

Documentation
of
SWASHES

v1.01.02
(2012-03-09)

Generated by Doxygen 1.8.0
on Fri Mar 9 2012 20:12:19

Chapter 1

Todo List

Member `Bedload::Bedload (Parameters &)`

Exceptions should be treated.

Member `Choice_solution::Choice_solution (Parameters &)`

Exceptions should be treated.

Member `Dressler_dam::Dressler_dam (Parameters &)`

Exceptions should be treated.

Member `MacDonald_like::MacDonald_like (Parameters &)`

Exceptions should be treated.

Member `MacDonald_like_diffus::MacDonald_like_diffus (Parameters &)`

Exceptions should be treated.

Member `Parameters::Parameters (int, char **)`

Exceptions should be treated.

Member `Solution::allocation ()`

Exceptions should be treated.

Chapter 2

Class Index

2.1 Class Hierarchy

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

Choice_solution	15
Parameters	30
Solution	36
Bedload	9
Bump	11
Dam_break	17
Dressler_dam	19
MacDonald_like	21
MacDonald_like_diffus	23
MacDonaldB1	26
MacDonaldB2	28
Sampson	34
Thacker	42
Thacker2D	43

Chapter 3

Class Index

3.1 Class List

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

Bedload	Computes solutions with bedload	9
Bump	Computes bump solutions	11
Choice_solution	Choice of the solution	15
Dam_break	Computes dam break solutions	17
Dressler_dam	Computes Dressler dam break solution	19
MacDonald_like	Computes Mac Donald solutions	21
MacDonald_like_diffus	Computes Mac Donald solutions with diffusion	23
MacDonaldB1	Computes Mac Donald pseudo 2d solutions	26
MacDonaldB2	Computes Mac Donald pseudo 2d solutions	28
Parameters	Gets parameters	30
Sampson	Computes Sampson solution	34
Solution	Analytic solution	36
Thacker	Computes Thacker solution	42
Thacker2D	Computes Thacker solutions in 2D	43

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Headers/ bedload.hpp	
Computes solutions with bedload	47
Headers/ bump.hpp	
Computes bumps solutions	48
Headers/ choice_solution.hpp	
Choice of the solution	49
Headers/ dam_break.hpp	
Computes dam break solutions	50
Headers/ dressler_dam.hpp	
Computes Dressler dam break solution	50
Headers/ macdonald_like.hpp	
Computes Mac Donald solutions	51
Headers/ macdonald_like_diffus.hpp	
Computes Mac Donald solutions with diffusion	52
Headers/ macdonaldb1.hpp	
Computes Mac Donald pseudo 2d solutions	53
Headers/ macdonaldb2.hpp	
Computes Mac Donald pseudo 2d solutions	54
Headers/ misc.hpp	
Definitions	54
Headers/ parameters.hpp	
Gets parameters	56
Headers/ sampson.hpp	
Computes Sampson solution	57
Headers/ solution.hpp	
Common file	58
Headers/ thacker.hpp	
Computes Thacker solution	58
Headers/ thacker2d.hpp	
Computes Thacker solutions in 2D	59
Sources/ bedload.cpp	
Computes solutions with bedload	60
Sources/ bump.cpp	
Computes bumps solutions	60

Sources/ choice_solution.cpp	
Choice of the solution	61
Sources/ dam_break.cpp	
Computes dam break solutions	62
Sources/ dressler_dam.cpp	
Computes Dressler dam break solution	62
Sources/ macdonald_like.cpp	
Computes Mac Donald solutions	63
Sources/ macdonald_like_diffus.cpp	
Computes Mac Donald solutions with diffusion	64
Sources/ macdonaldb1.cpp	
Computes Mac Donald pseudo 2d solutions	64
Sources/ macdonaldb2.cpp	
Computes Mac Donald pseudo 2d solutions	65
Sources/ parameters.cpp	
Gets parameters	66
Sources/ sampson.cpp	
Computes Sampson solution	66
Sources/ solution.cpp	
Common file	67
Sources/ swashes.cpp	
Main file	68
Sources/ thacker.cpp	
Computes Thacker solution	69
Sources/ thacker2d.cpp	
Computes Thacker solution in 2D	69

Chapter 5

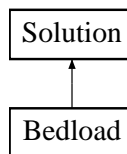
Class Documentation

5.1 Bedload Class Reference

Computes solutions with bedload.

```
#include <bedload.hpp>
```

Inheritance diagram for Bedload:



Public Member Functions

- [Bedload](#) ([Parameters](#) &)
Constructor.
- virtual [~Bedload](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- void [param](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.
- void [paramwarning](#) () const
Writes a warning about the the solution.

5.1.1 Detailed Description

Computes solutions with bedload.

Class that computes the solutions where the bed is moving with bedload, see [Berthon et al. \[2012\]](#).

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

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `Bedload::Bedload (Parameters & par)`

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

<code>in</code>	<code>par</code>	contains all the values from the parameters
-----------------	------------------	---

Warning

Problem: allocation of `z0` failed

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::T](#), [Solution::xex](#).

Note

If the vector `z0` cannot be allocated, the code will exit with failure termination code.

Todo Exceptions should be treated.

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

5.1.2.2 `Bedload::~~Bedload () [virtual]`

Destructor.

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

5.1.3 Member Function Documentation

5.1.3.1 `void Bedload::compute () [virtual]`

Computes the solution.

Computes the chosen bedload solution, see [Berthon et al. \[2012\]](#).

Modifies

[Solution::hex](#), [Solution::uex](#), [Solution::zex](#).

Implements [Solution](#).

Definition at line 155 of file `bedload.cpp`.

5.1.3.2 `void Bedload::param (SCALAR L, SCALAR dx_ex, SCALAR T, SCALAR uexl, SCALAR hexl, SCALAR z0l, SCALAR zexl, SCALAR uexr, SCALAR hexr, SCALAR z0r, SCALAR zexr, SCALAR alpha, SCALAR beta, SCALAR A, SCALAR q, SCALAR C, SCALAR p) const`

Writes the parameters of the solution.

Parameters

in	L	length of the domain
in	dx_ex	space step
in	T	final time
in	$uexl$	value of the velocity on the left boundary
in	$hexl$	value of the water height on the left boundary
in	$z0l$	value of the initial topography on the left boundary
in	$zexl$	value of the final topography on the left boundary
in	$uexr$	value of the velocity on the right boundary
in	$hexr$	value of the water height on the right boundary
in	$z0r$	value of the initial topography on the right boundary
in	$zexr$	value of the final topography on the right boundary
in	$alpha$	parameter for Exner equation
in	$beta$	parameter for Exner equation
in	A	parameter for Exner equation
in	q	parameter for Exner equation
in	C	parameter for Exner equation
in	ρ	parameter for Exner equation

Definition at line 175 of file bedload.cpp.

5.1.3.3 void Bedload::paramwarning () const

Writes a warning about the the solution.

Warning

WARNING: to compare your numerical result to this solution, you must be able to remove friction from the Shallow-Water part (see doc).

Definition at line 213 of file bedload.cpp.

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

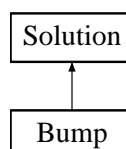
- Headers/[bedload.hpp](#)
- Sources/[bedload.cpp](#)

5.2 Bump Class Reference

Computes bump solutions.

```
#include <bump.hpp>
```

Inheritance diagram for Bump:



Public Member Functions

- [Bump](#) ([Parameters](#) &)
Constructor.
- virtual [~Bump](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- [SCALAR p](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Coefficient p for Cardano method.
- [SCALAR q](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Coefficient q for Cardano method.
- [SCALAR determinant](#) ([SCALAR](#), [SCALAR](#)) const
Determinant for Cardano method.
- [SCALAR height](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Computation of the 3rd order polynomia roots.
- void [abcd](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#) &, [SCALAR](#) &, [SCALAR](#) &, [SCALAR](#) &)
Defines a, b, c, d in order to solve $ah^3 + bh^2 + ch + d$.
- [SCALAR RHJump](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Steady state RH relation.
- void [param](#) ([SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.

5.2.1 Detailed Description

Computes bump solutions.

Class that computes the solutions with a bump for the topography, see [Delestre \[2010\]](#), [Goutal and Maurel \[1997\]](#).

Definition at line 71 of file bump.hpp.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Bump::Bump (Parameters & par)

Constructor.

Defines the physical parameters and prints the header with the configuration.

The solution is saved at the steady state.

Parameters

in	par	contains all the values from the parameters
----	-----	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::zex](#) to have the bump configuration.

Definition at line 60 of file bump.cpp.

5.2.2.2 `Bump::~~Bump()` [virtual]

Destructor.

Definition at line 166 of file bump.cpp.

5.2.3 Member Function Documentation**5.2.3.1** `void Bump::abcd (SCALAR q_in, SCALAR h_out, SCALAR zbx, SCALAR zbfm, SCALAR & a, SCALAR & b, SCALAR & c, SCALAR & d)`

Defines *a*, *b*, *c*, *d* in order to solve $ah^3 + bh^2 + ch + d$.

Enters the coefficients of the 3rd order polynomia we want to solve: $ah^3 + bh^2 + ch + d$.

Parameters

in	<i>q_in</i>	inflow discharge
in	<i>h_out</i>	water height at the outflow
in	<i>zbx</i>	bottom topography of the current cell
in	<i>zbfm</i>	bottom topography at the outflow
out	<i>a</i>	coefficient of the 3rd order polynomia
out	<i>b</i>	coefficient of the 3rd order polynomia
out	<i>c</i>	coefficient of the 3rd order polynomia
out	<i>d</i>	coefficient of the 3rd order polynomia

Definition at line 363 of file bump.cpp.

5.2.3.2 `void Bump::compute ()` [virtual]

Computes the solution.

Computes the chosen bump solution, see [Delestre \[2010\]](#), [Goutal and Maurel \[1997\]](#).

Modifies

[Solution::hex](#).

