SWASHES

1.00.02

Generated by Doxygen 1.6.3

Mon Dec 19 19:41:05 2011

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

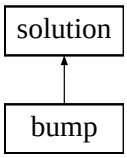
Class Documentation

4.1 bump Class Reference

class which allows to perform the bump analytic solutions

`#include <bump.hpp>`

Inheritance diagram for bump:



Public Member Functions

- bump (parameters &)
  Constructor.
- virtual ~bump ()
  Destructor.
- void compute ()
  Virtual method which is specific to each analytic solution.
- SCALAR function (SCALAR, SCALAR, SCALAR)
- SCALAR p (SCALAR a, SCALAR b, SCALAR c)
  coefficient Cardano method
- SCALAR q (SCALAR a, SCALAR b, SCALAR c, SCALAR d)
  coefficient Cardano method
- SCALAR determinant (SCALAR p, SCALAR q)
  Determinant in the Cardano method/related to number of roots.

- SCALAR height (SCALAR p, SCALAR q, SCALAR a, SCALAR b, SCALAR hnear)
  Computation of the 3rd order polynomia roots.
- void abcd (SCALAR q0, SCALAR hfin, SCALAR zbx, SCALAR zbfin, SCALAR &a, SCALAR &b, SCALAR &c, SCALAR &d)
  Enters the coefficients of the 3rd order polynomia we want to solve.
- SCALAR RHJump (SCALAR hplus, SCALAR hmois, SCALAR q)
  steady state RH relation (to check if zero at the shock)
- void param (SCALAR, SCALAR, int)

4.1.1 Detailed Description

class which allows to perform the bump analytic solutions

Definition at line 61 of file bump.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 bump::bump (parameters & *par*)

Constructor.

Definition at line 48 of file bump.cpp.

4.1.2.2 bump::~bump () `[virtual]`

Destructor.

Definition at line 147 of file bump.cpp.

4.1.3 Member Function Documentation

4.1.3.1 void bump::abcd (SCALAR *q0*, SCALAR *hfin*, SCALAR *zbx*, SCALAR *zbfin*, SCALAR & *a*, SCALAR & *b*, SCALAR & *c*, SCALAR & *d*)

Enters the coefficients of the 3rd order polynomia we want to solve.

Parameters

q0 : Inflow
hfin : exit height
zbx : bottom of the current cell
zbfin : exit bottom
a : coeff x3
b : coeff x2
c : coeff x
d : coeff 1

Definition at line 300 of file bump.cpp.

4.1.3.2 void bump::compute () [virtual]

Virtual method which is specific to each analytic solution.

Research of the limit x

PB !!

Computation of the height

Implements solution.

Definition at line 150 of file bump.cpp.

4.1.3.3 SCALAR bump::determinant (SCALAR *p*, SCALAR *q*)

Determinant in the Cardano method/related to number of roots.

Parameters

p : computed in function p

q : computed in function q

Definition at line 246 of file bump.cpp.

4.1.3.4 SCALAR bump::function (SCALAR, SCALAR, SCALAR)

4.1.3.5 SCALAR bump::height (SCALAR *p*, SCALAR *q*, SCALAR *a*, SCALAR *b*, SCALAR *hnear*)

Computation of the 3rd order polynomia roots.

Parameters

p : cardano coeff

q : cardano coeff

a :

b :

hnear : height of the previous or following cell (depending on the height computation direction)

Definition at line 252 of file bump.cpp.

4.1.3.6 SCALAR bump::p (SCALAR *a*, SCALAR *b*, SCALAR *c*)

coefficient Cardano method

Parameters

a,b,c : coeff of the 3rd order polynomia

Definition at line 234 of file bump.cpp.

4.1.3.7 void bump::param (SCALAR *L*, SCALAR *dx_ex*, int *Nx_ex*)

Definition at line 313 of file bump.cpp.

4.1.3.8 SCALAR bump::q (SCALAR *a*, SCALAR *b*, SCALAR *c*, SCALAR *d*)

coefficient Cardano method

Parameters

a,b,c,d : coeff of the 3rd order polynomia

Definition at line 240 of file bump.cpp.

4.1.3.9 SCALAR bump::RHJump (SCALAR *hplus*, SCALAR *hminus*, SCALAR *q*)

steady state RH relation (to check if zero at the shock)

Parameters

hplus : height right side

hminus : height left side

q : flow

Definition at line 308 of file bump.cpp.

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

- Headers/bump.hpp
- Sources/bump.cpp

4.2 choice_solution Class Reference

class which allows to choose the analytic solution.

`#include <choice_solution.hpp>`

Public Member Functions

- choice_solution (parameters &)
  Constructor.
- void compute ()
  Performs the solution.
- virtual ~choice_solution ()
  Destructor.

4.2.1 Detailed Description

class which allows to choose the analytic solution.

