

Documentation  
of  
SWASHES

v1.02.00  
(2014-04-29)

Generated by Doxygen 1.8.2  
on Tue Apr 29 2014 15:04:05



# 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

# Hierarchical Index

### 2.1 Class Hierarchy

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

Choice_solution . . . . .	15
Parameters . . . . .	34
Solution . . . . .	42
Bedload . . . . .	9
Bump . . . . .	11
Dam_break . . . . .	17
Dressler_dam . . . . .	19
Inclined_plane . . . . .	21
MacDonald_like . . . . .	25
MacDonald_like_diffus . . . . .	27
MacDonaldB1 . . . . .	30
MacDonaldB2 . . . . .	32
Sampson . . . . .	38
Selfsimilar_dam_break . . . . .	40
Thacker . . . . .	47
Thacker2D . . . . .	49



# Chapter 3

## Class Index

### 3.1 Class List

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

<a href="#">Bedload</a>	Computes solutions with bedload . . . . .	9
<a href="#">Bump</a>	Computes bump solutions . . . . .	11
<a href="#">Choice_solution</a>	Choice of the solution . . . . .	15
<a href="#">Dam_break</a>	Computes dam break solutions . . . . .	17
<a href="#">Dressler_dam</a>	Computes Dressler dam break solution . . . . .	19
<a href="#">Inclined_plane</a>	Computes the solutions over an inclined plane . . . . .	21
<a href="#">MacDonald_like</a>	Computes Mac Donald solutions . . . . .	25
<a href="#">MacDonald_like_diffus</a>	Computes Mac Donald solutions with diffusion . . . . .	27
<a href="#">MacDonaldB1</a>	Computes Mac Donald pseudo 2d solutions . . . . .	30
<a href="#">MacDonaldB2</a>	Computes Mac Donald pseudo 2d solutions . . . . .	32
<a href="#">Parameters</a>	Gets parameters . . . . .	34
<a href="#">Sampson</a>	Computes Sampson solution . . . . .	38
<a href="#">Selfsimilar_dam_break</a>	Computes self-similar dam break solutions . . . . .	40
<a href="#">Solution</a>	Analytic solution . . . . .	42
<a href="#">Thacker</a>	Computes Thacker solution . . . . .	47
<a href="#">Thacker2D</a>	Computes Thacker solutions in 2D . . . . .	49





# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

Headers/ <a href="#">bedload.hpp</a>	
Computes solutions with bedload . . . . .	53
Headers/ <a href="#">bump.hpp</a>	
Computes bumps solutions . . . . .	54
Headers/ <a href="#">choice_solution.hpp</a>	
Choice of the solution . . . . .	55
Headers/ <a href="#">dam_break.hpp</a>	
Computes dam break solutions . . . . .	56
Headers/ <a href="#">dressler_dam.hpp</a>	
Computes Dressler dam break solution . . . . .	56
Headers/ <a href="#">inclined_plane.hpp</a>	
Computes the solution over an inclined plane . . . . .	57
Headers/ <a href="#">macdonald_like.hpp</a>	
Computes Mac Donald solutions . . . . .	58
Headers/ <a href="#">macdonald_like_diffus.hpp</a>	
Computes Mac Donald solutions with diffusion . . . . .	59
Headers/ <a href="#">macdonaldb1.hpp</a>	
Computes Mac Donald pseudo 2d solutions . . . . .	60
Headers/ <a href="#">macdonaldb2.hpp</a>	
Computes Mac Donald pseudo 2d solutions . . . . .	60
Headers/ <a href="#">misc.hpp</a>	
Definitions . . . . .	61
Headers/ <a href="#">parameters.hpp</a>	
Gets parameters . . . . .	63
Headers/ <a href="#">sampson.hpp</a>	
Computes Sampson solution . . . . .	64
Headers/ <a href="#">selfsimilar_dam_break.hpp</a>	
Computes self-similar dam break solutions . . . . .	64
Headers/ <a href="#">solution.hpp</a>	
Common file . . . . .	65
Headers/ <a href="#">thacker.hpp</a>	
Computes Thacker solution . . . . .	66
Headers/ <a href="#">thacker2d.hpp</a>	
Computes Thacker solutions in 2D . . . . .	67

Sources/ <a href="#">bedload.cpp</a>	
Computes solutions with bedload . . . . .	68
Sources/ <a href="#">bump.cpp</a>	
Computes bumps solutions . . . . .	68
Sources/ <a href="#">choice_solution.cpp</a>	
Choice of the solution . . . . .	69
Sources/ <a href="#">dam_break.cpp</a>	
Computes dam break solutions . . . . .	70
Sources/ <a href="#">dressler_dam.cpp</a>	
Computes Dressler dam break solution . . . . .	70
Sources/ <a href="#">inclined_plane.cpp</a>	
Computes the solution over an inclined plane . . . . .	71
Sources/ <a href="#">macdonald_like.cpp</a>	
Computes Mac Donald solutions . . . . .	72
Sources/ <a href="#">macdonald_like_diffus.cpp</a>	
Computes Mac Donald solutions with diffusion . . . . .	72
Sources/ <a href="#">macdonald1.cpp</a>	
Computes Mac Donald pseudo 2d solutions . . . . .	73
Sources/ <a href="#">macdonald2.cpp</a>	
Computes Mac Donald pseudo 2d solutions . . . . .	74
Sources/ <a href="#">parameters.cpp</a>	
Gets parameters . . . . .	74
Sources/ <a href="#">sampson.cpp</a>	
Computes Sampson solution . . . . .	75
Sources/ <a href="#">selfsimilar_dam_break.cpp</a>	
Computes self-similar dam break solutions . . . . .	76
Sources/ <a href="#">solution.cpp</a>	
Common file . . . . .	76
Sources/ <a href="#">swashes.cpp</a>	
Main file . . . . .	77
Sources/ <a href="#">thacker.cpp</a>	
Computes Thacker solution . . . . .	78
Sources/ <a href="#">thacker2d.cpp</a>	
Computes Thacker solution in 2D . . . . .	78

## Chapter 5

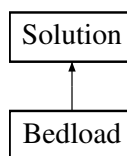
# Class Documentation

### 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.*

#### Additional Inherited Members

##### 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	$\rho$	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:

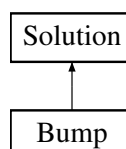
- Headers/[bedload.hpp](#)
- Sources/[bedload.cpp](#)

## 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.*

## Additional Inherited Members

### 5.2.1 Detailed Description

Computes bump solutions.

