp.

6.39.2.5 #define MAX(*a*, *b*) (a>=b?a:b)

Definition at line 72 of file misc.hpp.

6.39.2.6 #define MAX_SCAL DBL_MAX

Definition at line 93 of file misc.hpp.

6.39.2.7 #define MIN(*a*, *b*) (a<=b?a:b)

Definition at line 73 of file misc.hpp.

6.39.2.8 #define NB_CHAR 256

Definition at line 78 of file misc.hpp.

6.39.2.9 #define RATIO_CLOSE_CELL 1.e-3

Definition at line 84 of file misc.hpp.

6.39.2.10 #define VE_CA 1.e-10

Definition at line 81 of file misc.hpp.

6.39.2.11 #define VERSION "FullSWOF_1D version 1.00.02, 2013-05-12"

Definition at line 86 of file misc.hpp.

6.39.2.12 #define ZERO 0.

Definition at line 79 of file misc.hpp.

6.39.3 Typedef Documentation

6.39.3.1 typedef double SCALAR

Definition at line 90 of file misc.hpp.

6.39.3.2 typedef vector<SCALAR> VECT

Definition at line 95 of file misc.hpp.

6.40 Headers/libparameters/parameters.hpp File Reference

Gets parameters.

```
#include "misc.hpp"
```

Classes

- class Parameters

  Gets parameters.

6.40.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.00.02

Date

2013-05-11

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 - INRA (France)

Definition in file parameters.hpp.

6.41 Headers/libparser/parser.hpp File Reference

Parser

```
#include "misc.hpp"
```

Classes

- class Parser

  Parser to read the entries

6.41.1 Detailed Description

Parser

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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)

Definition in file parser.hpp.

6.42 Headers/libreconstructions/choice_reconstruction.hpp File Reference

Choice of reconstruction.

```
#include "reconstruction.hpp"
#include "muscl.hpp"
#include "eno.hpp"
#include "eno_mod.hpp"
```

Classes

- class Choice_reconstruction
  Choice of reconstruction.

Macros

- #define CHOICE_RECONSTRUCTION_HPP

6.42.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen reconstruction.

Copyright

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

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

Definition in file choice_reconstruction.hpp.

6.42.2 Macro Definition Documentation

6.42.2.1 #define CHOICE_RECONSTRUCTION_HPP

Definition at line 74 of file choice_reconstruction.hpp.

6.43 Headers/libreconstructions/eno.hpp File Reference

ENO reconstruction

```
#include "reconstruction.hpp"
```

Classes

- class ENO

  ENO recontruction

6.43.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: ENO.

Copyright

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

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

Definition in file eno.hpp.

6.44 Headers/libreconstructions/eno_mod.hpp File Reference

Modified ENO reconstruction.

```
#include "reconstruction.hpp"
```

Classes

- class ENO_mod

  Modified ENO recontruction.

6.44.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: modified ENO.

Copyright

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

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

Definition in file eno_mod.hpp.

6.45 Headers/libreconstructions/hydrostatic.hpp File Reference

Hydrostatic reconstruction

```
#include "parameters.hpp"
```

Classes

- class Hydrostatic

  Hydrostatic reconstruction

6.45.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Copyright

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

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

Definition in file hydrostatic.hpp.

6.46 Headers/libreconstructions/muscl.hpp File Reference

MUSCL reconstruction

```
#include "reconstruction.hpp"
```

Classes

- class MUSCL
  MUSCL recontruction

6.46.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: MUSCL.

Copyright

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

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

Definition in file muscl.hpp.

6.47 Headers/libreconstructions/reconstruction.hpp File Reference

Reconstruction

```
#include "parameters.hpp"
#include "choice_limiter.hpp"
```

Classes

- class Reconstruction

  Reconstruction of the variables

6.47.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the reconstructions.

Copyright

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

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

Definition in file reconstruction.hpp.

6.48 Headers/libsave/choice_output.hpp File Reference

Choice of output format.

```
#include "output.hpp"
#include "gnuplot.hpp"
```

Classes

- class Choice_output

  Choice of output format.

Macros

- #define CHOICE_OUTPUT_HPP

6.48.1 Detailed Description

Choice of output format.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file choice_output.hpp.