Definition at line 103 of file choice_solution.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 choice_solution::choice_solution (parameters & *par*)

Constructor.

Definition at line 45 of file choice_solution.cpp.

4.2.2.2 choice_solution::~choice_solution () `[virtual]`

Destructor.

Definition at line 481 of file choice_solution.cpp.

4.2.3 Member Function Documentation

4.2.3.1 void choice_solution::compute ()

Performs the solution.

Definition at line 477 of file choice_solution.cpp.

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

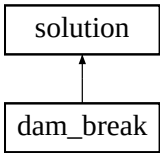
- Headers/choice_solution.hpp
- Sources/choice_solution.cpp

4.3 dam_break Class Reference

class which allows to perform the dam break analytic solution with wet and dry soil

`#include <dam_break.hpp>`

Inheritance diagram for dam_break:



Public Member Functions

- dam_break (parameters &)

  Constructor.

- virtual ~dam_break ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- SCALAR function (SCALAR, SCALAR, SCALAR)

  Function $x^6-9*cr^2*x^4+16*cl*cr^2*x^3-cr^2*(cr^2+8*cl^2)*x^2+cr^6$ to get the roots by dichotomy.

- void param (SCALAR, SCALAR, SCALAR, int, SCALAR)

4.3.1 Detailed Description

class which allows to perform the dam break analytic solution with wet and dry soil

Definition at line 61 of file dam_break.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 dam_break::dam_break (parameters & *par*)

Constructor.

Definition at line 48 of file dam_break.cpp.

4.3.2.2 dam_break::~dam_break () `[virtual]`

Destructor.

Definition at line 90 of file dam_break.cpp.

4.3.3 Member Function Documentation

4.3.3.1 void dam_break::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 95 of file dam_break.cpp.

4.3.3.2 SCALAR dam_break::function (SCALAR *x*, SCALAR *v_left*, SCALAR *v_right*)

Function $x^6-9*cr^2*x^4+16*cl*cr^2*x^3-cr^2*(cr^2+8*cl^2)*x^2+cr^6$ to get the roots by dichotomy.

Definition at line 163 of file dam_break.cpp.

4.3.3.3 void dam_break::param (SCALAR *L*, SCALAR *xdam*, SCALAR *dx_ex*, int *Nx_ex*, SCALAR *T*)

Definition at line 171 of file dam_break.cpp.

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

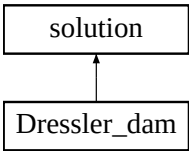
- Headers/dam_break.hpp
- Sources/dam_break.cpp

4.4 Dressler_dam Class Reference

class which allows to perform the Dressler dam break analytic solution

`#include <Dressler_dam.hpp>`

Inheritance diagram for Dressler_dam:



Public Member Functions

- Dressler_dam (parameters &)

  Constructor.

- virtual ~Dressler_dam ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, SCALAR, int, SCALAR)

4.4.1 Detailed Description

class which allows to perform the Dressler dam break analytic solution

Definition at line 61 of file Dressler_dam.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Dressler_dam::Dressler_dam (parameters & *par*)

Constructor.

Definition at line 48 of file Dressler_dam.cpp.

4.4.2.2 Dressler_dam::~Dressler_dam () `[virtual]`

Destructor.

Definition at line 78 of file Dressler_dam.cpp.

4.4.3 Member Function Documentation

4.4.3.1 void Dressler_dam::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 82 of file Dressler_dam.cpp.

4.4.3.2 void Dressler_dam::param (SCALAR *L*, SCALAR *xdam*, SCALAR *C*, SCALAR *dx_ex*, int *Nx_ex*, SCALAR *T*)

Definition at line 176 of file Dressler_dam.cpp.

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

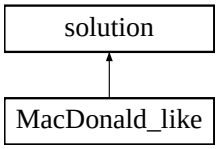
- Headers/Dressler_dam.hpp
- Sources/Dressler_dam.cpp

4.5 MacDonald_like Class Reference

class which allows to perform the MacDonald like analytic solution

`#include <MacDonald_like.hpp>`

Inheritance diagram for MacDonald_like:



Public Member Functions

- MacDonald_like (parameters &)

  Constructor.

- virtual ~MacDonald_like ()

  Destructor.

- void compute ()

  Performs the MacDonald_like solutions.

- SCALAR Delta_topo_Manning (SCALAR q, SCALAR h, SCALAR dh, SCALAR Rain, SCALAR n)

  Evaluation of the slope variation for Manning friction law.

- SCALAR Delta_topo_Darcy_Weisbach (SCALAR q, SCALAR h, SCALAR dh, SCALAR Rain, SCALAR n)

  Evaluation of the slope variation for Darcy-Weisbach friction law.