Implements [Solution](#).

Definition at line 169 of file bump.cpp.

5.2.3.3 `SCALAR Bump::determinant (SCALAR p, SCALAR q)` const

Determinant for Cardano method.

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

Parameters

in	<i>p</i>	computed by Bump::p
in	<i>q</i>	computed by Bump::q

Returns

Value of $q^2 + \frac{4}{27}p^3$.

Definition at line 289 of file bump.cpp.

5.2.3.4 SCALAR Bump::height (SCALAR p, SCALAR q, SCALAR a, SCALAR b, SCALAR hnear) const

Computation of the 3rd order polynomia roots.

Parameters

in	<i>p</i>	computed by Bump::p
in	<i>q</i>	computed by Bump::q
in	<i>a</i>	coefficient of the 3rd order polynomia
in	<i>b</i>	coefficient of the 3rd order polynomia
in	<i>hnear</i>	height of the previous or following cell (depending on the height computation direction)

Warning

Error: no positive height.

Error: Probably irregular solution.

Returns

h, the water height.

Definition at line 303 of file bump.cpp.

5.2.3.5 SCALAR Bump::p (SCALAR a, SCALAR b, SCALAR c) const

Coefficient p for Cardano method.

Parameters

in	<i>a</i>	coefficient of the 3rd order polynomia
in	<i>b</i>	coefficient of the 3rd order polynomia
in	<i>c</i>	coefficient of the 3rd order polynomia

Returns

Value of $-\frac{b^2}{3a^2} + \frac{c}{a}$.

Definition at line 260 of file bump.cpp.

5.2.3.6 void Bump::param (SCALAR L, SCALAR dx.ex) const

Writes the parameters of the solution.

Parameters

in	<i>L</i>	length of the domain
in	<i>dx_ex</i>	space step

Definition at line 399 of file bump.cpp.

5.2.3.7 SCALAR Bump::q (SCALAR *a*, SCALAR *b*, SCALAR *c*, SCALAR *d*) const

Coefficient *q* for Cardano method.

Parameters

in	<i>a</i>	coefficient of the 3rd order polynomia
in	<i>b</i>	coefficient of the 3rd order polynomia
in	<i>c</i>	coefficient of the 3rd order polynomia
in	<i>d</i>	coefficient of the 3rd order polynomia

Returns

$$\text{Value of } \frac{b}{27a} \left(\frac{2b^2}{a^2} - 9\frac{c}{a} \right).$$

Definition at line 273 of file bump.cpp.

5.2.3.8 SCALAR Bump::RHJump (SCALAR *hplus*, SCALAR *hminus*, SCALAR *q*) const

Steady state RH relation.

Parameters

in	<i>hplus</i>	water height on the right side
in	<i>hminus</i>	water height on the left side
in	<i>q</i>	discharge

Returns

$$\text{Value of } \left| q^2 \left(\frac{1}{hplus} - \frac{1}{hminus} \right) + \frac{g}{2} (hplus^2 - hminus^2) \right|.$$

Definition at line 385 of file bump.cpp.

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

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

5.3 Choice_solution Class Reference

Choice of the solution.

```
#include <choice_solution.hpp>
```

Public Member Functions

- [Choice_solution](#) (Parameters &)
Constructor.
- void [compute](#) ()
Computes the solution.
- virtual [~Choice_solution](#) ()
Destructor.

5.3.1 Detailed Description

Choice of the solution.

Class that calls the chosen solution.

Definition at line 117 of file choice_solution.hpp.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Choice_solution::Choice_solution (Parameters & par)

Constructor.

Parameters

in	par	contains all the values from the parameter
----	-----	--

Warning

Error: the dimension is ***
This *** solution for L=*** does not exist!
*** solutions for the domain *** do not exist!

Note

If the solution does not exists, the code will exit with failure termination code.

Todo Exceptions should be treated.

Definition at line 59 of file choice_solution.cpp.

5.3.2.2 Choice_solution::~~Choice_solution () [virtual]

Destructor.

Definition at line 541 of file choice_solution.cpp.

5.3.3 Member Function Documentation

5.3.3.1 void Choice_solution::compute ()

Computes the solution.

Definition at line 537 of file choice_solution.cpp.

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

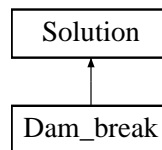
- Headers/[choice_solution.hpp](#)
- Sources/[choice_solution.cpp](#)

5.4 Dam_break Class Reference

Computes dam break solutions.

```
#include <dam_break.hpp>
```

Inheritance diagram for Dam_break:



Public Member Functions

- [Dam_break](#) ([Parameters](#) &)
Constructor.
- virtual [~Dam_break](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- [SCALAR](#) function ([SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Function $x^6 - 9v_{right}^2 x^4 + 16v_{left} v_{right}^2 x^3 - v_{right}^2 (v_{right}^2 + 8v_{left}^2) x^2 + v_{right}^6$ to get the roots by dichotomy.
- void [param](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.

5.4.1 Detailed Description

Computes dam break solutions.

Class that computes the solutions for a dam break without friction, see [Ritter \[1892\]](#) [Stoker \[1957\]](#).

Definition at line 69 of file dam_break.hpp.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Dam_break::Dam_break (Parameters & par)

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

in	par	contains all the values from the parameters
----	-----	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::T](#), [Solution::xex](#), [Solution::zex](#) to have the dam break configuration.

Definition at line 58 of file dam_break.cpp.

5.4.2.2 Dam_break::~~Dam_break () [virtual]

Destructor.

Definition at line 106 of file dam_break.cpp.

5.4.3 Member Function Documentation**5.4.3.1 void Dam_break::compute () [virtual]**

Computes the solution.

Computes the chosen dam break solution, see [Ritter \[1892\]](#) [Stoker \[1957\]](#).

Modifies

[Solution::hex](#).

Implements [Solution](#).

Definition at line 111 of file dam_break.cpp.

5.4.3.2 SCALAR Dam_break::function (SCALAR x, SCALAR v_left, SCALAR v_right) const

Function $x^6 - 9v_{right}^2 x^4 + 16v_{left} v_{right}^2 x^3 - v_{right}^2 (v_{right}^2 + 8v_{left}^2) x^2 + v_{right}^6$ to get the roots by dichotomy.

Function to solve by dichotomy the equation $cm^6 - 9v_{right}^2 cm^4 + 16v_{left} v_{right}^2 cm^3 - v_{right}^2 (v_{right}^2 + 8v_{left}^2) cm^2 + v_{right}^6 = 0$.

Returns

Value of $x^6 - 9v_{right}^2 x^4 + 16v_{left} v_{right}^2 x^3 - v_{right}^2 (v_{right}^2 + 8v_{left}^2) x^2 + v_{right}^6$.

Definition at line 186 of file dam_break.cpp.

5.4.3.3 void Dam_break::param (SCALAR L, SCALAR xdam, SCALAR dx_ex, SCALAR T) const

Writes the parameters of the solution.

Parameters

in	<i>L</i>	length of the domain
in	<i>xdam</i>	position of the dam
in	<i>dx_ex</i>	space step
in	<i>T</i>	final time

Definition at line 198 of file dam_break.cpp.

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

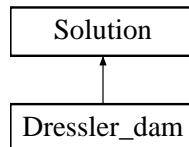
- [Headers/dam_break.hpp](#)
- [Sources/dam_break.cpp](#)

5.5 Dressler_dam Class Reference

Computes Dressler dam break solution.

```
#include <dressler_dam.hpp>
```

Inheritance diagram for Dressler_dam:



Public Member Functions

- [Dressler_dam \(Parameters &\)](#)
Constructor.
- [virtual ~Dressler_dam \(\)](#)
Destructor.
- [void compute \(\)](#)
Computes the solution.
- [void param \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\) const](#)
Writes the parameters of the solution.

5.5.1 Detailed Description

Computes Dressler dam break solution.

Class that computes the solutions for a dam break with friction, see [Dressler \[1952\]](#).

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

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Dressler_dam::Dressler_dam (Parameters & par)

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

<i>in</i>	<i>par</i>	contains all the values from the parameters
-----------	------------	---

Warning

Problem: allocation of hexd failed.

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::zex](#) to have Dressler dam break configuration.

Note

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

Todo Exceptions should be treated.

Definition at line 60 of file dressler_dam.cpp.

5.5.2.2 Dressler_dam::~~Dressler_dam() [virtual]

Destructor.

Definition at line 103 of file dressler_dam.cpp.

5.5.3 Member Function Documentation**5.5.3.1 void Dressler_dam::compute() [virtual]**

Computes the solution.

Computes Dressler solution, see [Dressler \[1952\]](#).

Modifies

[Solution::hex](#), [Solution::uex](#).

Implements [Solution](#).

Definition at line 107 of file dressler_dam.cpp.

5.5.3.2 void Dressler_dam::param (SCALAR L, SCALAR xdam, SCALAR C, SCALAR dx_ex, SCALAR T) const

Writes the parameters of the solution.

Parameters

in	L	length of the domain
in	x_{dam}	position of the dam
in	C	Chezy friction coefficient
in	dx_{ex}	space step
in	T	final time

Definition at line 208 of file dressler_dam.cpp.

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

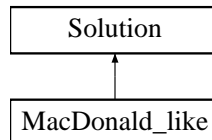
- [Headers/dressler_dam.hpp](#)
- [Sources/dressler_dam.cpp](#)

5.6 MacDonald_like Class Reference

Computes Mac Donald solutions.

```
#include <macdonald_like.hpp>
```

Inheritance diagram for MacDonald_like:



Public Member Functions

- [MacDonald_like \(Parameters &\)](#)
Constructor.
- [virtual ~MacDonald_like \(\)](#)
Destructor.
- [void compute \(\)](#)
Computes the solution.
- [SCALAR Delta_topo_Manning \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\) const](#)
Evaluation of the slope variation for Manning friction law.
- [SCALAR Delta_topo_Darcy_Weisbach \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\) const](#)
Evaluation of the slope variation for Darcy-Weisbach friction law.
- [void param \(SCALAR, SCALAR\) const](#)
Writes the parameters of the solution.