6.48.2 Macro Definition Documentation

6.48.2.1 #define CHOICE_OUTPUT_HPP

Definition at line 65 of file choice_output.hpp.

6.49 Headers/libsave/gnuplot.hpp File Reference

Gnuplot output

```
#include "output.hpp"
```

Classes

- class Gnuplot

  Gnuplot output

6.49.1 Detailed Description

Gnuplot output

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Output format: optimized for Gnuplot.

Copyright

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

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

Definition in file gnuplot.hpp.

6.50 Headers/libsave/output.hpp File Reference

Output format

```
#include "parameters.hpp"
```

Classes

- class Output

  Output format

6.50.1 Detailed Description

Output format

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Common part for all the output formats.

Copyright

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

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

Definition in file output.hpp.

6.51 Headers/libschemes/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

6.51.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen numerical scheme.

Copyright

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

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

Definition in file choice_scheme.hpp.

6.51.2 Macro Definition Documentation

6.51.2.1 #define CHOICE_SCHEME_HPP

Definition at line 70 of file choice_scheme.hpp.

6.52 Headers/libschemes/order1.hpp File Reference

Order 1 scheme.

```
#include "scheme.hpp"
```

Classes

- class Order1
  Order 1 scheme.

6.52.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical scheme: at order 1.

Copyright

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

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

Definition in file order1.hpp.

6.53 Headers/libschemes/order2.hpp File Reference

Order 2 scheme.

```
#include "scheme.hpp"
```

Classes

- class Order2
  Order 2 scheme.

6.53.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical scheme: at order 2.

Copyright

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

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

Definition in file order2.hpp.

6.54 Headers/libschemes/scheme.hpp File Reference

Numerical scheme.

```
#include "hydrostatic.hpp"
#include "choice_condition.hpp"
#include "choice_flux.hpp"
#include "choice_friction.hpp"
#include "choice_init_topo.hpp"
#include "choice_init_hu.hpp"
#include "choice_output.hpp"
#include "choice_reconstruction.hpp"
```

Classes

- class Scheme

  Numerical scheme.

6.54.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the numerical schemes.

Copyright

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

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

Definition in file scheme.hpp.

6.55 Sources/FullSWOF_1D.cpp File Reference

Main function.

```
#include "choice_scheme.hpp"
```

Functions

- int main ()

6.55.1 Detailed Description

Main function.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Runs the programm.

Copyright

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

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

Definition in file FullSWOF_1D.cpp.

6.55.2 Function Documentation

6.55.2.1 int main ( )

Main function

Declare the scheme and executes the program.

Returns

0 if the program finished correctly.

Note

The name of the input file (Inputs/parameters.txt) is written here.

Definition at line 57 of file FullSWOF_1D.cpp.

6.56 Sources/libboundaryconditions/bc_imp_discharge.cpp File Reference

Imposed discharge.

```
#include "bc_imp_discharge.hpp"
```

6.56.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison ulrich.razafison@math.cnrs.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.00.01

Date

2012-04-26

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

Copyright

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

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

Definition in file bc_imp_discharge.cpp.

6.57 Sources/libboundaryconditions/bc_imp_height.cpp File Reference

Imposed water height.

```
#include "bc_imp_height.hpp"
```

6.57.1 Detailed Description

Imposed water height.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: imposed water height (and discharge if necessary), based on the modified method of characteristics and Riemann invariants.

See Also

Olivier Delestre Ph.D thesis Annexe A Delestre [2010] http://tel.archives-ouvertes.-fr/tel-00587197

Copyright

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

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

Definition in file bc_imp_height.cpp.

6.58 Sources/libboundaryconditions/bc_neumann.cpp File Reference

Neumann condition.

```
#include "bc_neumann.hpp"
```

6.58.1 Detailed Description

Neumann condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file bc_neumann.cpp.

6.59 Sources/libboundaryconditions/bc_periodic.cpp File Reference

Periodic condition.

```
#include "bc_periodic.hpp"
```

6.59.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: periodic condition.

Copyright

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

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

Definition in file bc_periodic.cpp.

6.60 Sources/libboundaryconditions/bc_wall.cpp File Reference

Wall condition.

```
#include "bc_wall.hpp"
```

6.60.1 Detailed Description

Wall condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Boundary condition: wall condition (the discharge at the boundary is null).