Class that computes the solutions with a bump for the topography, see [Delestre et al. \[2013\]](#) and [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 et al. \[2013\]](#) and [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 <a href="#">Bump::p</a>
in	<i>q</i>	computed by <a href="#">Bump::q</a>

**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 <a href="#">Bump::p</a>
in	<i>q</i>	computed by <a href="#">Bump::q</a>
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 125 of file `choice_solution.hpp`.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 `Choice_solution::Choice_solution ( Parameters & par )`

Constructor.

#### Parameters

<code>in</code>	<code>par</code>	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 594 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 590 of file choice\_solution.cpp.

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

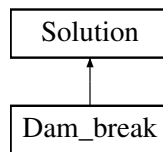
- Headers/[choice\\_solution.hpp](#)
- Sources/[choice\\_solution.cpp](#)

## 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.*

### Additional Inherited Members

#### 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	<i>par</i>	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:

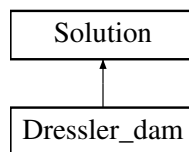
- Headers/dam\_break.hpp
- Sources/dam\_break.cpp

## 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.*

### Additional Inherited Members

#### 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

in	par	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](#)

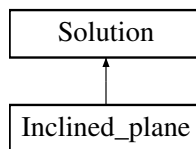
- Sources/[dressler\\_dam.cpp](#)

## 5.6 Inclined\_plane Class Reference

Computes the solutions over an inclined plane.

```
#include <inclined_plane.hpp>
```

Inheritance diagram for Inclined\_plane:



### Public Member Functions

- [Inclined\\_plane](#) ([Parameters](#) &)  
*Constructor.*
- virtual [~Inclined\\_plane](#) ()  
*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 polynomial roots.*
- void [abcd](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#) &, [SCALAR](#) &, [SCALAR](#) &, [SCALAR](#) &)  
*Defines a, b, c, d in order to solve  $ah^3 + bh^2 + ch + d$ .*
- void [param](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#)) const  
*Writes the parameters of the solution.*

### Additional Inherited Members

#### 5.6.1 Detailed Description

Computes the solutions over an inclined plane.

Class that computes the solutions over an inclined plane, see [Delestre et al. \[2012\]](#).

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

## 5.6.2 Constructor & Destructor Documentation

### 5.6.2.1 `Inclined_plane::Inclined_plane ( 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
-----------------	------------------	---

#### Modifies

[Solution::dx\\_ex](#), [Solution::L](#), [Solution::xex](#), [Solution::zex](#) to have the inclined plane configuration.

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

### 5.6.2.2 `Inclined_plane::~~Inclined_plane ( ) [virtual]`

Destructor.

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

## 5.6.3 Member Function Documentation

### 5.6.3.1 `void Inclined_plane::abcd ( SCALAR q_in, SCALAR h_in, SCALAR alpha, SCALAR x, 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

<code>in</code>	<code>q_in</code>	inflow discharge
<code>in</code>	<code>h_in</code>	water height at the inflow
<code>in</code>	<code>alpha</code>	the slope
<code>in</code>	<code>x</code>	the position
<code>out</code>	<code>a</code>	coefficient of the 3rd order polynomia
<code>out</code>	<code>b</code>	coefficient of the 3rd order polynomia
<code>out</code>	<code>c</code>	coefficient of the 3rd order polynomia
<code>out</code>	<code>d</code>	coefficient of the 3rd order polynomia

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

### 5.6.3.2 `void Inclined_plane::compute ( ) [virtual]`

Computes the solution.



Computes the solution on an inclined plane, see [Delestre et al. \[2012\]](#).

#### Modifies

[Solution::hex](#).

Implements [Solution](#).

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

#### 5.6.3.3 SCALAR `Inclined_plane::determinant ( SCALAR $p$ , SCALAR $q$ ) const`

Determinant for Cardano method.

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

#### Parameters

<code>in</code>	$p$	computed by <a href="#">Inclined_plane::p</a>
<code>in</code>	$q$	computed by <a href="#">Inclined_plane::q</a>

#### Returns

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

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

#### 5.6.3.4 SCALAR `Inclined_plane::height ( SCALAR $p$ , SCALAR $q$ , SCALAR $a$ , SCALAR $b$ , SCALAR $hnear$ ) const`

Computation of the 3rd order polynomia roots.

#### Parameters

<code>in</code>	$p$	computed by <a href="#">Inclined_plane::p</a>
<code>in</code>	$q$	computed by <a href="#">Inclined_plane::q</a>
<code>in</code>	$a$	coefficient of the 3rd order polynomia
<code>in</code>	$b$	coefficient of the 3rd order polynomia
<code>in</code>	$hnear$	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 169 of file `inclined_plane.cpp`.

### 5.6.3.5 SCALAR Inclined\_plane::p ( SCALAR a, SCALAR b, SCALAR c ) const

Coefficient p for Cardano method.

#### Parameters

in	a	coefficient of the 3rd order polynomia
in	b	coefficient of the 3rd order polynomia
in	c	coefficient of the 3rd order polynomia

#### Returns

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

Definition at line 126 of file inclined\_plane.cpp.

### 5.6.3.6 void Inclined\_plane::param ( SCALAR L, SCALAR dx\_ex, SCALAR alpha, SCALAR beta, SCALAR h0, SCALAR q0 ) const

Writes the parameters of the solution.

#### Parameters

in	L	length of the domain
in	dx_ex	space step
in	alpha	the slope of the plane
in	beta	the value of the topography for x=0
in	h0	the left (imposed) water height
in	q0	the left (imposed) water discharge

Definition at line 255 of file inclined\_plane.cpp.

### 5.6.3.7 SCALAR Inclined\_plane::q ( SCALAR a, SCALAR b, SCALAR c, SCALAR d ) const

Coefficient q for Cardano method.

#### Parameters

in	a	coefficient of the 3rd order polynomia
in	b	coefficient of the 3rd order polynomia
in	c	coefficient of the 3rd order polynomia
in	d	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 139 of file inclined\_plane.cpp.