- void param (SCALAR, SCALAR, int)

4.5.1 Detailed Description

class which allows to perform the MacDonald like analytic solution

Definition at line 61 of file MacDonald_like.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 MacDonald_like::MacDonald_like (parameters & *par*)

Constructor.

Definition at line 47 of file MacDonald_like.cpp.

4.5.2.2 MacDonald_like::~MacDonald_like () [virtual]

Destructor.

Definition at line 410 of file MacDonald_like.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void MacDonald_like::compute () [virtual]

Performs the MacDonald_like solutions.

Implements solution.

Definition at line 415 of file MacDonald_like.cpp.

4.5.3.2 SCALAR MacDonald_like::Delta_topo_Darcy_Weisbach (SCALAR *q*, SCALAR *h*, SCALAR *dh*, SCALAR *Rain*, SCALAR *n*)

Evaluation of the slope variation for Darcy-Weisbach friction law.

Definition at line 454 of file MacDonald_like.cpp.

4.5.3.3 SCALAR MacDonald_like::Delta_topo_Manning (SCALAR *q*, SCALAR *h*, SCALAR *dh*, SCALAR *Rain*, SCALAR *n*)

Evaluation of the slope variation for Manning friction law.

Definition at line 450 of file MacDonald_like.cpp.

4.5.3.4 void MacDonald_like::param (SCALAR *L*, SCALAR *dx_ex*, int *Nx_ex*)

Definition at line 459 of file MacDonald_like.cpp.

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

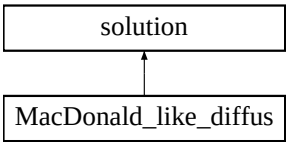
- Headers/MacDonald_like.hpp
- Sources/MacDonald_like.cpp

4.6 MacDonald_like_diffus Class Reference

class which allows to perform the MacDonald like analytic solution with diffusion

`#include <MacDonald_like_diffus.hpp>`

Inheritance diagram for MacDonald_like_diffus:



Public Member Functions

- MacDonald_like_diffus (parameters &)

  Constructor.

- virtual ~MacDonald_like_diffus ()

  Destructor.

- void compute ()

  Performs the MacDonald_like solutions with diffusion.

- SCALAR Delta_topo_diffus (SCALAR q, SCALAR h, SCALAR dh, SCALAR ddh, SCALAR kt, SCALAR kl, SCALAR muv, SCALAR muh)

  Evaluation of the slope variation.

- void param (SCALAR, SCALAR, int)

4.6.1 Detailed Description

class which allows to perform the MacDonald like analytic solution with diffusion

Definition at line 61 of file MacDonald_like_diffus.hpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MacDonald_like_diffus::MacDonald_like_diffus (parameters & *par*)

Constructor.

Definition at line 47 of file MacDonald_like_diffus.cpp.

4.6.2.2 MacDonald_like_diffus::~MacDonald_like_diffus () `[virtual]`

Destructor.

Definition at line 129 of file MacDonald_like_diffus.cpp.

4.6.3 Member Function Documentation

4.6.3.1 void MacDonald_like_diffus::compute () [virtual]

Performs the MacDonald_like solutions with diffusion.

Implements solution.

Definition at line 135 of file MacDonald_like_diffus.cpp.

4.6.3.2 SCALAR MacDonald_like_diffus::Delta_topo_diffus (SCALAR *q*, SCALAR *h*, SCALAR *dh*, SCALAR *ddh*, SCALAR *kt*, SCALAR *kl*, SCALAR *muv*, SCALAR *muh*)

Evaluation of the slope variation.

Definition at line 163 of file MacDonald_like_diffus.cpp.

4.6.3.3 void MacDonald_like_diffus::param (SCALAR *L*, SCALAR *dx_ex*, int *Nx_ex*)

Definition at line 167 of file MacDonald_like_diffus.cpp.

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

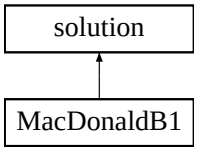
- Headers/MacDonald_like_diffus.hpp
- Sources/MacDonald_like_diffus.cpp

4.7 MacDonaldB1 Class Reference

class which allows to perform the MacDonald PSEUDO 2D analytic solution

`#include <MacDonaldB1.hpp>`

Inheritance diagram for MacDonaldB1:



Public Member Functions

- MacDonaldB1 (parameters &)

  Constructor.

- virtual ~MacDonaldB1 ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, int)
- SCALAR Delta_topo (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)

4.7.1 Detailed Description

class which allows to perform the MacDonald PSEUDO 2D analytic solution

Definition at line 61 of file MacDonaldB1.hpp.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 MacDonaldB1::MacDonaldB1 (parameters & *par*)

Constructor.

Definition at line 47 of file MacDonaldB1.cpp.

4.7.2.2 MacDonaldB1::~MacDonaldB1 () `[virtual]`

Destructor.