Copyright

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

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

Definition in file bc_wall.cpp.

6.61 Sources/libboundaryconditions/boundary_condition.cpp File Reference

Boundary condition.

```
#include "boundary_condition.hpp"
```

6.61.1 Detailed Description

Boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Common part for all the boundary conditions.

Copyright

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

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

Definition in file boundary_condition.cpp.

6.62 Sources/libboundaryconditions/choice_condition.cpp File Reference

Choice of boundary condition.

```
#include "choice_condition.hpp"
```

6.62.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen boundary condition.

Copyright

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

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

Definition in file choice_condition.cpp.

6.63 Sources/libflux/choice_flux.cpp File Reference

Choice of numerical flux.

```
#include "choice_flux.hpp"
```

6.63.1 Detailed Description

Choice of numerical flux.

Author

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

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen numerical flux.

Copyright

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

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

Definition in file choice_flux.cpp.

6.64 Sources/libflux/f_hll.cpp File Reference

HLL flux.

```
#include "f_hll.hpp"
```

6.64.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical flux: Harten, Lax, van Leer formulation.

Copyright

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

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

Definition in file f_hll.cpp.

6.65 Sources/libflux/f_hll2.cpp File Reference

HLL flux.

```
#include "f_hll2.hpp"
```

6.65.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file f_hll2.cpp.

6.66 Sources/libflux/f_kinetic.cpp File Reference

Kinetic flux.

```
#include "f_kinetic.hpp"
```

6.66.1 Detailed Description

Kinetic flux.

Author

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

Version

1.00.00

Date

2012-04-25

Numerical flux: kinetic formulation.

Copyright

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

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

Definition in file f_kinetic.cpp.

6.67 Sources/libflux/f_rusanov.cpp File Reference

Rusanov flux.

```
#include "f_rusanov.hpp"
```

6.67.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Numerical flux: Rusanov formulation.

Copyright

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

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

Definition in file f_rusanov.cpp.

6.68 Sources/libflux/f_vfroe.cpp File Reference

VFRoe flux.

```
#include "f_vfroe.hpp"
```

6.68.1 Detailed Description

VFRoe flux.

Author

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

Version

1.00.00

Date

2012-04-25

Numerical flux: VFRoe-ncv formulation.

Copyright

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

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

Definition in file f_vfroe.cpp.

6.69 Sources/libflux/flux.cpp File Reference

Numerical flux.

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

6.69.1 Detailed Description

Numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the numerical fluxes.

Copyright

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

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

Definition in file flux.cpp.

6.70 Sources/libfrictions/choice_friction.cpp File Reference

Choice of friction law.

```
#include "choice_friction.hpp"
```

6.70.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

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

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

Definition in file choice_friction.cpp.

6.71 Sources/libfrictions/fr_darcy_weisbach.cpp File Reference

Darcy-Weisbach law.

```
#include "fr_darcy_weisbach.hpp"
```

6.71.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Friction law: Darcy-Weisbach.

Copyright

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

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

Definition in file fr_darcy_weisbach.cpp.

6.72 Sources/libfrictions/fr_manning.cpp File Reference

Manning law.

```
#include "fr_manning.hpp"
```

6.72.1 Detailed Description

Manning law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Friction law: Manning.

Copyright

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

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

Definition in file fr_manning.cpp.

6.73 Sources/libfrictions/friction.cpp File Reference

Friction law

```
#include "friction.hpp"
```

6.73.1 Detailed Description

Friction law

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the friction laws.

Copyright

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

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

Definition in file friction.cpp.

6.74 Sources/libfrictions/no_friction.cpp File Reference

No friction.

```
#include "no_friction.hpp"
```

6.74.1 Detailed Description

No friction.

Author

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

Version

1.00.00

Date

2012-04-25

Friction law: does no computation.

Copyright

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

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

Definition in file no_friction.cpp.

6.75 Sources/libinitializations/choice_init_hu.cpp File Reference

Choice of initialization for h and u.

```
#include "choice_init_hu.hpp"
```

6.75.1 Detailed Description

Choice of initialization for h and u.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen initialization of the water height and of the velocity.