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

- Headers/[inclined\\_plane.hpp](#)

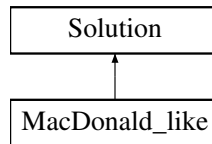
- Sources/[inclined\\_plane.cpp](#)

## 5.7 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.*

### Additional Inherited Members

#### 5.7.1 Detailed Description

Computes Mac Donald solutions.

Class that computes Mac Donald solutions in 1d, see [MacDonald \[1996\]](#), [MacDonald et al. \[1997\]](#), [Delestre et al. \[2013\]](#) and [Vo T. N. \[2008\]](#).

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

#### 5.7.2 Constructor & Destructor Documentation

##### 5.7.2.1 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.

### 5.7.2.2 MacDonald\_like::~MacDonald\_like ( ) [virtual]

Destructor.

Definition at line 435 of file macdonald\_like.cpp.

## 5.7.3 Member Function Documentation

### 5.7.3.1 void MacDonald\_like::compute ( ) [virtual]

Computes the solution.

Computes Mac Donald solutions, see [MacDonald \[1996\]](#), [MacDonald et al. \[1997\]](#), [Delestre et al. \[2013\]](#) and [Vo T. N. \[2008\]](#).

## Modifies

[Solution::zex](#).

Implements [Solution](#).

Definition at line 440 of file macdonald\_like.cpp.

### 5.7.3.2 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.

### 5.7.3.3 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.

### 5.7.3.4 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:

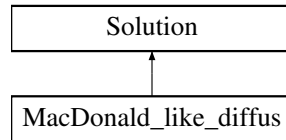
- Headers/[macdonald\\_like.hpp](#)
- Sources/[macdonald\\_like.cpp](#)

## 5.8 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

- [MacDonald\\_like\\_diffus](#) ([Parameters](#) &)  
*Constructor.*
- virtual [~MacDonald\\_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.*

## Additional Inherited Members

### 5.8.1 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`.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 `MacDonald_like_diffus::MacDonald_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`.

### 5.8.2.2 MacDonald\_like\_diffus::~MacDonald\_like\_diffus ( ) [virtual]

Destructor.

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

## 5.8.3 Member Function Documentation

### 5.8.3.1 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`.

### 5.8.3.2 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	$q$	discharge
in	$h$	water height
in	$dh$	variation of the water height
in	$ddh$	second order derivative of h
in	$kt$	turbulent coefficient
in	$kl$	laminar coefficient
in	$muv$	vertical viscosity
in	$muh$	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`.

5.8.3.3 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:

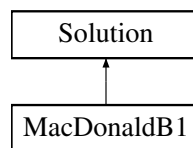
- Headers/[macdonald\\_like\\_diffus.hpp](#)
- Sources/[macdonald\\_like\\_diffus.cpp](#)

## 5.9 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.*

#### Additional Inherited Members

##### 5.9.1 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.



## 5.9.2 Constructor & Destructor Documentation

### 5.9.2.1 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.

### 5.9.2.2 MacDonaldB1::~MacDonaldB1 ( ) [virtual]

Destructor.

Definition at line 227 of file macdonaldb1.cpp.

## 5.9.3 Member Function Documentation

### 5.9.3.1 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.

### 5.9.3.2 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.

### 5.9.3.3 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:

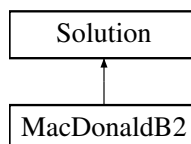
- Headers/[macdonaldb1.hpp](#)
- Sources/[macdonaldb1.cpp](#)

## 5.10 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.*

## Additional Inherited Members

### 5.10.1 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.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 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.

#### 5.10.2.2 MacDonaldB2::~~MacDonaldB2 ( ) [virtual]

Destructor.

Definition at line 197 of file macdonaldb2.cpp.

### 5.10.3 Member Function Documentation

#### 5.10.3.1 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.

#### 5.10.3.2 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.

## 5.10.3.3 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](#)

## 5.11 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](#)

#### 5.11.1 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.

#### 5.11.2 Constructor & Destructor Documentation

##### 5.11.2.1 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.

**5.11.2.2 Parameters::~Parameters ( ) [virtual]**

Destructor.

Definition at line 110 of file parameters.cpp.

**5.11.3 Member Function Documentation****5.11.3.1 int Parameters::get\_choice ( ) const**

Gives the chosen solution.

**Returns**

The chosen solution.

Definition at line 219 of file parameters.cpp.

**5.11.3.2 SCALAR Parameters::get\_choicedim ( ) const**

Gives the dimension.

**Returns**

The dimension of the solution.

Definition at line 229 of file parameters.cpp.

**5.11.3.3 int Parameters::get\_choicedomain ( ) const**

Gives the domain.

**Returns**

The domain of the solution.

Definition at line 249 of file parameters.cpp.

**5.11.3.4 int Parameters::get\_choicetype ( ) const**

Gives the type.

**Returns**

The type of the solution.

Definition at line 239 of file parameters.cpp.

#### 5.11.3.5 `int Parameters::get_nxex ( ) const`

Gives the number of cells in x.

##### Returns

The number of cells in x.

Definition at line 199 of file parameters.cpp.

#### 5.11.3.6 `int Parameters::get_nyex ( ) const`

Gives the number of cells in y.

##### Returns

The number of cells in y.

Definition at line 209 of file parameters.cpp.

#### 5.11.3.7 `void Parameters::help ( ) const`

Prints help.

Prints how to use the code.

Definition at line 113 of file parameters.cpp.

### 5.11.4 Member Data Documentation

#### 5.11.4.1 `int Parameters::choice` [protected]

Value corresponding to the chosen solution.

Definition at line 81 of file parameters.hpp.

#### 5.11.4.2 **SCALAR** `Parameters::choicedim` [protected]

Value corresponding to the dimension of the solution.

Definition at line 77 of file parameters.hpp.

#### 5.11.4.3 `int Parameters::choicedomain` [protected]

Value corresponding to the domain of the solution.

Definition at line 83 of file parameters.hpp.

#### 5.11.4.4 `int Parameters::choicetype` [protected]

Value corresponding to the type of the solution.

Definition at line 79 of file parameters.hpp.

#### 5.11.4.5 `int Parameters::nx_ex` [protected]

Number of cells in x.

Definition at line 73 of file parameters.hpp.

#### 5.11.4.6 `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:

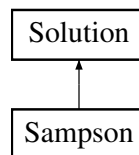
- Headers/[parameters.hpp](#)
- Sources/[parameters.cpp](#)

## 5.12 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.*

### Additional Inherited Members

#### 5.12.1 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.



### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 `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`.