5.6.1 Detailed Description

Computes Mac Donald solutions.

Class that computes Mac Donald solutions in 1d, see [MacDonald \[1996\]](#) [MacDonald et al. \[1997\]](#) [Delestre \[2010\]](#) [Vo \[2008\]](#).

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

5.6.2 Constructor & Destructor Documentation

5.6.2.1 MacDonald_like::MacDonald_like (Parameters & par)

Constructor.

Defines the physical parameters and prints the header with the configuration.

The solution is saved at the steady state.

Parameters

<i>in</i>	<i>par</i>	contains all the values from the parameters
-----------	------------	---

Warning

Problem: allocation of dhex failed.

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::hex](#), [Solution::qex](#) to have Mac Donald configuration.

Note

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

Todo Exceptions should be treated.

Definition at line 59 of file macdonald_like.cpp.

5.6.2.2 MacDonald_like::~MacDonald_like () [virtual]

Destructor.

Definition at line 435 of file macdonald_like.cpp.

5.6.3 Member Function Documentation

5.6.3.1 void MacDonald_like::compute () [virtual]

Computes the solution.

Computes Mac Donald solutions, see [MacDonald \[1996\]](#) [MacDonald et al. \[1997\]](#) [Delestre \[2010\]](#) [Vo \[2008\]](#).

Modifies

[Solution::zex](#).

Implements [Solution](#).

Definition at line 440 of file macdonald_like.cpp.

5.6.3.2 SCALAR MacDonald_like::Delta_topo_Darcy_Weisbach (SCALAR *q*, SCALAR *h*, SCALAR *dh*, SCALAR *Rain*, SCALAR *c*) const

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

Parameters

<i>in</i>	<i>q</i>	discharge
<i>in</i>	<i>h</i>	water height
<i>in</i>	<i>dh</i>	variation of the water height
<i>in</i>	<i>Rain</i>	rain quantity
<i>in</i>	<i>c</i>	friction coefficient

Returns

$$\text{Value of } \left(1 - \frac{q^2}{gh^3}\right) dh + 2Rain \frac{q}{gh^2} + c \frac{q^2}{8gh^3}.$$

Definition at line 497 of file macdonald_like.cpp.

5.6.3.3 SCALAR MacDonald_like::Delta_topo_Manning (SCALAR *q*, SCALAR *h*, SCALAR *dh*, SCALAR *Rain*, SCALAR *c*) const

Evaluation of the slope variation for Manning friction law.

Parameters

in	<i>q</i>	discharge
in	<i>h</i>	water height
in	<i>dh</i>	variation of the water height
in	<i>Rain</i>	rain quantity
in	<i>c</i>	friction coefficient

Returns

$$\text{Value of } \left(1 - \frac{q^2}{gh^3}\right) dh + 2Rain \frac{q}{gh^2} + \frac{c^2 q^2}{h^{10/3}}.$$

Definition at line 482 of file macdonald_like.cpp.

5.6.3.4 void MacDonald_like::param (SCALAR *L*, SCALAR *dx_ex*) const

Writes the parameters of the solution.

Parameters

in	<i>L</i>	length of the domain
in	<i>dx_ex</i>	space step

Definition at line 513 of file macdonald_like.cpp.

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

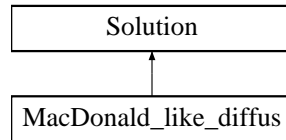
- Headers/[macdonald_like.hpp](#)
- Sources/[macdonald_like.cpp](#)

5.7 MacDonald_like_diffus Class Reference

Computes Mac Donald solutions with diffusion.

```
#include <macdonald_like_diffus.hpp>
```

Inheritance diagram for MacDonald_like_diffus:



Public Member Functions

- [MacDonal_like_diffus](#) ([Parameters](#) &)
Constructor.
- virtual [~MacDonal_like_diffus](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- [SCALAR Delta_topo_diffus](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Evaluation of the slope variation.
- void [param](#) ([SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.

5.7.1 Detailed Description

Computes Mac Donald solutions with diffusion.

Class that computes Mac Donald solutions in 1d with diffusion, see [Delestre and Marche \[2010\]](#).

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `MacDonal_like_diffus::MacDonal_like_diffus (Parameters & par)`

Constructor.

Defines the physical parameters and prints the header with the configuration.

The solution is saved at the steady state.

Parameters

<code>in</code>	<code>par</code>	contains all the values from the parameters
-----------------	------------------	---

Warning

Problem: allocation of `dhex` failed.

Problem: allocation of `ddhex` failed.

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::hex](#), [Solution::qex](#) to have Mac Donald configuration.

Note

If the vector `dhex` (or `ddhex`) cannot be allocated, the code will exit with failure termination code.

Todo Exceptions should be treated.

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

5.7.2.2 MacDonald_like_diffus::~MacDonald_like_diffus () [virtual]

Destructor.

Definition at line 155 of file `macdonald_like_diffus.cpp`.

5.7.3 Member Function Documentation**5.7.3.1 void MacDonald_like_diffus::compute ()** [virtual]

Computes the solution.

Computes Mac Donald solutions with diffusion, see [Delestre and Marche \[2010\]](#).

Modifies

[Solution::zex](#).

Implements [Solution](#).

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

5.7.3.2 SCALAR MacDonald_like_diffus::Delta_topo_diffus (SCALAR q, SCALAR h, SCALAR dh, SCALAR ddh, SCALAR kt, SCALAR kl, SCALAR muv, SCALAR muh) const

Evaluation of the slope variation.

Parameters

in	<i>q</i>	discharge
in	<i>h</i>	water height
in	<i>dh</i>	variation of the water height
in	<i>ddh</i>	second order derivative of h
in	<i>kt</i>	turbulent coefficient
in	<i>kl</i>	laminar coefficient
in	<i>muv</i>	vertical viscosity
in	<i>muh</i>	horizontal viscosity

Returns

$$\text{Value of } \left(1 - \frac{q^2}{gh^3}\right) dh + \frac{klq}{gh^2 \left(1 + \frac{klh}{3muv}\right)} + \frac{ktq^2}{gh^2 \left(1 + \frac{klh}{3muv}\right)^2} + 4muh \frac{qddh - \frac{qdh^2}{h}}{gh^2}.$$

Definition at line 196 of file `macdonald_like_diffus.cpp`.

5.7.3.3 void MacDonald_like_diffus::param (SCALAR L, SCALAR dx_ex) const

Writes the parameters of the solution.

Parameters

in	L	length of the domain
in	dx_ex	space step

Definition at line 214 of file macdonald_like_diffus.cpp.

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

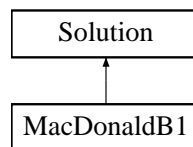
- Headers/[macdonald_like_diffus.hpp](#)
- Sources/[macdonald_like_diffus.cpp](#)

5.8 MacDonaldB1 Class Reference

Computes Mac Donald pseudo 2d solutions.

```
#include <macdonaldb1.hpp>
```

Inheritance diagram for MacDonaldB1:



Public Member Functions

- [MacDonaldB1 \(Parameters &\)](#)
Constructor.
- virtual [~MacDonaldB1 \(\)](#)
Destructor.
- void [compute \(\)](#)
Computes the solution.
- void [param \(SCALAR, SCALAR, SCALAR\) const](#)
Writes the parameters of the solution.
- [SCALAR Delta_topo \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\) const](#)
Evaluation of the slope variation.

5.8.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Class that computes Mac Donald pseudo 2d solutions with bottom B1, see [MacDonald \[1996\]](#).

Definition at line 69 of file macdonaldb1.hpp.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 MacDonaldB1::MacDonaldB1 (Parameters & *par*)

Constructor.

Defines the physical parameters and prints the header with the configuration.

The solution is saved at the steady state.

Parameters

<i>in</i>	<i>par</i>	contains all the values from the parameters
-----------	------------	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::hex](#) to have Mac Donald configuration.

Definition at line 58 of file macdonaldb1.cpp.

5.8.2.2 MacDonaldB1::~~MacDonaldB1 () [virtual]

Destructor.

Definition at line 227 of file macdonaldb1.cpp.

5.8.3 Member Function Documentation

5.8.3.1 void MacDonaldB1::compute () [virtual]

Computes the solution.

Computes Mac Donald solutions with bottom B1, see [MacDonald \[1996\]](#).

Modifies

[Solution::zex](#).

Implements [Solution](#).

Definition at line 162 of file macdonaldb1.cpp.

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

Evaluation of the slope variation.

Parameters

<i>in</i>	<i>h</i>	water height
<i>in</i>	<i>hp</i>	derivative of the water height
<i>in</i>	<i>b</i>	boundary function
<i>in</i>	<i>bp</i>	derivative of the boundary function
<i>in</i>	<i>Q</i>	discharge
<i>in</i>	<i>n</i>	friction coefficient
<i>in</i>	<i>Z</i>	slope
<i>in</i>	<i>exp1</i>	exponent, equal to 4/3
<i>in</i>	<i>exp2</i>	exponent, equal to 10/3

Returns

$$\text{Value of } hp \left(\frac{Q^2(b+2Zh)}{g(h(b+Zh))^3} - 1 \right) - Q^2 n^2 \frac{(b+2h\sqrt{1+Z^2})^{exp1}}{(h(b+Zh))^{exp2}} + \frac{Q^2 bp}{gh^2(b+Zh)^3}.$$

Definition at line 186 of file macdonaldb1.cpp.

5.8.3.3 void MacDonaldB1::param (SCALAR L, SCALAR dx_ex, SCALAR n) const

Writes the parameters of the solution.

Parameters

in	<i>L</i>	length of the domain
in	<i>dx_ex</i>	space step
in	<i>n</i>	friction coefficient

Definition at line 205 of file macdonaldb1.cpp.