Definition at line 177 of file MacDonaldB1.cpp.

4.7.3 Member Function Documentation

4.7.3.1 void MacDonaldB1::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 142 of file MacDonaldB1.cpp.

4.7.3.2 SCALAR MacDonaldB1::Delta_topo (SCALAR *h*, SCALAR *hp*, SCALAR *b*, SCALAR *bp*, SCALAR *Q*, SCALAR *n*, SCALAR *Z*, SCALAR *exp1*, SCALAR *exp2*)

Definition at line 159 of file MacDonaldB1.cpp.

4.7.3.3 void MacDonaldB1::param (SCALAR *L*, SCALAR *dx_ex*, SCALAR *n*, int *Nx_ex*)

Definition at line 163 of file MacDonaldB1.cpp.

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

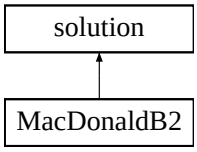
- Headers/MacDonaldB1.hpp
- Sources/MacDonaldB1.cpp

4.8 MacDonaldB2 Class Reference

class which allows to perform the MacDonald PSEUDO 2D analytic solution

`#include <MacDonaldB2.hpp>`

Inheritance diagram for MacDonaldB2:



Public Member Functions

- MacDonaldB2 (parameters &)

  Constructor.

- virtual ~MacDonaldB2 ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, int)
- SCALAR Delta_topo (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)

4.8.1 Detailed Description

class which allows to perform the MacDonald PSEUDO 2D analytic solution

Definition at line 61 of file MacDonaldB2.hpp.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 MacDonaldB2::MacDonaldB2 (parameters & *par*)

Constructor.

Definition at line 47 of file MacDonaldB2.cpp.

4.8.2.2 MacDonaldB2::~MacDonaldB2 () `[virtual]`

Destructor.

Definition at line 147 of file MacDonaldB2.cpp.

4.8.3 Member Function Documentation

4.8.3.1 void MacDonaldB2::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 112 of file MacDonaldB2.cpp.

4.8.3.2 SCALAR MacDonaldB2::Delta_topo (SCALAR *h*, SCALAR *hp*, SCALAR *b*, SCALAR *bp*, SCALAR *Q*, SCALAR *n*, SCALAR *Z*, SCALAR *exp1*, SCALAR *exp2*)

Definition at line 143 of file MacDonaldB2.cpp.

4.8.3.3 void MacDonaldB2::param (SCALAR *L*, SCALAR *dx_ex*, SCALAR *n*, int *Nx_ex*)

Definition at line 129 of file MacDonaldB2.cpp.

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

- Headers/MacDonaldB2.hpp
- Sources/MacDonaldB2.cpp

4.9 parameters Class Reference

class that defines the common parameters.

`#include <parameters.hpp>`

Public Member Functions

- parameters (int, char **)
- void setparameters (const char *)
- virtual ~parameters ()
- void help ()
- int get_Nxex () const
- int get_Nyex () const
- SCALAR get_choicedim () const
- int get_choicetype () const
- int get_choice () const
- int get_choicedomain () const

Protected Attributes

- int Nx_ex
- int Ny_ex
- SCALAR choicedim
- int choicetype
- int choice
- int choicedomain

4.9.1 Detailed Description

class that defines the common parameters.

Definition at line 57 of file parameters.hpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 parameters::parameters (int *argc*, char ** *argv*)

Definition at line 46 of file parameters.cpp.

4.9.2.2 parameters::~parameters () [virtual]

Definition at line 83 of file parameters.cpp.

4.9.3 Member Function Documentation

4.9.3.1 int parameters::get_choice () const

Definition at line 155 of file parameters.cpp.

4.9.3.2 SCALAR parameters::get_choicedim () const

Definition at line 159 of file parameters.cpp.

4.9.3.3 int parameters::get_choicedomain () const

Definition at line 167 of file parameters.cpp.

4.9.3.4 int parameters::get_choicetype () const

Definition at line 163 of file parameters.cpp.

4.9.3.5 int parameters::get_Nxex () const

Definition at line 147 of file parameters.cpp.

4.9.3.6 int parameters::get_Nyex () const

Definition at line 151 of file parameters.cpp.

4.9.3.7 void parameters::help ()

Definition at line 85 of file parameters.cpp.

4.9.3.8 void parameters::setparameters (const char *)

4.9.4 Member Data Documentation

4.9.4.1 int parameters::choice [protected]

Definition at line 62 of file parameters.hpp.

4.9.4.2 SCALAR parameters::choicedim [protected]

Definition at line 60 of file parameters.hpp.

4.9.4.3 int parameters::choicedomain [protected]

Definition at line 63 of file parameters.hpp.

4.9.4.4 int parameters::choicetype [protected]

Definition at line 61 of file parameters.hpp.