#### 5.12.2.2 `Sampson::~~Sampson ( ) [virtual]`

Destructor.

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

### 5.12.3 Member Function Documentation

#### 5.12.3.1 `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`.

#### 5.12.3.2 `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:

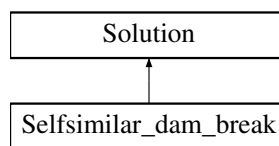
- Headers/[sampson.hpp](#)
- Sources/[sampson.cpp](#)

## 5.13 Selfsimilar\_dam\_break Class Reference

Computes self-similar dam break solutions.

```
#include <selfsimilar_dam_break.hpp>
```

Inheritance diagram for Selfsimilar\_dam\_break:



### Public Member Functions

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

### Additional Inherited Members

#### 5.13.1 Detailed Description

Computes self-similar dam break solutions.

Class that computes the self-similar solutions for dam break with friction, see Self-similar\_solutions.pdf in the doc folder or in the bibliography of sourcesup.

Definition at line 71 of file selfsimilar\_dam\_break.hpp.

#### 5.13.2 Constructor & Destructor Documentation

##### 5.13.2.1 Selfsimilar\_dam\_break::Selfsimilar\_dam\_break ( [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
-----------	------------	---

## Modifies

[Solution::dx\\_ex](#), [Solution::L](#), [Solution::T](#), [Solution::xex](#), [Solution::zex](#) to have the self-similar dam break configuration.

Definition at line 59 of file selfsimilar\_dam\_break.cpp.

### 5.13.2.2 Selfsimilar\_dam\_break::~Selfsimilar\_dam\_break ( ) [virtual]

Destructor.

Definition at line 138 of file selfsimilar\_dam\_break.cpp.

## 5.13.3 Member Function Documentation

### 5.13.3.1 void Selfsimilar\_dam\_break::compute ( ) [virtual]

Computes the solution.

Computes the chosen self-similar dam break solution,

see Self-similar\_solutions.pdf in the doc folder or in the bibliography of sourcesup.

## Modifies

[Solution::hex](#).

Implements [Solution](#).

Definition at line 143 of file selfsimilar\_dam\_break.cpp.

### 5.13.3.2 void Selfsimilar\_dam\_break::param ( SCALAR L, SCALAR xL, SCALAR xR, SCALAR hinit, SCALAR k1, SCALAR dx\_ex, SCALAR T ) const

Writes the parameters of the solution.

## Parameters

<i>in</i>	<i>L</i>	length of the domain
<i>in</i>	<i>xL</i>	left bound for the water
<i>in</i>	<i>xR</i>	right bound for the water
<i>in</i>	<i>hinit</i>	initial height of the fluid
<i>in</i>	<i>k1</i>	inverse of the friction coefficient in SW equations
<i>in</i>	<i>dx_ex</i>	space step
<i>in</i>	<i>T</i>	final time

Definition at line 181 of file selfsimilar\_dam\_break.cpp.

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

- Headers/[selfsimilar\\_dam\\_break.hpp](#)

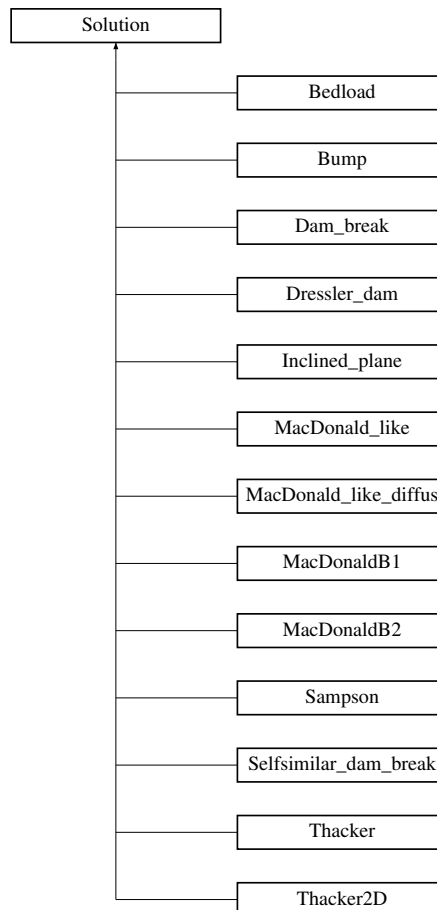
- Sources/[selfsimilar\\_dam\\_break.cpp](#)

## 5.14 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 [deallocation](#) ()  
*deallocation 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`

#### 5.14.1 Detailed Description

Analytic solution.

Class that contains all the common declarations for the solutions.

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

#### 5.14.2 Constructor & Destructor Documentation

##### 5.14.2.1 `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`.

##### 5.14.2.2 `Solution::~~Solution ( ) [virtual]`

Destructor.

Definition at line 186 of file `solution.cpp`.

### 5.14.3 Member Function Documentation

#### 5.14.3.1 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.

#### 5.14.3.2 virtual void Solution::compute ( ) [pure virtual]

Function to be specified in case.

Implemented in [MacDonald\\_like](#), [Bump](#), [Inclined\\_plane](#), [Selfsimilar\\_dam\\_break](#), [Bedload](#), [Dressler\\_dam](#), [MacDonald\\_like\\_diffus](#), [MacDonaldB2](#), [Sampson](#), [Dam\\_break](#), [MacDonaldB1](#), [Thacker](#), and [Thacker2D](#).

#### 5.14.3.3 void Solution::deallocation ( )

deallocation of the tables

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

Definition at line 244 of file solution.cpp.

#### 5.14.3.4 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.

5.14.3.5 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 x
in	<i>vex2D</i>	flow velocity in y
in	<i>zex2D</i>	topography

Definition at line 135 of file solution.cpp.

5.14.3.6 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.

5.14.3.7 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.

#### 5.14.3.8 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.

### 5.14.4 Member Data Documentation

#### 5.14.4.1 SCALAR Solution::dx\_ex [protected]

Space step in x.

Definition at line 83 of file solution.hpp.

#### 5.14.4.2 SCALAR Solution::dy\_ex [protected]

Space step in y.

Definition at line 85 of file solution.hpp.