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

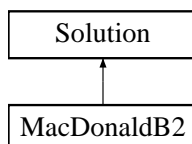
- Headers/[macdonaldb1.hpp](#)
- Sources/[macdonaldb1.cpp](#)

5.9 MacDonaldB2 Class Reference

Computes Mac Donald pseudo 2d solutions.

```
#include <macdonaldb2.hpp>
```

Inheritance diagram for MacDonaldB2:

**Public Member Functions**

- [MacDonaldB2 \(Parameters &\)](#)
Constructor.
- virtual [~MacDonaldB2 \(\)](#)
Destructor.
- void [compute \(\)](#)
Computes the solution.
- void [param \(SCALAR, SCALAR, SCALAR\) const](#)
Writes the parameters of the solution.
- [SCALAR Delta_topo \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\) const](#)
Evaluation of the slope variation.

5.9.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Class that computes Mac Donald pseudo 2d solutions with bottom B2, see [MacDonald \[1996\]](#).

Definition at line 70 of file macdonaldb2.hpp.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 MacDonaldB2::MacDonaldB2 (Parameters & par)

Constructor.

Defines the physical parameters and prints the header with the configuration.

The solution is saved at the steady state.

Parameters

<i>in</i>	<i>par</i>	contains all the values from the parameters
-----------	------------	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::hex](#) to have Mac Donald configuration.

Definition at line 58 of file macdonaldb2.cpp.

5.9.2.2 MacDonaldB2::~~MacDonaldB2 () [virtual]

Destructor.

Definition at line 197 of file macdonaldb2.cpp.

5.9.3 Member Function Documentation

5.9.3.1 void MacDonaldB2::compute () [virtual]

Computes the solution.

Computes Mac Donald solutions with bottom B2, see [MacDonald \[1996\]](#).

Modifies

[Solution::zex](#).

Implements [Solution](#).

Definition at line 132 of file macdonaldb2.cpp.

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

Evaluation of the slope variation.

Parameters

in	h	water height
in	hp	derivative of the water height
in	b	boundary function
in	bp	derivative of the boundary function
in	Q	discharge
in	n	friction coefficient
in	Z	slope
in	$exp1$	exponent, equal to 4/3
in	$exp2$	exponent, equal to 10/3

Returns

$$\text{Value of } hp \left(\frac{Q^2(b+2Zh)}{g(h(b+Zh))^3} - 1 \right) - Q^2 n^2 \frac{(b+2h\sqrt{1+Z^2})^{exp1}}{(h(b+Zh))^{exp2}} + \frac{Q^2 bp}{gh^2(b+Zh)^3}.$$

Definition at line 178 of file macdonaldb2.cpp.

5.9.3.3 void MacDonaldB2::param (SCALAR L, SCALAR dx_ex, SCALAR n) const

Writes the parameters of the solution.

Parameters

in	L	length of the domain
in	dx_ex	space step
in	n	friction coefficient

Definition at line 156 of file macdonaldb2.cpp.

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

- Headers/[macdonaldb2.hpp](#)
- Sources/[macdonaldb2.cpp](#)

5.10 Parameters Class Reference

Gets parameters.

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

Public Member Functions

- [Parameters](#) (int, char **)
 - Constructor.*
- virtual [~Parameters](#) ()
 - Destructor.*
- void [help](#) () const
 - Prints help.*
- int [get_nxex](#) () const

- Gives the number of cells in x.*
- int [get_nyex](#) () const
Gives the number of cells in y.
- **SCALAR** [get_choicedim](#) () const
Gives the dimension.
- int [get_choicetype](#) () const
Gives the type.
- int [get_choice](#) () const
Gives the chosen solution.
- int [get_choicedomain](#) () const
Gives the domain.

Protected Attributes

- int [nx_ex](#)
- int [ny_ex](#)
- **SCALAR** [choicedim](#)
- int [choicetype](#)
- int [choice](#)
- int [choicedomain](#)

5.10.1 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 69 of file parameters.hpp.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Parameters::Parameters (int *argc*, char ** *argv*)

Constructor.

Checks the arguments

Parameters

in	<i>argc</i>	number of arguments
in	<i>argv</i>	value of the arguments

Warning

The number of cells in x must be positive!
The number of cells in y must be positive!

Modifies

[Parameters::choicedim](#), [Parameters::choicetype](#), [Parameters::choicedomain](#), [Parameters::choice](#) with the values given in argument.

Note

If the arguments are incompatible, the code will exit with failure termination code.

Todo Exceptions should be treated.

Definition at line 58 of file parameters.cpp.

5.10.2.2 Parameters::~Parameters () [virtual]

Destructor.

Definition at line 110 of file parameters.cpp.

5.10.3 Member Function Documentation**5.10.3.1 int Parameters::get_choice () const**

Gives the chosen solution.

Returns

The chosen solution.

Definition at line 207 of file parameters.cpp.

5.10.3.2 SCALAR Parameters::get_choicedim () const

Gives the dimension.

Returns

The dimension of the solution.

Definition at line 217 of file parameters.cpp.

5.10.3.3 int Parameters::get_choicedomain () const

Gives the domain.

Returns

The domain of the solution.

Definition at line 237 of file parameters.cpp.

5.10.3.4 int Parameters::get_choicetype () const

Gives the type.

Returns

The type of the solution.

Definition at line 227 of file parameters.cpp.

5.10.3.5 int Parameters::get_nxex () const

Gives the number of cells in x.

Returns

The number of cells in x.

Definition at line 187 of file parameters.cpp.

5.10.3.6 int Parameters::get_nyex () const

Gives the number of cells in y.

Returns

The number of cells in y.

Definition at line 197 of file parameters.cpp.

5.10.3.7 void Parameters::help () const

Prints help.

Prints how to use the code.

Definition at line 113 of file parameters.cpp.

5.10.4 Member Data Documentation

5.10.4.1 int Parameters::choice [protected]

Value corresponding to the chosen solution.

Definition at line 81 of file parameters.hpp.

5.10.4.2 SCALAR Parameters::choicedim [protected]

Value corresponding to the dimension of the solution.

Definition at line 77 of file parameters.hpp.

5.10.4.3 int Parameters::choicedomain [protected]

Value corresponding to the domain of the solution.

Definition at line 83 of file parameters.hpp.

5.10.4.4 int Parameters::choicetype [protected]

Value corresponding to the type of the solution.

Definition at line 79 of file parameters.hpp.

5.10.4.5 int Parameters::nx_ex [protected]

Number of cells in x.

Definition at line 73 of file parameters.hpp.

5.10.4.6 int Parameters::ny_ex [protected]

Number of cells in y.

Definition at line 75 of file parameters.hpp.

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

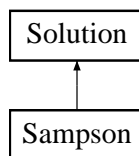
- [Headers/parameters.hpp](#)
- [Sources/parameters.cpp](#)

5.11 Sampson Class Reference

Computes Sampson solution.

```
#include <sampson.hpp>
```

Inheritance diagram for Sampson:



Public Member Functions

- [Sampson](#) ([Parameters](#) &)
Constructor.
- virtual [~Sampson](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- void [param](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.

5.11.1 Detailed Description

Computes Sampson solution.

Class that computes the solution for Sampson parabola with friction, see [Sampson et al. \[2006\]](#) [Sampson et al. \[2008\]](#).

Definition at line 70 of file sampson.hpp.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 `Sampson::Sampson (Parameters & par)`

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

<code>in</code>	<code>par</code>	contains all the values from the parameters
-----------------	------------------	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::T](#), [Solution::xex](#), [Solution::zex](#) to have Sampson configuration.

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

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

Destructor.

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

5.11.3 Member Function Documentation

5.11.3.1 `void Sampson::compute ()` [virtual]

Computes the solution.

Computes Sampson solution, see [Sampson et al. \[2006\]](#) [Sampson et al. \[2008\]](#).

Modifies

[Solution::hex](#), [Solution::uex](#).

Implements [Solution](#).

Definition at line 94 of file `sampson.cpp`.

5.11.3.2 `void Sampson::param (SCALAR L, SCALAR h0, SCALAR a, SCALAR B, SCALAR tau, SCALAR dx_ex, SCALAR T) const`

Writes the parameters of the solution.

Parameters

<code>in</code>	<code>L</code>	length of the domain
<code>in</code>	<code>h0</code>	value of the topography in the center of the domain
<code>in</code>	<code>a</code>	parameter of the topography
<code>in</code>	<code>B</code>	constant for the initial condition
<code>in</code>	<code>tau</code>	friction coefficient
<code>in</code>	<code>dx_ex</code>	space step
<code>in</code>	<code>T</code>	final time

Definition at line 121 of file sampson.cpp.

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

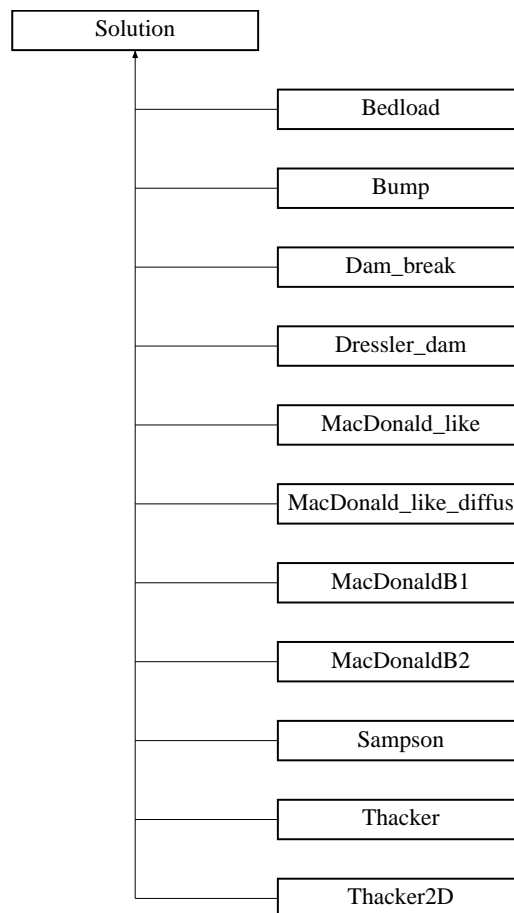
- Headers/[sampson.hpp](#)
- Sources/[sampson.cpp](#)

5.12 Solution Class Reference

Analytic solution.

```
#include <solution.hpp>
```

Inheritance diagram for Solution:



Public Member Functions

- [Solution](#) ([Parameters](#) &)
Constructor.
- void [allocation](#) ()
Allocations of the tables.
- void [desallocation](#) ()
Desallocation of the tables.
- virtual void [compute](#) ()=0
Function to be specified in case.