4.9.4.5 int parameters::Nx_ex [protected]

Definition at line 59 of file parameters.hpp.

4.9.4.6 int parameters::Ny_ex [protected]

Definition at line 59 of file parameters.hpp.

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

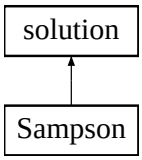
- Headers/parameters.hpp
- Sources/parameters.cpp

4.10 Sampson Class Reference

class which allows to perform the Sampson analytic solution

`#include <Sampson.hpp>`

Inheritance diagram for Sampson:



Public Member Functions

- Sampson (parameters &)

  Constructor.

- virtual ~Sampson ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)

4.10.1 Detailed Description

class which allows to perform the Sampson analytic solution

Definition at line 61 of file Sampson.hpp.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Sampson::Sampson (parameters & *par*)

Constructor.

Definition at line 47 of file Sampson.cpp.

4.10.2.2 Sampson::~Sampson () `[virtual]`

Destructor.

Definition at line 73 of file Sampson.cpp.

4.10.3 Member Function Documentation

4.10.3.1 void Sampson::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 78 of file Sampson.cpp.

4.10.3.2 void Sampson::param (SCALAR *L*, SCALAR *h0*, SCALAR *a*, SCALAR *B*, SCALAR *tau*, SCALAR *dx_ex*, SCALAR *T*, int *Nx_ex*)

Definition at line 98 of file Sampson.cpp.

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

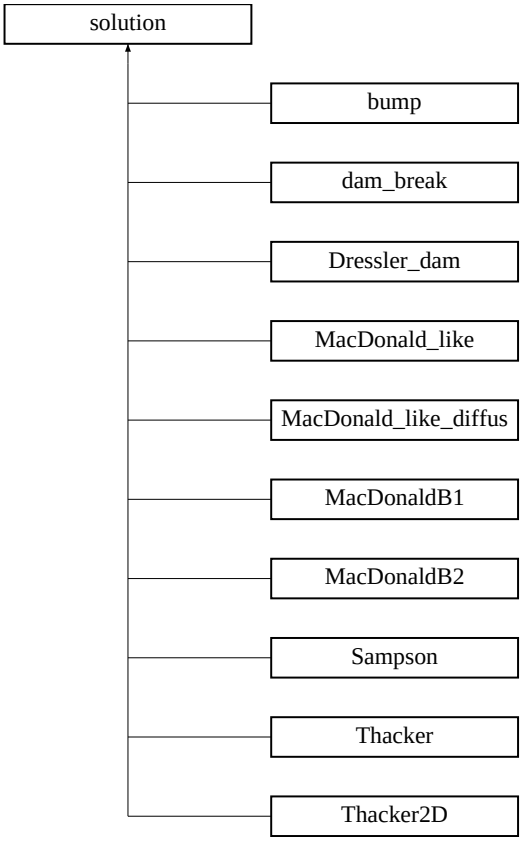
- Headers/Sampson.hpp
- Sources/Sampson.cpp

4.11 solution Class Reference

class which allows to perform the analytic solutions.

`#include <solution.hpp>`

Inheritance diagram for solution:



Public Member Functions

- solution (parameters &)

  Constructor.

- void allocation ()

  Tables allocation.

- void desallocation ()

  Tables desallocation.

- virtual void compute ()=0

  Virtual method which is specific to each analytic solution.

- void savefinalcritical (SCALAR *, SCALAR *, SCALAR *, SCALAR *)

  Save the analytic solution at the final time with the critical height.

- void savefinalmu (SCALAR *, SCALAR *, SCALAR *)

  Save the analytic solution at the final time without u.

- void savefinal2D (SCALAR *, SCALAR *, TAB, TAB, TAB, TAB)

  Save the analytic solution at the final time in 2D.

- void head (parameters &, string, string)

  Write the version of the software and the choice of the solution.

- virtual ~solution ()

  Destructor.

Protected Attributes

- int Nx_ex
- int Ny_ex
- SCALAR T
- SCALAR L
- SCALAR l
- SCALAR dx_ex
- SCALAR dy_ex
- SCALAR * xex
- SCALAR * yex
- SCALAR * hex
- SCALAR * uex
- SCALAR * qex
- SCALAR * zex

4.11.1 Detailed Description

class which allows to perform the analytic solutions.

Definition at line 60 of file solution.hpp.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 solution::solution (parameters & *par*)

Constructor.

Definition at line 47 of file solution.cpp.

4.11.2.2 solution::~solution () `[virtual]`

Destructor.

Definition at line 90 of file solution.cpp.

4.11.3 Member Function Documentation

4.11.3.1 void solution::allocation ()

Tables allocation.

Definition at line 95 of file solution.cpp.

4.11.3.2 virtual void solution::compute () [pure virtual]

Virtual method which is specific to each analytic solution.