#### 5.14.4.3 SCALAR\* Solution::hex [protected]

Array for the water height.

Definition at line 92 of file solution.hpp.

#### 5.14.4.4 SCALAR Solution::L [protected]

Dimensions of the domain in x.

Definition at line 79 of file solution.hpp.

#### 5.14.4.5 SCALAR Solution::l [protected]

Dimensions of the domain in y.

Definition at line 81 of file solution.hpp.

#### 5.14.4.6 const int Solution::NX\_EX [protected]

Number of cells in x.

Definition at line 72 of file solution.hpp.



**5.14.4.7** `const int Solution::NY_EX` [protected]

Number of cells in y.

Definition at line 74 of file solution.hpp.

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

Array for the flow discharge.

Definition at line 96 of file solution.hpp.

**5.14.4.9** `SCALAR Solution::T` [protected]

Final time.

Definition at line 77 of file solution.hpp.

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

Array for the flow velocity.

Definition at line 94 of file solution.hpp.

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

Array for the first coordinate.

Definition at line 88 of file solution.hpp.

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

Array for the second coordinate.

Definition at line 90 of file solution.hpp.

**5.14.4.13** `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:

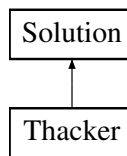
- Headers/[solution.hpp](#)
- Sources/[solution.cpp](#)

## 5.15 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.*

## Additional Inherited Members

### 5.15.1 Detailed Description

Computes Thacker solution.

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

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

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 Thacker::Thacker ( 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 Thacker configuration.

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

#### 5.15.2.2 Thacker::~~Thacker ( ) [virtual]

Destructor.

Definition at line 88 of file `thacker.cpp`.

### 5.15.3 Member Function Documentation

### 5.15.3.1 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`.

### 5.15.3.2 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:

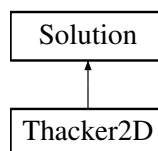
- [Headers/thacker.hpp](#)
- [Sources/thacker.cpp](#)

## 5.16 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.*

## Additional Inherited Members

### 5.16.1 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`.

### 5.16.2 Constructor & Destructor Documentation

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

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

#### 5.16.2.2 Thacker2D::~Thacker2D ( ) [virtual]

Destructor.

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

### 5.16.3 Member Function Documentation

#### 5.16.3.1 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`.

5.16.3.2 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.*

#### Macros

- #define [BEDLOAD\\_HPP](#)

#### 6.1.1 Detailed Description

Computes solutions with bedload.

#### Author

Minh Hoang Le [lemhoang@math.cnrs.fr](mailto:lemhoang@math.cnrs.fr) (2012)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](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 Macro Definition 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](mailto:olivierdelestre41@yahoo.fr) (2010)

Anne-Celine Boulanger [anne-celine.boulanger@inria.fr](mailto:anne-celine.boulanger@inria.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2012)

#### Version

1.01.01

#### Date

2012-03-08

Analytic solution: with a bump, see [Delestre et al. \[2013\]](#) and [Goutal and Maurel \[1997\]](#).

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](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 "selfsimilar_dam_break.hpp"
#include "dressler_dam.hpp"
#include "inclined_plane.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.*

### Macros

- #define [CHOICE\\_SOLUTION\\_HPP](#)

### 6.3.1 Detailed Description

Choice of the solution.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2010-2014)

#### Version

1.02.00

#### Date

2014-01-28

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](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 Macro Definition Documentation

### 6.3.2.1 #define CHOICE\_SOLUTION\_HPP

Definition at line 116 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](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](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](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](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/inclined\_plane.hpp File Reference

Computes the solution over an inclined plane.

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

## Classes

- class [Inclined\\_plane](#)  
*Computes the solutions over an inclined plane.*

### 6.6.1 Detailed Description

Computes the solution over an inclined plane.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Anne-Celine Boulanger [anne-celine.boulanger@inria.fr](mailto:anne-celine.boulanger@inria.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014)

**Version**

1.01.05

**Date**

2014-01-30

Analytic solution: [Delestre et al. \[2012\]](#).

**Copyright**

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](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 [inclined\\_plane.hpp](#).

## 6.7 Headers/macdonald\_like.hpp File Reference

Computes Mac Donald solutions.

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

**Classes**

- class [MacDonald\\_like](#)  
*Computes Mac Donald solutions.*

### 6.7.1 Detailed Description

Computes Mac Donald solutions.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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 et al. \[2013\]](#) and [Vo T. N. \[2008\]](#).

**Copyright**

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [macdonald\\_like.hpp](#).

## 6.8 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.*

### 6.8.1 Detailed Description

Computes Mac Donald solutions with diffusion.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [macdonald\\_like\\_diffus.hpp](#).

## 6.9 Headers/macdonaldb1.hpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

### Classes

- class [MacDonaldB1](#)  
*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](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.10 Headers/macdonaldb2.hpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

### Classes

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

### 6.10.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

#### Author

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.11 Headers/misc.hpp File Reference

Definitions.

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

#### Macros

- #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 1.02.00, 2014-04-29"

## Typedefs

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

### 6.11.1 Detailed Description

Definitions.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2012)

#### Version

1.02.00

#### Date

2014-04-29

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

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

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

### 6.11.2 Macro Definition Documentation

#### 6.11.2.1 #define EPSILON\_H 1.e-12

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

#### 6.11.2.2 #define GRAV 9.81

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

#### 6.11.2.3 #define GRAV\_DEM 4.905

Definition at line 68 of file [misc.hpp](#).

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

Definition at line 65 of file [misc.hpp](#).



### 6.11.2.5 #define PI 3.14159265

Definition at line 69 of file misc.hpp.

### 6.11.2.6 #define VERSION "SWASHES version 1.02.00, 2014-04-29"

Definition at line 72 of file misc.hpp.

## 6.11.3 Typedef Documentation

### 6.11.3.1 typedef double SCALAR

Definition at line 76 of file misc.hpp.

### 6.11.3.2 typedef vector< vector< SCALAR > > TAB

Definition at line 77 of file misc.hpp.

## 6.12 Headers/parameters.hpp File Reference

Gets parameters.

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

### Classes

- class [Parameters](#)  
*Gets parameters.*

### 6.12.1 Detailed Description

Gets parameters.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.13 Headers/sampson.hpp File Reference

Computes Sampson solution.