- void `savefinalcritical` (SCALAR *, SCALAR *, SCALAR *, SCALAR *) const
Saves the analytic solution at the final time with the critical height.
- void `savefinalcriticalinit` (SCALAR *, SCALAR *, SCALAR *, SCALAR *, SCALAR *) const
Saves the analytic solution at the final time with the critical height and the initial topography.
- void `savefinalmu` (SCALAR *, SCALAR *, SCALAR *) const
Saves the analytic solution at the final time without u.
- void `savefinal2D` (SCALAR *, SCALAR *, TAB, TAB, TAB, TAB) const
Saves the analytic solution at the final time in 2D.
- void `head` (Parameters &, string, string) const
Writes the version of the software and the choice of the solution.
- virtual `~Solution` ()
Destructor.

Protected Attributes

- const int `NX_EX`
- const int `NY_EX`
- SCALAR `T`
- SCALAR `L`
- SCALAR `I`
- SCALAR `dx_ex`
- SCALAR `dy_ex`
- SCALAR * `xex`
- SCALAR * `yex`
- SCALAR * `hex`
- SCALAR * `uex`
- SCALAR * `qex`
- SCALAR * `zex`

5.12.1 Detailed Description

Analytic solution.

Class that contains all the common declarations for the solutions.

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

5.12.2 Constructor & Destructor Documentation

5.12.2.1 `Solution::Solution (Parameters & par)`

Constructor.

Parameters

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

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

5.12.2.2 Solution::~~Solution() [virtual]

Destructor.

Definition at line 186 of file solution.cpp.

5.12.3 Member Function Documentation

5.12.3.1 void Solution::allocation()

Allocations of the tables.

Allocation of [Solution::xex](#), [Solution::yex](#), [Solution::hex](#), [Solution::uex](#), [Solution::qex](#), [Solution::zex](#).

Warning

Problem: allocation of xex failed.
 Problem: allocation of yex failed.
 Problem: allocation of hex failed.
 Problem: allocation of uex failed.
 Problem: allocation of qex failed.
 Problem: allocation of zex failed.

Note

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

Todo Exceptions should be treated.

Definition at line 191 of file solution.cpp.

5.12.3.2 virtual void Solution::compute() [pure virtual]

Function to be specified in case.

Implemented in [MacDonald_like](#), [Bump](#), [Bedload](#), [Dressler_dam](#), [MacDonald_like_diffus](#), [MacDonaldB2](#), [Sampson](#), [Dam_break](#), [MacDonaldB1](#), [Thacker](#), and [Thacker2D](#).

5.12.3.3 void Solution::desallocation()

Desallocation of the tables.

Desallocation of [Solution::xex](#), [Solution::yex](#), [Solution::hex](#), [Solution::uex](#), [Solution::qex](#), [Solution::zex](#).

Definition at line 244 of file solution.cpp.

5.12.3.4 void Solution::head(Parameters & par, string solutiontype, string solutionchoice) const

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

Parameters

in	<i>par</i>	parameter, contains all the values from the parameters file
in	<i>solutiontype</i>	name of the type of the solution
in	<i>solutionchoice</i>	name of the solution

Definition at line 166 of file solution.cpp.

5.12.3.5 void Solution::savefinal2D (SCALAR * *xex*, SCALAR * *yex*, TAB *hex2D*, TAB *uex2D*, TAB *vex2D*, TAB *zex2D*) const

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

Saves *x* and *y* (the position), *h* (the water height), *u* and *v* (the flow velocities in *x* and *y*), *z+h* (the free surface), *z* (the topography), *U* (the norm of the velocity), *Fr* (the Froude number), *qx*, *qy* and *q* (the flow discharge in *x*, *y* and its norm).

Parameters

in	<i>xex</i>	abscissae
in	<i>yex</i>	ordinates
in	<i>hex2D</i>	water height
in	<i>uex2D</i>	flow velocity in <i>x</i>
in	<i>vex2D</i>	flow velocity in <i>y</i>
in	<i>zex2D</i>	topography

Definition at line 135 of file solution.cpp.

5.12.3.6 void Solution::savefinalcritical (SCALAR * *xex*, SCALAR * *hex*, SCALAR * *uex*, SCALAR * *zex*) const

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

Saves *x* (the position), *h* (the water height), *u* (the flow velocity), *z* (the topography), *q* (the flow discharge), *z+h* (the free surface), *Fr* (the Froude number) and *z+hc* (the critical surface).

Parameters

in	<i>xex</i>	abscissae
in	<i>hex</i>	water height
in	<i>uex</i>	flow velocity
in	<i>zex</i>	topography

Definition at line 69 of file solution.cpp.

5.12.3.7 void Solution::savefinalcriticalinit (SCALAR * *xex*, SCALAR * *hex*, SCALAR * *uex*, SCALAR * *zex*, SCALAR * *z0*) const

Saves the analytic solution at the final time with the critical height and the initial topography.

Saves *x* (the position), *h* (the water height), *u* (the flow velocity), *z* (the topography), *q* (the flow discharge), *z+h* (the free surface), *Fr* (the Froude number), *z+hc* (the critical surface), *z0* (the initial topography) and *z0+h* (the initial surface).

Parameters

in	<i>xex</i>	abscissae
in	<i>hex</i>	water height
in	<i>uex</i>	flow velocity

in	<i>zex</i>	topography
in	<i>z0</i>	initial topography

Definition at line 93 of file solution.cpp.

5.12.3.8 void Solution::savefinalmu (SCALAR * *xex*, SCALAR * *hex*, SCALAR * *zex*) const

Saves the analytic solution at the final time without u.

Saves x (the position), h (the water height), z (the topography) and z+h (the free surface).

Parameters

in	<i>xex</i>	abscissae
in	<i>hex</i>	water height
in	<i>zex</i>	topography

Definition at line 118 of file solution.cpp.

5.12.4 Member Data Documentation

5.12.4.1 SCALAR Solution::dx_ex [protected]

Space step in x.

Definition at line 83 of file solution.hpp.

5.12.4.2 SCALAR Solution::dy_ex [protected]

Space step in y.

Definition at line 85 of file solution.hpp.

5.12.4.3 SCALAR* Solution::hex [protected]

Array for the water height.

Definition at line 92 of file solution.hpp.

5.12.4.4 SCALAR Solution::L [protected]

Dimensions of the domain in x.

Definition at line 79 of file solution.hpp.

5.12.4.5 SCALAR Solution::l [protected]

Dimensions of the domain in y.

Definition at line 81 of file solution.hpp.

5.12.4.6 `const int Solution::NX_EX` [protected]

Number of cells in x.

Definition at line 72 of file solution.hpp.

5.12.4.7 `const int Solution::NY_EX` [protected]

Number of cells in y.

Definition at line 74 of file solution.hpp.

5.12.4.8 `SCALAR* Solution::qex` [protected]

Array for the flow discharge.

Definition at line 96 of file solution.hpp.

5.12.4.9 `SCALAR Solution::T` [protected]

Final time.

Definition at line 77 of file solution.hpp.

5.12.4.10 `SCALAR* Solution::uex` [protected]

Array for the flow velocity.

Definition at line 94 of file solution.hpp.

5.12.4.11 `SCALAR* Solution::xex` [protected]

Array for the first coordinate.

Definition at line 88 of file solution.hpp.

5.12.4.12 `SCALAR* Solution::yex` [protected]

Array for the second coordinate.

Definition at line 90 of file solution.hpp.

5.12.4.13 `SCALAR* Solution::zex` [protected]

Array for the topography.

Definition at line 98 of file solution.hpp.

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

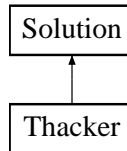
- Headers/[solution.hpp](#)
- Sources/[solution.cpp](#)

5.13 Thacker Class Reference

Computes Thacker solution.

```
#include <thacker.hpp>
```

Inheritance diagram for Thacker:



Public Member Functions

- [Thacker](#) ([Parameters](#) &)
Constructor.
- virtual [~Thacker](#) ()
Destructor.
- void [compute](#) ()
Computes the solution.
- void [param](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const
Writes the parameters of the solution.

5.13.1 Detailed Description

Computes Thacker solution.

Class that computes the solution for Thacker parabola, see [Thacker](#) [1981].

Definition at line 69 of file thacker.hpp.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Thacker::Thacker (Parameters & par)

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

in	par	contains all the values from the parameters
----	-----	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::T](#), [Solution::xex](#), [Solution::zex](#) to have Thacker configuration.

Definition at line 58 of file thacker.cpp.

5.13.2.2 Thacker::~Thacker () [virtual]

Destructor.

Definition at line 88 of file thacker.cpp.

5.13.3 Member Function Documentation

5.13.3.1 void Thacker::compute () [virtual]

Computes the solution.

Computes Thacker solution, see [Thacker \[1981\]](#).

Modifies

[Solution::hex](#), [Solution::uex](#).

Implements [Solution](#).

Definition at line 93 of file thacker.cpp.

5.13.3.2 void Thacker::param (SCALAR L, SCALAR h0, SCALAR a, SCALAR dx_ex, SCALAR T) const

Writes the parameters of the solution.