Implemented in bump, dam_break, Dressler_dam, MacDonald_like, MacDonald_like_diffus, MacDonaldB1, MacDonaldB2, Sampson, Thacker, and Thacker2D.

4.11.3.3 void solution::desallocation ()

Tables desallocation.

Definition at line 128 of file solution.cpp.

4.11.3.4 void solution::head (parameters & *par*, string *solutiontype*, string *solutionchoice*)

Write the version of the software and the choice of the solution.

Definition at line 78 of file solution.cpp.

4.11.3.5 void solution::savefinal2D (SCALAR * *xex*, SCALAR * *yex*, TAB *hex2D*, TAB *uex2D*, TAB *vex2D*, TAB *zex2D*)

Save the analytic solution at the final time in 2D.

Definition at line 68 of file solution.cpp.

4.11.3.6 void solution::savefinalcritical (SCALAR * *xex*, SCALAR * *hex*, SCALAR * *uex*, SCALAR * *zex*)

Save the analytic solution at the final time with the critical height.

Definition at line 54 of file solution.cpp.

4.11.3.7 void solution::savefinalmu (SCALAR * *xex*, SCALAR * *hex*, SCALAR * *zex*)

Save the analytic solution at the final time without u.

Definition at line 61 of file solution.cpp.

4.11.4 Member Data Documentation

4.11.4.1 SCALAR solution::dx_ex [protected]

Definition at line 63 of file solution.hpp.

4.11.4.2 SCALAR solution::dy_ex [protected]

Definition at line 63 of file solution.hpp.

4.11.4.3 SCALAR* solution::hex [protected]

Definition at line 67 of file solution.hpp.

4.11.4.4 SCALAR solution::l [protected]

Definition at line 63 of file solution.hpp.

4.11.4.5 SCALAR solution::L [protected]

Definition at line 63 of file solution.hpp.

4.11.4.6 int solution::Nx_ex [protected]

Definition at line 62 of file solution.hpp.

4.11.4.7 int solution::Ny_ex [protected]

Definition at line 62 of file solution.hpp.

4.11.4.8 SCALAR* solution::qex [protected]

Definition at line 69 of file solution.hpp.

4.11.4.9 SCALAR solution::T [protected]

Definition at line 63 of file solution.hpp.

4.11.4.10 SCALAR* solution::uex [protected]

Definition at line 68 of file solution.hpp.

4.11.4.11 SCALAR* solution::xex [protected]

Definition at line 65 of file solution.hpp.

4.11.4.12 SCALAR* solution::yex [protected]

Definition at line 66 of file solution.hpp.

4.11.4.13 SCALAR* solution::zex [protected]

Definition at line 70 of file solution.hpp.

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

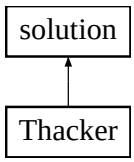
- Headers/solution.hpp
- Sources/solution.cpp

4.12 Thacker Class Reference

class which allows to perform the Thacker analytic solution

`#include <Thacker.hpp>`

Inheritance diagram for Thacker:



Public Member Functions

- Thacker (parameters &)

  Constructor.

- virtual ~Thacker ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)

4.12.1 Detailed Description

class which allows to perform the Thacker analytic solution

Definition at line 61 of file Thacker.hpp.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Thacker::Thacker (parameters & *par*)

Constructor.

Definition at line 47 of file Thacker.cpp.

4.12.2.2 Thacker::~Thacker () `[virtual]`

Destructor.

Definition at line 72 of file Thacker.cpp.

4.12.3 Member Function Documentation

4.12.3.1 void Thacker::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 77 of file Thacker.cpp.

4.12.3.2 void Thacker::param (SCALAR *L*, SCALAR *h0*, SCALAR *a*, SCALAR *dx_ex*, SCALAR *T*, int *Nx_ex*)

Definition at line 98 of file Thacker.cpp.

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

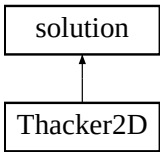
- Headers/Thacker.hpp
- Sources/Thacker.cpp

4.13 Thacker2D Class Reference

class which allows to perform the Thacker 2D analytic solution

`#include <Thacker2D.hpp>`

Inheritance diagram for Thacker2D:



Public Member Functions

- Thacker2D (parameters &)

  Constructor.

- virtual ~Thacker2D ()

  Destructor.

- void compute ()

  Virtual method which is specific to each analytic solution.

- void param (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int, int)

4.13.1 Detailed Description

class which allows to perform the Thacker 2D analytic solution

Definition at line 61 of file Thacker2D.hpp.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Thacker2D::Thacker2D (parameters & *par*)

Constructor.

Definition at line 47 of file Thacker2D.cpp.

4.13.2.2 Thacker2D::~Thacker2D () `[virtual]`

Destructor.