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

### Classes

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

### 6.13.1 Detailed Description

Computes Sampson solution.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.14 Headers/selfsimilar\_dam\_break.hpp File Reference

Computes self-similar dam break solutions.

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

## Classes

- class [Selfsimilar\\_dam\\_break](#)  
*Computes self-similar dam break solutions.*

### 6.14.1 Detailed Description

Computes self-similar dam break solutions.

#### Author

Serge Bodjona (2013)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014)

#### Version

1.02.00

#### Date

2014-02-20

Analytic solution: self-similar solution for dam break with friction,  
see [Self-similar\\_solutions.pdf](#) in the doc folder or in the bibliography of sourcesup.

#### Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [selfsimilar\\_dam\\_break.hpp](#).

## 6.15 Headers/solution.hpp File Reference

Common file.

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

## Classes

- class [Solution](#)  
*Analytic solution.*

### 6.15.1 Detailed Description

Common file.

**Author**

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

**Version**

1.01.03

**Date**

2012-05-11

Common part for all the solutions.

**Copyright**

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.16 Headers/thacker.hpp File Reference

Computes Thacker solution.

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

**Classes**

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

### 6.16.1 Detailed Description

Computes Thacker solution.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.17 Headers/thacker2d.hpp File Reference

Computes Thacker solutions in 2D.

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

**Classes**

- class [Thacker2D](#)

*Computes Thacker solutions in 2D.*

### 6.17.1 Detailed Description

Computes Thacker solutions in 2D.

**Author**

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.18 Sources/bedload.cpp File Reference

Computes solutions with bedload.

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

### 6.18.1 Detailed Description

Computes solutions with bedload.

#### Author

Minh Hoang Le [lemhoang@math.cnrs.fr](mailto:lemhoang@math.cnrs.fr) (2012)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.19 Sources/bump.cpp File Reference

Computes bumps solutions.

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

### 6.19.1 Detailed Description

Computes bumps solutions.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Anne-Celine Boulanger [anne-celine.boulanger@inria.fr](mailto:anne-celine.boulanger@inria.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2010-2012)

**Version**

1.01.01

**Date**

2012-03-08

Analytic solution: with a bump, see [Delestre et al. \[2013\]](#), [Goutal and Maurel \[1997\]](#).

**Copyright**

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](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](#).

## 6.20 Sources/choice\_solution.cpp File Reference

Choice of the solution.

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

### 6.20.1 Detailed Description

Choice of the solution.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2012-2014)

**Version**

1.02.00

**Date**

2014-02-19

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

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

Definition in file [choice\\_solution.cpp](#).

## 6.21 Sources/dam\_break.cpp File Reference

Computes dam break solutions.

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

### 6.21.1 Detailed Description

Computes dam break solutions.

#### Author

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

#### Version

1.01.05

#### Date

2013-11-20

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

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

Definition in file [dam\\_break.cpp](#).

## 6.22 Sources/dressler\_dam.cpp File Reference

Computes Dressler dam break solution.

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

### 6.22.1 Detailed Description

Computes Dressler dam break solution.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [dressler\\_dam.cpp](#).

## 6.23 Sources/inclined\_plane.cpp File Reference

Computes the solution over an inclined plane.

```
#include "inclined_plane.hpp"
```

### 6.23.1 Detailed Description

Computes the solution over an inclined plane.

**Author**Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)Anne-Celine Boulanger [anne-celine.boulanger@inria.fr](mailto:anne-celine.boulanger@inria.fr) (2010)Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014)**Version**

1.02.00

**Date**

2014-03-11

Analytic solution: [Delestre et al. \[2012\]](#).

**Copyright**

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](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 [inclined\\_plane.cpp](#).

## 6.24 Sources/macdonald\_like.cpp File Reference

Computes Mac Donald solutions.

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

### 6.24.1 Detailed Description

Computes Mac Donald solutions.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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 et al. \[2013\]](#) and [Vo T. N. \[2008\]](#).

#### Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [macdonald\\_like.cpp](#).

## 6.25 Sources/macdonald\_like\_diffus.cpp File Reference

Computes Mac Donald solutions with diffusion.

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

### 6.25.1 Detailed Description

Computes Mac Donald solutions with diffusion.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [macdonald\\_like\\_diffus.cpp](#).

## 6.26 Sources/macdonaldb1.cpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

### 6.26.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

**Author**

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.27 Sources/macdonaldb2.cpp File Reference

Computes Mac Donald pseudo 2d solutions.

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

### 6.27.1 Detailed Description

Computes Mac Donald pseudo 2d solutions.

#### Author

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.28 Sources/parameters.cpp File Reference

Gets parameters.

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

### 6.28.1 Detailed Description

Gets parameters.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2010-2014)

**Version**

1.02.00

**Date**

2014-02-19

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

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

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

## 6.29 Sources/sampson.cpp File Reference

Computes Sampson solution.

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

### 6.29.1 Detailed Description

Computes Sampson solution.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.30 Sources/selfsimilar\_dam\_break.cpp File Reference

Computes self-similar dam break solutions.

```
#include "selfsimilar_dam_break.hpp"
```

### 6.30.1 Detailed Description

Computes self-similar dam break solutions.

#### Author

Serge Bodjona (2013)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014)

#### Version

1.02.00

#### Date

2014-03-11

Analytic solution: self-similar solution for dam break with friction,  
see [Self-similar\\_solutions.pdf](#) in the doc folder or in the bibliography of sourcesup.

#### Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

Definition in file [selfsimilar\\_dam\\_break.cpp](#).

## 6.31 Sources/solution.cpp File Reference

Common file.

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

### 6.31.1 Detailed Description

Common file.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2010-2012)

**Version**

1.01.03

**Date**

2012-05-11

Common part for all the solutions.

**Copyright**

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.32 Sources/swashes.cpp File Reference

Main file.

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

**Functions**

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

### 6.32.1 Detailed Description

Main file.

**Author**

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2012)

**Version**

1.01.03

**Date**

2012-05-11

For more details, we refer to [Delestre et al. \[2013\]](#).

**Copyright**

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.32.2 Function Documentation

### 6.32.2.1 `int main ( int argc, char ** argv )`

Main function of SWASHES, see [Delestre et al. \[2013\]](#).

#### Parameters

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

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

## 6.33 Sources/thacker.cpp File Reference

Computes Thacker solution.