Parameters

in	L	length of the domain
in	$h0$	value of the topography in the center of the domain
in	a	parameter of the topography
in	dx_ex	space step
in	T	final time

Definition at line 122 of file thacker.cpp.

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

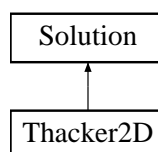
- Headers/[thacker.hpp](#)
- Sources/[thacker.cpp](#)

5.14 Thacker2D Class Reference

Computes Thacker solutions in 2D.

```
#include <thacker2d.hpp>
```

Inheritance diagram for Thacker2D:



Public Member Functions

- [Thacker2D \(Parameters &\)](#)

Constructor.

- virtual `~Thacker2D ()`

Destructor.

- void `compute ()`

Computes the solution.

- void `param (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR) const`

Writes the parameters of the solution.

5.14.1 Detailed Description

Computes Thacker solutions in 2D.

Class that computes the solutions for Thacker paraboloid, see [Thacker \[1981\]](#).

Definition at line 69 of file `thacker2d.hpp`.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Thacker2D::Thacker2D (Parameters & *par*)

Constructor.

Defines the physical parameters, the final time and prints the header with the configuration.

Parameters

<code>in</code>	<code>par</code>	contains all the values from the parameters
-----------------	------------------	---

Modifies

[Solution::dx_ex](#), [Solution::L](#), [Solution::l](#), [Solution::T](#), [Solution::xex](#), [Solution::yex](#), [Thacker2D::zex2D](#) to have Thacker 2D configuration.

Definition at line 58 of file `thacker2d.cpp`.

5.14.2.2 Thacker2D::~Thacker2D () [virtual]

Destructor.

Definition at line 131 of file `thacker2d.cpp`.

5.14.3 Member Function Documentation

5.14.3.1 void Thacker2D::compute () [virtual]

Computes the solution.

Computes the chosen Thacker 2D solution, see [Thacker \[1981\]](#).

Modifies

[Thacker2D::hex2D](#), [Thacker2D::uex2D](#), [Thacker2D::vex2D](#).

Implements [Solution](#).

Definition at line 147 of file `thacker2d.cpp`.

5.14.3.2 void Thacker2D::param (SCALAR L , SCALAR l , SCALAR $h0$, SCALAR a , SCALAR dx_ex , SCALAR dy_ex , SCALAR T) const

Writes the parameters of the solution.

Parameters

in	L	length of the domain in x
in	l	length of the domain in y
in	$h0$	value of the topography in the center of the domain
in	a	parameter of the topography
in	dx_ex	space step in x
in	dy_ex	space step in y
in	T	final time

Definition at line 184 of file thacker2d.cpp.

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

- Headers/[thacker2d.hpp](#)
- Sources/[thacker2d.cpp](#)

Chapter 6

File Documentation

6.1 Headers/bedload.hpp File Reference

Computes solutions with bedload.

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

Classes

- class [Bedload](#)
Computes solutions with bedload.

Defines

- #define [BEDLOAD_HPP](#)

6.1.1 Detailed Description

Computes solutions with bedload.

Author

Minh Hoang Le lemhoang@math.cnrs.fr (2012)
Carine Lucas carine.lucas@univ-orleans.fr (2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: the bed is moving with bedload, see [Berthon et al. \[2012\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [bedload.hpp](#).

6.1.2 Define Documentation

6.1.2.1 #define BEDLOAD_HPP

Definition at line 61 of file [bedload.hpp](#).

6.2 Headers/bump.hpp File Reference

Computes bumps solutions.

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

Classes

- class [Bump](#)

Computes bump solutions.

6.2.1 Detailed Description

Computes bumps solutions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Anne-Celine Boulanger anne-celine.boulanger@inria.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: with a bump, see [Delestre \[2010\]](#), [Goutal and Maurel \[1997\]](#).

Copyright

License Cecill-V2

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

(c) CNRS - Universite d'Orleans - INRA - Universite Pierre et Marie Curie (France)

Definition in file [bump.hpp](#).

6.3 Headers/choice_solution.hpp File Reference

Choice of the solution.

```
#include "solution.hpp"
#include "dam_break.hpp"
#include "dressler_dam.hpp"
#include "bump.hpp"
#include "macdonald_like.hpp"
#include "macdonald_like_diffus.hpp"
#include "thacker.hpp"
#include "bedload.hpp"
#include "thacker2d.hpp"
#include "macdonaldb1.hpp"
#include "macdonaldb2.hpp"
#include "sampson.hpp"
```

Classes

- class [Choice_solution](#)
Choice of the solution.

Defines

- #define [CHOICE_SOLUTION_HPP](#)

6.3.1 Detailed Description

Choice of the solution.

Author

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

Version

1.01.01

Date

2012-03-07

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

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_solution.hpp](#).

6.3.2 Define Documentation

6.3.2.1 #define CHOICE_SOLUTION_HPP

Definition at line 108 of file choice_solution.hpp.

6.4 Headers/dam_break.hpp File Reference

Computes dam break solutions.

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

Classes

- class [Dam_break](#)
Computes dam break solutions.

6.4.1 Detailed Description

Computes dam break solutions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: dam break without friction, see [Ritter \[1892\]](#) [Stoker \[1957\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [dam_break.hpp](#).

6.5 Headers/dressler_dam.hpp File Reference

Computes Dressler dam break solution.

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

Classes

- class [Dressler_dam](#)
Computes Dressler dam break solution.

6.5.1 Detailed Description

Computes Dressler dam break solution.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: dam break with friction, see [Dressler \[1952\]](#).

Copyright

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

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

Definition in file [dressler_dam.hpp](#).

6.6 Headers/macdonald_like.hpp File Reference

Computes Mac Donald solutions.

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

Classes

- class [MacDonald_like](#)
Computes Mac Donald solutions.

6.6.1 Detailed Description

Computes Mac Donald solutions.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald solutions in 1d, see [MacDonald \[1996\]](#) [MacDonald et al. \[1997\]](#) [Delestre \[2010\]](#) [Vo \[2008\]](#).

Copyright

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

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

Definition in file [macdonald_like.hpp](#).

6.7 Headers/macdonald_like_diffus.hpp File Reference

Computes Mac Donald solutions with diffusion.

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

Classes

- class [MacDonald_like_diffus](#)
Computes Mac Donald solutions with diffusion.

6.7.1 Detailed Description

Computes Mac Donald solutions with diffusion.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald solutions in 1d with diffusion, see [Delestre and Marche \[2010\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [macdonald_like_diffus.hpp](#).

6.8 Headers/macdonaldb1.hpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

Classes

- class [MacDonaldB1](#)
Computes Mac Donald pseudo 2d solutions.

6.8.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald pseudo 2d solutions with bottom B1, see [MacDonald \[1996\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [macdonaldb1.hpp](#).

6.9 Headers/macdonaldb2.hpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

Classes

- class [MacDonaldB2](#)
Computes Mac Donald pseudo 2d solutions.

6.9.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald pseudo 2d solutions with bottom B2, see [MacDonald](#) [1996].

Copyright

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

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

Definition in file [macdonaldb2.hpp](#).

6.10 Headers/misc.hpp File Reference

Definitions.

```
#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 [GRAV](#) 9.81
- #define [GRAV_DEM](#) 4.905
- #define [PI](#) 3.14159265
- #define [EPSILON_H](#) 1.e-12
- #define [VERSION](#) " SWASHES version 1.01.02, 2012-03-09"

Typedefs

- typedef double [SCALAR](#)
- typedef vector< vector< [SCALAR](#) > > [TAB](#)

6.10.1 Detailed Description

Definitions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.02

Date

2012-03-09

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](#).

6.10.2 Define Documentation

6.10.2.1 #define [EPSILON_H](#) 1.e-12

Definition at line 70 of file [misc.hpp](#).

6.10.2.2 #define [GRAV](#) 9.81

Definition at line 67 of file [misc.hpp](#).

6.10.2.3 #define GRAV_DEM 4.905

Definition at line 68 of file misc.hpp.

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

Definition at line 65 of file misc.hpp.

6.10.2.5 #define PI 3.14159265

Definition at line 69 of file misc.hpp.

6.10.2.6 #define VERSION " SWASHES version 1.01.02, 2012-03-09"

Definition at line 72 of file misc.hpp.

6.10.3 Typedef Documentation

6.10.3.1 typedef double SCALAR

Definition at line 76 of file misc.hpp.

6.10.3.2 typedef vector< vector< SCALAR > > TAB

Definition at line 77 of file misc.hpp.

6.11 Headers/parameters.hpp File Reference

Gets parameters.

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

Classes

- class [Parameters](#)
Gets parameters.

6.11.1 Detailed Description

Gets parameters.

Author

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

Version

1.01.01

Date

2012-03-07

Reads the parameters, checks their values, returns the use if needed.

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](#).

6.12 Headers/sampson.hpp File Reference

Computes Sampson solution.

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

Classes

- class [Sampson](#)
Computes Sampson solution.

6.12.1 Detailed Description

Computes Sampson solution.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Sampson parabola with friction, see [Sampson et al. \[2006\]](#) [Sampson et al. \[2008\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [sampson.hpp](#).

6.13 Headers/solution.hpp File Reference

Common file.

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

Classes

- class [Solution](#)
Analytic solution.

6.13.1 Detailed Description

Common file.

Author

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

Version

1.01.01

Date

2012-03-07

Common part for all the solutions.

Copyright

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

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

Definition in file [solution.hpp](#).

6.14 Headers/thacker.hpp File Reference

Computes Thacker solution.

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

Classes

- class [Thacker](#)
Computes Thacker solution.

6.14.1 Detailed Description

Computes Thacker solution.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Thacker parabola, see [Thacker \[1981\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [thacker.hpp](#).

6.15 Headers/thacker2d.hpp File Reference

Computes Thacker solutions in 2D.

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

Classes

- class [Thacker2D](#)
Computes Thacker solutions in 2D.

6.15.1 Detailed Description

Computes Thacker solutions in 2D.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Thacker paraboloid, see [Thacker \[1981\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [thacker2d.hpp](#).

6.16 Sources/bedload.cpp File Reference

Computes solutions with bedload.