Copyright

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

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

Definition in file choice_init_hu.cpp.

6.76 Sources/libinitializations/choice_init_topo.cpp File Reference

Choice of initialization for the topography.

```
#include "choice_init_topo.hpp"
```

6.76.1 Detailed Description

Choice of initialization for the topography.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

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

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

Definition in file choice_init_topo.cpp.

6.77 Sources/libinitializations/hu_generated.cpp File Reference

No water configuration.

```
#include "hu_generated.hpp"
```

6.77.1 Detailed Description

No water configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a wet domain.

Copyright

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

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

Definition in file hu_generated.cpp.

6.78 Sources/libinitializations/hu_generated_dressler_dam_break.cpp File Reference

Dressler dam break configuration.

```
#include "hu_generated_dressler_dam_break.hpp"
```

6.78.1 Detailed Description

Dressler dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break as studied by Dressler (with friction).

Copyright

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

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

Definition in file hu_generated_dressler_dam_break.cpp.

6.79 Sources/libinitializations/hu_generated_dry_dam_break.cpp File Reference

Dry dam break configuration.

```
#include "hu_generated_dry_dam_break.hpp"
```

6.79.1 Detailed Description

Dry dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break on a dry domain.

Copyright

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

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

Definition in file hu_generated_dry_dam_break.cpp.

6.80 Sources/libinitializations/hu_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "hu_generated_thacker.hpp"
```

6.80.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of Thacker's benchmark.

Copyright

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

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

Definition in file hu_generated_thacker.cpp.

6.81 Sources/libinitializations/hu_generated_wet_dam_break.cpp File Reference

Wet dam break configuration.

```
#include "hu_generated_wet_dam_break.hpp"
```

6.81.1 Detailed Description

Wet dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the water height and of the velocity: case of a dam break on a wet domain.

Copyright

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

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

Definition in file hu_generated_wet_dam_break.cpp.

6.82 Sources/libinitializations/hu_read.cpp File Reference

File configuration.

```
#include "hu_read.hpp"
```

6.82.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2013)

Version

1.00.02

Date

2013-05-12

Initialization of the water height and of the velocity: the values are read in a file.

Copyright

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

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

Definition in file hu_read.cpp.

6.83 Sources/libinitializations/initialization_hu.cpp File Reference

Initialization of h and u

```
#include "initialization_hu.hpp"
```

6.83.1 Detailed Description

Initialization of h and u

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the initialization of the water height and of the velocity.

Copyright

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

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

Definition in file initialization_hu.cpp.

6.84 Sources/libinitializations/initialization_topo.cpp File Reference

Initialization of z

```
#include "initialization_topo.hpp"
```

6.84.1 Detailed Description

Initialization of z

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the initialization of the topography.

Copyright

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

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

Definition in file initialization_topo.cpp.

6.85 Sources/libinitializations/topo_generated_bump.cpp File Reference

Bump configuration.

```
#include "topo_generated_bump.hpp"
```

6.85.1 Detailed Description

Bump configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: the topography is a bump.

Copyright

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

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

Definition in file topo_generated_bump.cpp.

6.86 Sources/libinitializations/topo_generated_flat.cpp File Reference

Flat configuration.

```
#include "topo_generated_flat.hpp"
```

6.86.1 Detailed Description

Flat configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Initialization of the topography: the topography is flat, its value is 0.

Copyright

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

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

Definition in file topo_generated_flat.cpp.

6.87 Sources/libinitializations/topo_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "topo_generated_thacker.hpp"
```

6.87.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.02

Date

2013-05-11

Initialization of the topography: parabolic topography for Thacker's Benchmark.

Copyright

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

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

Definition in file topo_generated_thacker.cpp.

6.88 Sources/libinitializations/topo_read.cpp File Reference

File configuration.

```
#include "topo_read.hpp"
```

6.88.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2013)

Version

1.00.02

Date

2013-05-12

Initialization of the topography: the values are read in a file.

Copyright

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

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

Definition in file topo_read.cpp.

6.89 Sources/liblimitations/choice_limiter.cpp File Reference

Choice of slope limiter.

```
#include "choice_limiter.hpp"
```

6.89.1 Detailed Description

Choice of slope limiter.