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

### 6.33.1 Detailed Description

Computes Thacker solution.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2010)  
 Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

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

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

## 6.34 Sources/thacker2d.cpp File Reference

Computes Thacker solution in 2D.

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



### 6.34.1 Detailed Description

Computes Thacker solution in 2D.

#### Author

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2011)

Carine Lucas [carine.lucas@univ-orleans.fr](mailto: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](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>.
- 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).
- O. Delestre, S. Cordier, F. Darboux, and F. James. A limitation of the hydrostatic reconstruction technique for Shallow Water equations. *Comptes Rendus Mathématique*, 350(13-14):677–681, 2012. doi:[10.1016/J.crma.2012.08.004](https://doi.org/10.1016/J.crma.2012.08.004). URL <http://hal.archives-ouvertes.fr/hal-00710654>.
- O. Delestre, C. Lucas, P.-A. Ksinant, F. Darboux, C. Laguerre, T. N. T. Vo, F. James, and S. Cordier. SWASHES: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal of Numerical Methods in Fluids*, 72(3):269–300, May 2013. doi:[10.1002/flid.3741](https://doi.org/10.1002/flid.3741). URL <http://hal.archives-ouvertes.fr/hal-00628246>. See Annex on the HAL page for complementary results (with illustrations of each case).
- 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.
- N. Goutal and F. Maurel. Proceedings of the 2<sup>nd</sup> workshop on dam-break wave simulation. Technical Report HE-43/97/016/B, Electricité de France, Direction des études et recherches, 1997.
- I. MacDonald. *Analysis and computation of steady open channel flow*. PhD thesis, University of Reading — Department of Mathematics, Sept. 1996.
- 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.
- A. Ritter. Die Fortpflanzung der Wasserwellen. *Zeitschrift des Vereines Deuscher Ingenieure*, 36(33):947–954, 1892.
- 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 7<sup>th</sup> 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>.
- 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 8<sup>th</sup> 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>.
- J. J. Stoker. *Water Waves: The Mathematical Theory with Applications*. Pure and Applied Mathematics. Interscience Publishers, New York, USA, 1957.

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

# Index

- ~Bedload
  - Bedload, [10](#)
- ~Bump
  - Bump, [12](#)
- ~Choice\_solution
  - Choice\_solution, [16](#)
- ~Dam\_break
  - Dam\_break, [18](#)
- ~Dressler\_dam
  - Dressler\_dam, [20](#)
- ~Inclined\_plane
  - Inclined\_plane, [22](#)
- ~MacDonaldB1
  - MacDonaldB1, [31](#)
- ~MacDonaldB2
  - MacDonaldB2, [33](#)
- ~MacDonald\_like
  - MacDonald\_like, [26](#)
- ~MacDonald\_like\_diffus
  - MacDonald\_like\_diffus, [29](#)
- ~Parameters
  - Parameters, [36](#)
- ~Sampson
  - Sampson, [39](#)
- ~Selfsimilar\_dam\_break
  - Selfsimilar\_dam\_break, [41](#)
- ~Solution
  - Solution, [43](#)
- ~Thacker
  - Thacker, [48](#)
- ~Thacker2D
  - Thacker2D, [50](#)
- abcd
  - Bump, [13](#)
  - Inclined\_plane, [22](#)
- allocation
  - Solution, [44](#)
- BEDLOAD\_HPP
  - bedload.hpp, [54](#)
- Bedload, [9](#)
  - ~Bedload, [10](#)
  - Bedload, [10](#)
  - compute, [10](#)
  - param, [10](#)
  - paramwarning, [11](#)
- bedload.hpp
  - BEDLOAD\_HPP, [54](#)
- Bump, [11](#)
  - ~Bump, [12](#)
  - abcd, [13](#)
  - Bump, [12](#)
  - compute, [13](#)
  - determinant, [13](#)
  - height, [14](#)
  - p, [14](#)
  - param, [14](#)
  - q, [15](#)
  - RHJump, [15](#)
- choice
  - Parameters, [37](#)
- Choice\_solution, [15](#)
  - ~Choice\_solution, [16](#)
  - Choice\_solution, [16](#)
  - Choice\_solution, [16](#)
  - compute, [16](#)
- choicedim
  - Parameters, [37](#)
- choicedomain
  - Parameters, [37](#)
- choicetype
  - Parameters, [37](#)
- compute
  - Bedload, [10](#)
  - Bump, [13](#)
  - Choice\_solution, [16](#)
  - Dam\_break, [18](#)
  - Dressler\_dam, [20](#)
  - Inclined\_plane, [22](#)
  - MacDonald\_like, [26](#)
  - MacDonald\_like\_diffus, [29](#)
  - MacDonaldB1, [31](#)
  - MacDonaldB2, [33](#)
  - Sampson, [39](#)
  - Selfsimilar\_dam\_break, [41](#)
  - Solution, [44](#)
  - Thacker, [48](#)
  - Thacker2D, [50](#)
- Dam\_break, [17](#)