```
#include "bedload.hpp"
```

6.16.1 Detailed Description

Computes solutions with bedload.

Author

Minh Hoang Le lemhoang@math.cnrs.fr (2012)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: the bed is moving with bedload, see [Berthon et al. \[2012\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [bedload.cpp](#).

6.17 Sources/bump.cpp File Reference

Computes bumps solutions.

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

6.17.1 Detailed Description

Computes bumps solutions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)
Anne-Celine Boulanger anne-celine.boulanger@inria.fr (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: with a bump, see [Delestre \[2010\]](#), [Goutal and Maurel \[1997\]](#).

Copyright

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

(c) CNRS - Universite d'Orleans - INRA - Universite Pierre et Marie Curie (France)

Definition in file [bump.cpp](#).

6.18 Sources/choice_solution.cpp File Reference

Choice of the solution.

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

6.18.1 Detailed Description

Choice of the solution.

Author

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

Version

1.01.01

Date

2012-03-06

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

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_solution.cpp](#).

6.19 Sources/dam_break.cpp File Reference

Computes dam break solutions.

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

6.19.1 Detailed Description

Computes dam break solutions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: dam break without friction, see [Ritter \[1892\]](#) [Stoker \[1957\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [dam_break.cpp](#).

6.20 Sources/dressler_dam.cpp File Reference

Computes Dressler dam break solution.

```
#include "dressler_dam.hpp"
```


6.20.1 Detailed Description

Computes Dressler dam break solution.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: dam break with friction, see [Dressler \[1952\]](#).

Copyright

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

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

Definition in file [dressler_dam.cpp](#).

6.21 Sources/macdonald_like.cpp File Reference

Computes Mac Donald solutions.

```
#include "macdonald_like.hpp"
```

6.21.1 Detailed Description

Computes Mac Donald solutions.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald solutions in 1d, see [MacDonald \[1996\]](#) [MacDonald et al. \[1997\]](#) [Delestre \[2010\]](#) [Vo \[2008\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [macdonald_like.cpp](#).

6.22 Sources/macdonald_like_diffus.cpp File Reference

Computes Mac Donald solutions with diffusion.

```
#include "macdonald_like_diffus.hpp"
```

6.22.1 Detailed Description

Computes Mac Donald solutions with diffusion.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald solutions in 1d with diffusion, see [Delestre and Marche \[2010\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [macdonald_like_diffus.cpp](#).

6.23 Sources/macdonaldb1.cpp File Reference

Computes Mac Donald pseudo 2d solutions.

```
#include "macdonaldb1.hpp"
```

6.23.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald pseudo 2d solutions with bottom B1, see [MacDonald \[1996\]](#).

Copyright

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

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

Definition in file [macdonaldb1.cpp](#).

6.24 Sources/macdonaldb2.cpp File Reference

Computes Mac Donald pseudo 2d solutions.

```
#include "macdonaldb2.hpp"
```

6.24.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: Mac Donald pseudo 2d solutions with bottom B2, see [MacDonald \[1996\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [macdonaldb2.cpp](#).

6.25 Sources/parameters.cpp File Reference

Gets parameters.

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

6.25.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-06

Reads the parameters, checks their values, returns the use if needed.

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](#).

6.26 Sources/sampson.cpp File Reference

Computes Sampson solution.

```
#include "sampson.hpp"
```

6.26.1 Detailed Description

Computes Sampson solution.

Author

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Sampson parabola with friction, see [Sampson et al. \[2006\]](#) [Sampson et al. \[2008\]](#).

Copyright

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

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

Definition in file [sampson.cpp](#).

6.27 Sources/solution.cpp File Reference

Common file.

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

6.27.1 Detailed Description

Common file.

Author

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

Version

1.01.02

Date

2012-03-09

Common part for all the solutions.

Copyright

License Cecill-V2

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

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

Definition in file [solution.cpp](#).

6.28 Sources/swashes.cpp File Reference

Main file.

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

Functions

- int [main](#) (int argc, char **argv)

6.28.1 Detailed Description

Main file.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

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

Version

1.01.01

Date

2012-03-08

For more details, we refer to [Delestre et al. \[2011\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [swashes.cpp](#).

6.28.2 Function Documentation

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

Main function of SWASHES, see [Delestre et al. \[2011\]](#).

Parameters

<code>in</code>	<code>argc</code>	number of the arguments.
<code>in</code>	<code>argv</code>	value of the arguments.

Definition at line 58 of file swashes.cpp.

6.29 Sources/thacker.cpp File Reference

Computes Thacker solution.

```
#include "thacker.hpp"
```

6.29.1 Detailed Description

Computes Thacker solution.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Carine Lucas carine.lucas@univ-orleans.fr (201-2012)

Version

1.01.01

Date

2012-03-08

Analytic solution: Thacker parabola, see [Thacker \[1981\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [thacker.cpp](#).

6.30 Sources/thacker2d.cpp File Reference

Computes Thacker solution in 2D.