Author

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

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

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

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

Definition in file choice_limiter.cpp.

6.90 Sources/liblimitations/limiter.cpp File Reference

Slope limiter.

```
#include "limiter.hpp"
```

6.90.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the slope limiters.

Copyright

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

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

Definition in file limiter.cpp.

6.91 Sources/liblimitations/minmod.cpp File Reference

Minmod limiter

```
#include "minmod.hpp"
```

6.91.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: minmod.

Copyright

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

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

Definition in file minmod.cpp.

6.92 Sources/liblimitations/vanalbada.cpp File Reference

Van Albada limiter.

```
#include "vanalbada.hpp"
```

6.92.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: Van Albada.

Copyright

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

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

Definition in file vanalbada.cpp.

6.93 Sources/liblimitations/vanleer.cpp File Reference

Van Leer limiter.

```
#include "vanleer.hpp"
```

6.93.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Slope limiter: Van Leer.

Copyright

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

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

Definition in file vanleer.cpp.

6.94 Sources/libparameters/parameters.cpp File Reference

Gets parameters.

```
#include "parser.hpp"
#include "parameters.hpp"
```

6.94.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.00.00

Date

2012-04-25

Reads the parameters, checks their values.

Copyright

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

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

Definition in file parameters.cpp.

6.95 Sources/libparser/parser.cpp File Reference

Parser

```
#include "parser.hpp"
```

6.95.1 Detailed Description

Parser

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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)

Definition in file parser.cpp.

6.96 Sources/libreconstructions/choice_reconstruction.cpp File Reference

Choice of reconstruction.

```
#include "choice_reconstruction.hpp"
```

6.96.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen reconstruction.

Copyright

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

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

Definition in file choice_reconstruction.cpp.

6.97 Sources/libreconstructions/eno.cpp File Reference

ENO reconstruction

```
#include "eno.hpp"
```

6.97.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: ENO.

Copyright

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

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

Definition in file eno.cpp.

6.98 Sources/libreconstructions/eno_mod.cpp File Reference

Modified ENO reconstruction.

```
#include "eno_mod.hpp"
```

6.98.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: modified ENO.

Copyright

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

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

Definition in file eno_mod.cpp.

6.99 Sources/libreconstructions/hydrostatic.cpp File Reference

Hydrostatic reconstruction

```
#include "hydrostatic.hpp"
```

6.99.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Copyright

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

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

Definition in file hydrostatic.cpp.

6.100 Sources/libreconstructions/muscl.cpp File Reference

MUSCL reconstruction

```
#include "muscl.hpp"
```

6.100.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Linear reconstruction: MUSCL.

Copyright

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

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

Definition in file muscl.cpp.

6.101 Sources/libreconstructions/reconstruction.cpp File Reference

Reconstruction

```
#include "reconstruction.hpp"
```

6.101.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

Common part for all the reconstructions.

Copyright

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

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

Definition in file reconstruction.cpp.

6.102 Sources/libsave/choice_output.cpp File Reference

Choice of output format.

```
#include "choice_output.hpp"
```

6.102.1 Detailed Description

Choice of output format.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

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

Copyright

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

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

Definition in file choice_output.cpp.

6.103 Sources/libsave/gnuplot.cpp File Reference

Gnuplot output

```
#include "gnuplot.hpp"
```

6.103.1 Detailed Description

Gnuplot output

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Output format: optimized for Gnuplot (for huz_evolution.dat).

Copyright

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

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

Definition in file gnuplot.cpp.

6.104 Sources/libsave/output.cpp File Reference

Output format

```
#include "output.hpp"
```

6.104.1 Detailed Description

Output format

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.00.00

Date

2012-04-25

Common part for all the output formats.

Copyright

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

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

Definition in file output.cpp.

6.105 Sources/libschemes/choice_scheme.cpp File Reference

Choice of numerical scheme.

```
#include "choice_scheme.hpp"
```

6.105.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.00

Date

2012-04-25

From the value of the corresponding parameter, calls the chosen numerical scheme.

Copyright

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

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

Definition in file choice_scheme.cpp.

6.106 Sources/libschemes/order1.cpp File Reference

Order 1 scheme.

```
#include "order1.hpp"
```

6.106.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.01

Date

2012-08-10

Numerical scheme: at order 1.