Definition at line 115 of file Thacker2D.cpp.

4.13.3 Member Function Documentation

4.13.3.1 void Thacker2D::compute () [virtual]

Virtual method which is specific to each analytic solution.

Implements solution.

Definition at line 131 of file Thacker2D.cpp.

4.13.3.2 void Thacker2D::param (SCALAR *L*, SCALAR *l*, SCALAR *h0*, SCALAR *a*, SCALAR *dx_ex*, SCALAR *dy_ex*, SCALAR *T*, int *Nx_ex*, int *Ny_ex*)

Definition at line 160 of file Thacker2D.cpp.

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

- Headers/Thacker2D.hpp
- Sources/Thacker2D.cpp

Chapter 5

File Documentation

5.1 Headers/bump.hpp File Reference

Performs the bump analytic solutions.

```
#include "solution.hpp"
#include <vector>
#include <iomanip>
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <fstream>
#include <complex>
#include <cstdlib>
```

Classes

- class bump

  class which allows to perform the bump analytic solutions

Defines

- #define Classe_bump

5.1.1 Detailed Description

Performs the bump analytic solutions.

Author

Anne-Celine Boulanger, Olivier Delestre

Definition in file bump.hpp.

5.1.2 Define Documentation

5.1.2.1 #define Classe_bump

Definition at line 48 of file bump.hpp.

5.2 Headers/choice_solution.hpp File Reference

Allows to choose the analytic solution.

```
#include "solution.hpp"
#include "dam_break.hpp"
```

Classes

- class choice_solution

  class which allows to choose the analytic solution.

Defines

- #define Class_choice_solution

5.2.1 Detailed Description

Allows to choose the analytic solution.

Author

Olivier Delestre

Definition in file choice_solution.hpp.

5.2.2 Define Documentation

5.2.2.1 #define Class_choice_solution

Definition at line 90 of file choice_solution.hpp.

5.3 Headers/dam_break.hpp File Reference

Performs the dam break analytic solution.

```
#include "solution.hpp"
```

Classes

- class dam_break

  class which allows to perform the dam break analytic solution with wet and dry soil

5.3.1 Detailed Description

Performs the dam break analytic solution.

Author

Olivier Delestre

Definition in file dam_break.hpp.

5.4 Headers/Dressler_dam.hpp File Reference

Performs classical and modified (personnal communication with Valerio Caleffi, illustration in Valiani et al. 1999) Dressler analytic solution.

```
#include "solution.hpp"
```

Classes

- class Dressler_dam

  class which allows to perform the Dressler dam break analytic solution

5.4.1 Detailed Description

Performs classical and modified (personnal communication with Valerio Caleffi, illustration in Valiani et al. 1999) Dressler analytic solution.

Author

Olivier Delestre

Definition in file Dressler_dam.hpp.

5.5 Headers/MacDonald_like.hpp File Reference

Performs the MacDonald like analytic solutions.

```
#include "solution.hpp"
```

Classes

- class MacDonald_like

  class which allows to perform the MacDonald like analytic solution

5.5.1 Detailed Description

Performs the MacDonald like analytic solutions.

Author

Olivier Delestre

Definition in file MacDonald_like.hpp.

5.6 Headers/MacDonald_like_diffus.hpp File Reference

Performs the MacDonald like analytic solutions with diffusion.

```
#include "solution.hpp"
```

Classes

- class MacDonald_like_diffus

  class which allows to perform the MacDonald like analytic solution with diffusion

5.6.1 Detailed Description

Performs the MacDonald like analytic solutions with diffusion.

Author

Olivier Delestre

Definition in file MacDonald_like_diffus.hpp.

5.7 Headers/MacDonaldB1.hpp File Reference

Performs the MacDonald PSEUDO 2D analytic solutions.

```
#include "solution.hpp"
```

Classes

- class MacDonaldB1

  class which allows to perform the MacDonald PSEUDO 2D analytic solution

5.7.1 Detailed Description

Performs the MacDonald PSEUDO 2D analytic solutions.

Author

Pierre-Antoine Ksinant and Carine Lucas

Definition in file MacDonaldB1.hpp.

5.8 Headers/MacDonaldB2.hpp File Reference

Performs the MacDonald PSEUDO 2D analytic solutions.

```
#include "solution.hpp"
```

Classes

- class MacDonaldB2

  class which allows to perform the MacDonald PSEUDO 2D analytic solution

5.8.1 Detailed Description

Performs the MacDonald PSEUDO 2D analytic solutions.

Author

Pierre-Antoine Ksinant and Carine Lucas

Definition in file MacDonaldB2.hpp.