```
#include "thacker2d.hpp"
```

6.30.1 Detailed Description

Computes Thacker solution in 2D.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2011)

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

Version

1.01.01

Date

2012-03-08

Analytic solution: Thacker paraboloid, see [Thacker \[1981\]](#).

Copyright

License Cecill-V2

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

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

Definition in file [thacker2d.cpp](#).

Bibliography

- C. Berthon, S. Cordier, O. Delestre, and M. H. Le. An analytical solution of the Shallow Water system coupled to the Exner equation. *C. R. Acad. Sci. Paris, Ser. I*, 350(3–4):183–186, 2012. doi:[10.1016/j.crma.2012.01.007](https://doi.org/10.1016/j.crma.2012.01.007). URL <http://hal.archives-ouvertes.fr/hal-00648343>. 9, 10, 47, 60
- 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-00587197>. 12, 13, 21, 22, 48, 52, 61, 63
- O. Delestre and F. Marche. A numerical scheme for a viscous shallow water model with friction. *Journal of Scientific Computing*, pages 1–11, 2010. ISSN 0885-7474. doi:[10.1007/s10915-010-9393-y](https://doi.org/10.1007/s10915-010-9393-y). 24, 25, 53, 64
- O. Delestre, C. Lucas, P.-A. Ksinant, F. Darboux, C. Laguerre, T. N. T. Vo, F. James, and S. Cordier. SWASHES: a library of Shallow Water Analytic Solutions for Hydraulic and Environmental Studies. Submitted, 2011. URL <http://hal.archives-ouvertes.fr/hal-00628246>. 68
- 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. 19, 20, 51, 63
- N. Goutal and F. Maurel. Proceedings of the 2nd workshop on dam-break wave simulation. Technical Report HE-43/97/016/B, Electricité de France, Direction des études et recherches, 1997. 12, 13, 48, 61
- I. MacDonald. *Analysis and computation of steady open channel flow*. PhD thesis, University of Reading — Department of Mathematics, Sept. 1996. 21, 22, 26, 27, 29, 52, 53, 54, 63, 65
- I. MacDonald, M. J. Baines, N. K. Nichols, and P. G. Samuels. Analytic benchmark solutions for open-channel flows. *Journal of Hydraulic Engineering*, 123(11):1041–1045, Nov. 1997. 21, 22, 52, 63
- A. Ritter. Die Fortpflanzung der Wasserwellen. *Zeitschrift des Vereines Deutscher Ingenieure*, 36(33):947–954, 1892. 17, 18, 50, 62
- J. Sampson, A. Easton, and M. Singh. Moving boundary shallow water flow above parabolic bottom topography. In A. Stacey, B. Blyth, J. Shepherd, and A. J. Roberts, editors, *Proceedings of the 7th Biennial Engineering Mathematics and Applications Conference, EMAC-2005*, volume 47 of *ANZIAM Journal*, pages C373–C387. Australian Mathematical Society, oct 2006. URL <http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/1050>. 34, 35, 57, 67
- J. Sampson, A. Easton, and M. Singh. Moving boundary shallow water flow in a region with quadratic bathymetry. In G. N. Mercer and A. J. Roberts, editors, *Proceedings of the 8th Biennial Engineering Mathematics and Applications Conference, EMAC-2007*, volume 49 of *ANZIAM Journal*, pages C666–C680. Australian Mathematical Society, 2008. URL <http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/306>. 34, 35, 57, 67
- J. J. Stoker. *Water Waves: The Mathematical Theory with Applications*. Pure and Applied Mathematics. Interscience Publishers, New York, USA, 1957. 17, 18, 50, 62
- W. C. Thacker. Some exact solutions to the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508, 1981. doi:[10.1017/S0022112081001882](https://doi.org/10.1017/S0022112081001882). URL <http://>

[//journals.cambridge.org/action/displayAbstract?fromPage=online&aid=389055&fulltextType=RA&fileId=S0022112081001882](http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=389055&fulltextType=RA&fileId=S0022112081001882). 42, 43, 44, 59, 60, 69, 70

T. N. T. Vo. One dimensional Saint-Venant system. Master's thesis, Université d'Orléans, France, June 2008.
URL <http://dumas.ccsd.cnrs.fr/dumas-00597434>. 21, 22, 52, 63

Index

- ~Bedload
 - Bedload, [10](#)
- ~Bump
 - Bump, [12](#)
- ~Choice_solution
 - Choice_solution, [16](#)
- ~Dam_break
 - Dam_break, [18](#)
- ~Dressler_dam
 - Dressler_dam, [20](#)
- ~MacDonaldB1
 - MacDonaldB1, [27](#)
- ~MacDonaldB2
 - MacDonaldB2, [29](#)
- ~MacDonald_like
 - MacDonald_like, [22](#)
- ~MacDonald_like_diffus
 - MacDonald_like_diffus, [24](#)
- ~Parameters
 - Parameters, [31](#)
- ~Sampson
 - Sampson, [35](#)
- ~Solution
 - Solution, [37](#)
- ~Thacker
 - Thacker, [42](#)
- ~Thacker2D
 - Thacker2D, [44](#)
- abcd
 - Bump, [13](#)
- allocation
 - Solution, [37](#)
- BEDLOAD_HPP
 - bedload.hpp, [48](#)
- Bedload, [9](#)
 - ~Bedload, [10](#)
 - Bedload, [10](#)
 - compute, [10](#)
 - param, [10](#)
 - paramwarning, [11](#)
- bedload.hpp
 - BEDLOAD_HPP, [48](#)
- Bump, [11](#)
 - ~Bump, [12](#)
- abcd, [13](#)
- Bump, [12](#)
- compute, [13](#)
- determinant, [13](#)
- height, [14](#)
- p, [14](#)
- param, [14](#)
- q, [15](#)
- RHJump, [15](#)
- choice
 - Parameters, [33](#)
- Choice_solution, [15](#)
 - ~Choice_solution, [16](#)
 - Choice_solution, [16](#)
 - Choice_solution, [16](#)
 - compute, [16](#)
- choicedim
 - Parameters, [33](#)
- choicedomain
 - Parameters, [33](#)
- choicetype
 - Parameters, [33](#)
- compute
 - Bedload, [10](#)
 - Bump, [13](#)
 - Choice_solution, [16](#)
 - Dam_break, [18](#)
 - Dressler_dam, [20](#)
 - MacDonald_like, [22](#)
 - MacDonald_like_diffus, [25](#)
 - MacDonaldB1, [27](#)
 - MacDonaldB2, [29](#)
 - Sampson, [35](#)
 - Solution, [38](#)
 - Thacker, [42](#)
 - Thacker2D, [44](#)
- Dam_break, [17](#)
 - ~Dam_break, [18](#)
 - compute, [18](#)
 - Dam_break, [17](#)
 - Dam_break, [17](#)
 - function, [18](#)
 - param, [18](#)
- Delta_topo

- MacDonaldB1, [27](#)
- MacDonaldB2, [29](#)
- Delta_topo_Darcy_Weisbach
 - MacDonald_like, [22](#)
- Delta_topo_Manning
 - MacDonald_like, [23](#)
- Delta_topo_diffus
 - MacDonald_like_diffus, [25](#)
- desallocation
 - Solution, [38](#)
- determinant
 - Bump, [13](#)
- Dressler_dam, [19](#)
 - ~Dressler_dam, [20](#)
 - compute, [20](#)
 - Dressler_dam, [19](#)
 - Dressler_dam, [19](#)
 - param, [20](#)
- dx_ex
 - Solution, [40](#)
- dy_ex
 - Solution, [40](#)
- EPSILON_H
 - misc.hpp, [55](#)
- function
 - Dam_break, [18](#)
- GRAV
 - misc.hpp, [55](#)
- GRAV_DEM
 - misc.hpp, [55](#)
- get_choice
 - Parameters, [32](#)
- get_choicedim
 - Parameters, [32](#)
- get_choicedomain
 - Parameters, [32](#)
- get_choicetype
 - Parameters, [32](#)
- get_nxex
 - Parameters, [32](#)
- get_nyex
 - Parameters, [32](#)
- head
 - Solution, [38](#)
- Headers/bedload.hpp, [47](#)
- Headers/bump.hpp, [48](#)
- Headers/choice_solution.hpp, [49](#)
- Headers/dam_break.hpp, [50](#)
- Headers/dressler_dam.hpp, [50](#)
- Headers/macdonald_like.hpp, [51](#)
- Headers/macdonald_like_diffus.hpp, [52](#)
- Headers/macdonaldb1.hpp, [53](#)
- Headers/macdonaldb2.hpp, [54](#)
- Headers/misc.hpp, [54](#)
- Headers/parameters.hpp, [56](#)
- Headers/sampson.hpp, [57](#)
- Headers/solution.hpp, [58](#)
- Headers/thacker.hpp, [58](#)
- Headers/thacker2d.hpp, [59](#)
- height
 - Bump, [14](#)
- help
 - Parameters, [33](#)
- hex
 - Solution, [40](#)
- L
 - Solution, [40](#)
- I
 - Solution, [40](#)
- MAX
 - misc.hpp, [56](#)
- MacDonald_like, [21](#)
 - ~MacDonald_like, [22](#)
 - compute, [22](#)
 - Delta_topo_Darcy_Weisbach, [22](#)
 - Delta_topo_Manning, [23](#)
 - MacDonald_like, [21](#)
 - MacDonald_like, [21](#)
 - param, [23](#)
- MacDonald_like_diffus, [23](#)
 - ~MacDonald_like_diffus, [24](#)
 - compute, [25](#)
 - Delta_topo_diffus, [25](#)
 - MacDonald_like_diffus, [24](#)
 - MacDonald_like_diffus, [24](#)
 - param, [25](#)
- MacDonaldB1, [26](#)
 - ~MacDonaldB1, [27](#)
 - compute, [27](#)
 - Delta_topo, [27](#)
 - MacDonaldB1, [26](#)
 - MacDonaldB1, [26](#)
 - param, [27](#)
- MacDonaldB2, [28](#)
 - ~MacDonaldB2, [29](#)
 - compute, [29](#)
 - Delta_topo, [29](#)
 - MacDonaldB2, [29](#)
 - MacDonaldB2, [29](#)
 - param, [30](#)
- main
 - swashes.cpp, [68](#)
- misc.hpp

- EPSILON_H, [55](#)
 - GRAV, [55](#)
 - GRAV_DEM, [55](#)
 - MAX, [56](#)
 - PI, [56](#)
 - SCALAR, [56](#)
 - TAB, [56](#)
 - VERSION, [56](#)
- NX_EX
 - Solution, [40](#)
- NY_EX
 - Solution, [40](#)
- nx_ex
 - Parameters, [33](#)
- ny_ex
 - Parameters, [33](#)
- p
 - Bump, [14](#)
- PI
 - misc.hpp, [56](#)
- param
 - Bedload, [10](#)
 - Bump, [14](#)
 - Dam_break, [18](#)
 - Dressler_dam, [20](#)
 - MacDonald_like, [23](#)
 - MacDonald_like_diffus, [25](#)
 - MacDonaldB1, [27](#)
 - MacDonaldB2, [30](#)
 - Sampson, [35](#)
 - Thacker, [43](#)
 - Thacker2D, [44](#)
- Parameters, [30](#)
 - ~Parameters, [31](#)
 - choice, [33](#)
 - choicedim, [33](#)
 - choicedomain, [33](#)
 - choicetype, [33](#)
 - get_choice, [32](#)
 - get_choicedim, [32](#)
 - get_choicedomain, [32](#)
 - get_choicetype, [32](#)
 - get_nxex, [32](#)
 - get_nyex, [32](#)
 - help, [33](#)
 - nx_ex, [33](#)
 - ny_ex, [33](#)
 - Parameters, [31](#)
- paramwarning
 - Bedload, [11](#)
- q
 - Bump, [15](#)
 - qex
 - Solution, [40](#)
 - RHJump
 - Bump, [15](#)
 - SCALAR
 - misc.hpp, [56](#)
 - Sampson, [34](#)
 - ~Sampson, [35](#)
 - compute, [35](#)
 - param, [35](#)
 - Sampson, [34](#)
 - savefinal2D
 - Solution, [38](#)
 - savefinalcritical
 - Solution, [39](#)
 - savefinalcriticalinit
 - Solution, [39](#)
 - savefinalmu
 - Solution, [39](#)
 - Solution, [35](#)
 - ~Solution, [37](#)
 - allocation, [37](#)
 - compute, [38](#)
 - deallocation, [38](#)
 - dx_ex, [40](#)
 - dy_ex, [40](#)
 - head, [38](#)
 - hex, [40](#)
 - L, [40](#)
 - l, [40](#)
 - NX_EX, [40](#)
 - NY_EX, [40](#)
 - qex, [40](#)
 - savefinal2D, [38](#)
 - savefinalcritical, [39](#)
 - savefinalcriticalinit, [39](#)
 - savefinalmu, [39](#)
 - Solution, [37](#)
 - T, [41](#)
 - uex, [41](#)
 - xex, [41](#)
 - yex, [41](#)
 - zex, [41](#)
 - Sources/bedload.cpp, [60](#)
 - Sources/bump.cpp, [60](#)
 - Sources/choice_solution.cpp, [61](#)
 - Sources/dam_break.cpp, [62](#)
 - Sources/dressler_dam.cpp, [62](#)
 - Sources/macdonald_like.cpp, [63](#)
 - Sources/macdonald_like_diffus.cpp, [64](#)
 - Sources/macdonaldb1.cpp, [64](#)

Sources/macdonaldb2.cpp, [65](#)
Sources/parameters.cpp, [66](#)
Sources/sampson.cpp, [66](#)
Sources/solution.cpp, [67](#)
Sources/swashes.cpp, [68](#)
Sources/thacker.cpp, [69](#)
Sources/thacker2d.cpp, [69](#)
swashes.cpp
 main, [68](#)

T

 Solution, [41](#)

TAB

 misc.hpp, [56](#)

Thacker, [41](#)

 ~Thacker, [42](#)

 compute, [42](#)

 param, [43](#)

 Thacker, [42](#)

Thacker2D, [43](#)

 ~Thacker2D, [44](#)

 compute, [44](#)

 param, [44](#)

 Thacker2D, [44](#)

 Thacker2D, [44](#)

uex

 Solution, [41](#)

VERSION

 misc.hpp, [56](#)

xex

 Solution, [41](#)

yex

 Solution, [41](#)

zex

 Solution, [41](#)