- ~Dam\_break, 18
  - compute, 18
  - Dam\_break, 17
  - Dam\_break, 17
  - function, 18
  - param, 18
- deallocation
  - Solution, 44
- Delta\_topo
  - MacDonaldB1, 31
  - MacDonaldB2, 33
- Delta\_topo\_Darcy\_Weisbach
  - MacDonald\_like, 26
- Delta\_topo\_Manning
  - MacDonald\_like, 27
- Delta\_topo\_diffus
  - MacDonald\_like\_diffus, 29
- determinant
  - Bump, 13
  - Inclined\_plane, 23
- Dressler\_dam, 19
  - ~Dressler\_dam, 20
  - compute, 20
  - Dressler\_dam, 19
  - Dressler\_dam, 19
  - param, 20
- dx\_ex
  - Solution, 46
- dy\_ex
  - Solution, 46
- EPSILON\_H
  - misc.hpp, 62
- function
  - Dam\_break, 18
- GRAV
  - misc.hpp, 62
- GRAV\_DEM
  - misc.hpp, 62
- get\_choice
  - Parameters, 36
- get\_choicedim
  - Parameters, 36
- get\_choicedomain
  - Parameters, 36
- get\_choicetype
  - Parameters, 36
- get\_nxex
  - Parameters, 36
- get\_nyex
  - Parameters, 37
- head
  - Solution, 44
- Headers/bedload.hpp, 53
- Headers/bump.hpp, 54
- Headers/choice\_solution.hpp, 55
- Headers/dam\_break.hpp, 56
- Headers/dressler\_dam.hpp, 56
- Headers/inclined\_plane.hpp, 57
- Headers/macdonald\_like.hpp, 58
- Headers/macdonald\_like\_diffus.hpp, 59
- Headers/macdonaldb1.hpp, 60
- Headers/macdonaldb2.hpp, 60
- Headers/misc.hpp, 61
- Headers/parameters.hpp, 63
- Headers/sampson.hpp, 64
- Headers/selfsimilar\_dam\_break.hpp, 64
- Headers/solution.hpp, 65
- Headers/thacker.hpp, 66
- Headers/thacker2d.hpp, 67
- height
  - Bump, 14
  - Inclined\_plane, 23
- help
  - Parameters, 37
- hex
  - Solution, 46
- Inclined\_plane, 21
  - ~Inclined\_plane, 22
  - abcd, 22
  - compute, 22
  - determinant, 23
  - height, 23
  - Inclined\_plane, 22
  - Inclined\_plane, 22
  - p, 23
  - param, 24
  - q, 24
- L
  - Solution, 46
- I
  - Solution, 46
- MAX
  - misc.hpp, 62
- MacDonald\_like, 25
  - ~MacDonald\_like, 26
  - compute, 26
  - Delta\_topo\_Darcy\_Weisbach, 26
  - Delta\_topo\_Manning, 27
  - MacDonald\_like, 25
  - MacDonald\_like, 25
  - param, 27
- MacDonald\_like\_diffus, 27

- ~MacDonald\_like\_diffus, 29
  - compute, 29
  - Delta\_topo\_diffus, 29
  - MacDonald\_like\_diffus, 28
  - MacDonald\_like\_diffus, 28
  - param, 29
- MacDonaldB1, 30
  - ~MacDonaldB1, 31
  - compute, 31
  - Delta\_topo, 31
  - MacDonaldB1, 31
  - MacDonaldB1, 31
  - param, 32
- MacDonaldB2, 32
  - ~MacDonaldB2, 33
  - compute, 33
  - Delta\_topo, 33
  - MacDonaldB2, 33
  - MacDonaldB2, 33
  - param, 34
- main
  - swashes.cpp, 78
- misc.hpp
  - EPSILON\_H, 62
  - GRAV, 62
  - GRAV\_DEM, 62
  - MAX, 62
  - PI, 62
  - SCALAR, 63
  - TAB, 63
  - VERSION, 63
- NX\_EX
  - Solution, 46
- NY\_EX
  - Solution, 46
- nx\_ex
  - Parameters, 37
- ny\_ex
  - Parameters, 38
- p
  - Bump, 14
  - Inclined\_plane, 23
- PI
  - misc.hpp, 62
- param
  - Bedload, 10
  - Bump, 14
  - Dam\_break, 18
  - Dressler\_dam, 20
  - Inclined\_plane, 24
  - MacDonald\_like, 27
  - MacDonald\_like\_diffus, 29
  - MacDonaldB1, 32
  - MacDonaldB2, 34
  - Sampson, 39
  - Selfsimilar\_dam\_break, 41
  - Thacker, 49
  - Thacker2D, 50
- Parameters, 34
  - ~Parameters, 36
  - choice, 37
  - choicedim, 37
  - choicedomain, 37
  - choicetype, 37
  - get\_choice, 36
  - get\_choicedim, 36
  - get\_choicedomain, 36
  - get\_choicetype, 36
  - get\_nxex, 36
  - get\_nyex, 37
  - help, 37
  - nx\_ex, 37
  - ny\_ex, 38
  - Parameters, 35
- paramwarning
  - Bedload, 11
- q
  - Bump, 15
  - Inclined\_plane, 24
- qex
  - Solution, 47
- RHJump
  - Bump, 15
- SCALAR
  - misc.hpp, 63
- Sampson, 38
  - ~Sampson, 39
  - compute, 39
  - param, 39
  - Sampson, 39
- savefinal2D
  - Solution, 44
- savefinalcritical
  - Solution, 45
- savefinalcriticalinit
  - Solution, 45
- savefinalmu
  - Solution, 45
- Selfsimilar\_dam\_break, 40
  - ~Selfsimilar\_dam\_break, 41
  - compute, 41
  - param, 41
  - Selfsimilar\_dam\_break, 40

- Selfsimilar\_dam\_break, 40
- Solution, 42
  - ~Solution, 43
  - allocation, 44
  - compute, 44
  - deallocation, 44
  - dx\_ex, 46
  - dy\_ex, 46
  - head, 44
  - hex, 46
  - L, 46
  - l, 46
  - NX\_EX, 46
  - NY\_EX, 46
  - qex, 47
  - savefinal2D, 44
  - savefinalcritical, 45
  - savefinalcriticalinit, 45
  - savefinalmu, 45
  - Solution, 43
  - T, 47
  - uex, 47
  - xex, 47
  - yex, 47
  - zex, 47
- Sources/bedload.cpp, 68
- Sources/bump.cpp, 68
- Sources/choice\_solution.cpp, 69
- Sources/dam\_break.cpp, 70
- Sources/dressler\_dam.cpp, 70
- Sources/inclined\_plane.cpp, 71
- Sources/macdonald\_like.cpp, 72
- Sources/macdonald\_like\_diffus.cpp, 72
- Sources/macdonaldb1.cpp, 73
- Sources/macdonaldb2.cpp, 74
- Sources/parameters.cpp, 74
- Sources/sampson.cpp, 75
- Sources/selfsimilar\_dam\_break.cpp, 76
- Sources/solution.cpp, 76
- Sources/swashes.cpp, 77
- Sources/thacker.cpp, 78
- Sources/thacker2d.cpp, 78
- swashes.cpp
  - main, 78
- T
  - Solution, 47
- TAB
  - misc.hpp, 63
- Thacker, 47
  - ~Thacker, 48
  - compute, 48
  - param, 49
  - Thacker, 48
- Thacker2D, 49
  - ~Thacker2D, 50
  - compute, 50
  - param, 50
  - Thacker2D, 50
  - Thacker2D, 50
- uex
  - Solution, 47
- VERSION
  - misc.hpp, 63
- xex
  - Solution, 47
- yex
  - Solution, 47
- zex
  - Solution, 47