5.9 Headers/misc.hpp File Reference

```
#include <vector>
#include <iomanip>
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <fstream>
#include <complex>
#include <cstdlib>
```

Defines

- #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 zero 0.
- #define PI 3.14159265
- #define version " SWASHES version 1.00.02, December 19, 2011"

Typedefs

- typedef double SCALAR
- typedef vector< vector< SCALAR > > TAB

5.9.1 Define Documentation

5.9.1.1 #define grav 9.81

Definition at line 55 of file misc.hpp.

5.9.1.2 #define grav_dem 4.905

Definition at line 56 of file misc.hpp.

5.9.1.3 #define max(a, b) (a>=b?a:b)

Definition at line 52 of file misc.hpp.

5.9.1.4 #define min(a, b) (a<=b?a:b)

Definition at line 53 of file misc.hpp.

5.9.1.5 #define PI 3.14159265

Definition at line 58 of file misc.hpp.

5.9.1.6 #define version " SWASHES version 1.00.02, December 19, 2011"

Definition at line 60 of file misc.hpp.

5.9.1.7 #define zero 0.

Definition at line 57 of file misc.hpp.

5.9.2 Typedef Documentation

5.9.2.1 typedef double SCALAR

Definition at line 64 of file misc.hpp.

5.9.2.2 typedef vector< vector< SCALAR > > TAB

Definition at line 65 of file misc.hpp.

5.10 Headers/parameters.hpp File Reference

Defines the common parameters.

Classes

- class parameters

  class that defines the common parameters.

5.10.1 Detailed Description

Defines the common parameters.

Author

Olivier Delestre

Definition in file parameters.hpp.

5.11 Headers/Sampson.hpp File Reference

Performs the Sampson analytic solution.

```
#include "solution.hpp"
```

Classes

- class Sampson

  class which allows to perform the Sampson analytic solution

Defines

- #define Class_sampson

5.11.1 Detailed Description

Performs the Sampson analytic solution.

Author

Olivier Delestre

Definition in file Sampson.hpp.

5.11.2 Define Documentation

5.11.2.1 #define Class_sampson

Definition at line 48 of file Sampson.hpp.

5.12 Headers/solution.hpp File Reference

Performs the analytic solutions.

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

Classes

- class solution

  class which allows to perform the analytic solutions.

5.12.1 Detailed Description

Performs the analytic solutions.

Author

Olivier Delestre

Definition in file solution.hpp.

5.13 Headers/Thacker.hpp File Reference

Performs the Thacker analytic solution.

```
#include "solution.hpp"
```

Classes

- class Thacker

  class which allows to perform the Thacker analytic solution

5.13.1 Detailed Description

Performs the Thacker analytic solution.

Author

Olivier Delestre

Definition in file Thacker.hpp.

5.14 Headers/Thacker2D.hpp File Reference

Performs the Thacker 2D analytic solutions.

```
#include "solution.hpp"
```

Classes

- class Thacker2D

  class which allows to perform the Thacker 2D analytic solution

5.14.1 Detailed Description

Performs the Thacker 2D analytic solutions.

Author

Pierre-Antoine Ksinant and Carine Lucas

Definition in file Thacker2D.hpp.

5.15 Sources/bump.cpp File Reference

```
#include "bump.hpp"
```

5.16 Sources/choice_solution.cpp File Reference

```
#include "choice_solution.hpp"
```

5.17 Sources/dam_break.cpp File Reference

```
#include "dam_break.hpp"
```

5.18 Sources/Dressler_dam.cpp File Reference

```
#include "Dressler_dam.hpp"
```

5.19 Sources/MacDonald_like.cpp File Reference

```
#include "MacDonald_like.hpp"
```

5.20 Sources/MacDonald_like_diffus.cpp File Reference

```
#include "MacDonald_like_diffus.hpp"
```

5.21 Sources/MacDonaldB1.cpp File Reference

```
#include "MacDonaldB1.hpp"
```

5.22 Sources/MacDonaldB2.cpp File Reference

```
#include "MacDonaldB2.hpp"
```

5.23 Sources/parameters.cpp File Reference

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

5.24 Sources/Sampson.cpp File Reference

```
#include "Sampson.hpp"
```

5.25 Sources/solution.cpp File Reference

```
#include "solution.hpp"
```

5.26 Sources/swashes.cpp File Reference

Main function. Declares the solution and calculates the chosen analytic solution for 1D Shallow Water equations.

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

Functions

- int main (int argc, char **argv)

5.26.1 Detailed Description

Main function. Declares the solution and calculates the chosen analytic solution for 1D Shallow Water equations.

Author

Olivier Delestre

Definition in file swashes.cpp.

5.26.2 Function Documentation

5.26.2.1 int main (int *argc*, char ** *argv*)

Definition at line 54 of file swashes.cpp.

5.27 Sources/Thacker.cpp File Reference

```
#include "Thacker.hpp"
```

5.28 Sources/Thacker2D.cpp File Reference

```
#include "Thacker2D.hpp"
```

Index