Copyright

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

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

Definition in file order1.cpp.

6.107 Sources/libschemes/order2.cpp File Reference

Order 2 scheme.

```
#include "order2.hpp"
```

6.107.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.01

Date

2012-08-10

Numerical scheme: at order 2.

Copyright

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

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

Definition in file order2.cpp.

6.108 Sources/libschemes/scheme.cpp File Reference

Numerical scheme.

```
#include "scheme.hpp"
```

6.108.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.00.01

Date

2012-04-26

Common part for all the numerical schemes.

Copyright

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

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

Definition in file scheme.cpp.

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. 50, 68

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. 74

F. Bouchut. *Nonlinear stability of finite volume methods for hyperbolic conservation laws, and well-balanced schemes for sources*, volume 2/2004. Birkhäuser Basel, 2004. doi:10.1007/b95203. 43, 45, 47, 49, 52

F. Bouchut. Chapter 4 efficient numerical finite volume schemes for shallow water models. In V. Zeitlin, editor, *Nonlinear Dynamics of Rotating Shallow Water: Methods and Advances*, volume 2 of *Edited Series on Advances in Nonlinear Science and Complexity*, pages 189 – 256. Elsevier Science, 2007. doi:DOI: 10.1016/S1574-6909(06)02004-1. URL http://www.sciencedirect.com/science/article/B8JG3-4PS6TNS-6/2/f4c3dbcf476626c1cb6f353e9bc66cc9. 43, 45, 83

M.-O. Bristeau and B. Coussin. Boundary conditions for the shallow water equations solved by kinetic schemes. Technical Report RR-4282, INRIA, 2001. 25

O. Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles*. PhD thesis, Université d'Orléans, Orléans, France, July 2010. URL http://tel.archives-ouvertes.fr/tel-00531377. 130, 177

R. F. Dressler. Hydraulic resistance effect upon the dam-break functions. *Journal of Research of the National Bureau of Standards*, 49(3):217–225, Sept. 1952. 66

T. Gallouët, J.-M. Hérard, and N. Seguin. Some approximate godunov schemes to compute shallow-water equations with topography. *Computers & Fluids*, 32:479–513, 2003. 53

N. Goutal and F. Maurel. Proceedings of the 2^{nd} workshop on dam-break wave simulation. Technical Report HE-43/97/016/B, Electricité de France, Direction des études et recherches, 1997. 68, 71, 119

A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy. Some results on uniformly high-order accurate essentially nonoscillatory schemes. *Applied Numerical Mathematics*, 2(3-5):347 – 377, 1986. ISSN 0168-9274. doi:DOI: 10.1016/0168-9274(86)90039-5. URL http://www.sciencedirect.com/science/article/B6TYD-45GVV8T-J/2/830ca2dce5e3fb34ace9fa53ae5ab76b. Special Issue in Honor of Milt Rose's Sixtieth Birthday. 43

A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III. *Journal of Computational Physics*, 71:231–303, 1987. 43

A. Y. Le Roux. Conditions aux limites et problèmes hyperboliques : un point de vue numérique. Available in the document of the Conférence at IHP-Paris, 2001. 25

C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439 – 471, 1988. ISSN 0021-9991. doi:DOI: 10.1016/0021-9991(88)90177-5. URL http://www.sciencedirect.com/science/article/B6WHY-4DD1T6W-MM/2/bef99e4b67bd7132c8af1984c34ce57e. 43

M. W. Smith, N. J. Cox, and L. J. Bracken. Applying flow resistance equations to overland flows. *Progress in Physical Geography*, 31(4):363–387, 2007. doi:10.1177/0309133307081289. 58, 59

W. C. Thacker. Some exact solutions to the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508, 1981. doi:10.1017/S0022112081001882. URL http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=389055&fulltextType=RA&fileId=S0022112081001882. 70, 122

B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101 – 136, 1979. ISSN 0021-9991. doi:DOI:10.1016/0021-9991(79)90145-1. URL http://www.sciencedirect.com/science/article/B6WHY-4DD1N8T-C5/2/9b051d1cfcff715a3d0f4b7b7b0397cc. 83

Index