UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Modelling of the solid transport and morphological evolution with SLIM

Dissertation presented by
Matthieu FLAMAND

for obtaining the Master's degree in
Civil Engineering

Supervisor(s)
Sandra SOARES-FRAZÃO, Éric DELEERSNIJDER

Reader(s)
Miltiadis PAPALEXANDRIS, Jonathan LAMBRECHTS , Le Hoang ANH

Academic year 2016-2017

Dedication and acknowledgements

First of all I want to thank my promoters: the Professor Soares and the Professor Deleersnijder for being very available when meetings had to set up. Thank you for your help and the directions you gave me to progress in my master thesis.

I will never thank David and Jonathan enough for helping me every time I had a question. Thank you David for according me some of your time every single time I entered your office. I would never be able to work with SLIM if it wasn't without you. Thank you Jonathan for helping me out when I had a problem with SLIM. Even if you were really busy, you could always find some time to help me understanding the complex basics of SLIM and the discontinuous finite elements.

A big thank to Anh, who helped me a lot by explaining the basics of SLIM and help me to prepare the meetings.

Thanks also to the rest of the SLIM team and other researchers of the Euler building who helped me at one point this semester.

A special thank to the Professor Papalexandris who accepted enthusiastically when I asked to be a lector of this mémoire.

Thank you to all my friends, family and classmates who supported me at one point during this semester. Thank you Zab, for all the talks and advices about the mémoire and thank you Typhaine for your incentive when it was needed. Thank you Dani for the long hours we worked together, always a pleasure! And thank you Guillaume for your good advices on anything but the thesis. Thank you Alicia for the walks when we were tired, thank you Aurélie for the dinners when nobody is in Louvain-la-Neuve, thank you Vio and Simon for the good times at the kot, thank you Lulu for helping me with Shakespear's language and a big thanks to my parents.

Thank you Thomas, Ysa, Tuki, Aurélie, Guillaume, Alex, Matthieu, Aline, Didi, Vincent, Julien, Arthur, Js, Manon, Sophie, Edouard, Brieux, Stevenn & Perez, Fefe, Adrien, Juju, Axel and all the others.

And to you, reader of this thesis, if your name is not mentioned above and you are reading this, it means quite a lot to me. Thank you! I hope you will find what you are searching for and that it can help you to move forward.

Table of Contents

List of abbreviations and symbols

A	Empirical constant for the Grass formula
C_{ss}	Weight concentration of suspended sediments
C_s	Smagorinsky coefficient
C_s	Average concentration of suspended sediments and bedload
d_{50}	Diameter of sediment particle greater then 50% of all particles
E	Erosion rate
f_s	Slip factor
g	Gravitational acceleration
h	Bathymetry under a reference level
H	Total water level $(\eta + h)$
k	Horizontal diffusivity
M	Erodibility coefficient
m	Empirical constant for the Grass formula
N_n	Total number of nodes on the domain
N_e	Number of elements on the domain
n	Manning coefficient
n	Order of the Runge Kutta method
p_{atm}	Atmospheric pressure
$\boldsymbol{q_{sb}}$	Unit bedload transport rate
$q_{sb,n}$	Component of the bed transport rate normal to the edge of an element
$\boldsymbol{q}_{sb}^h$	Approximation of $\boldsymbol{q_{sb}}$
$\boldsymbol{q_{sb,i}}$	$\boldsymbol{q_{sb}}$ evaluated at the node i
r^h	Interpolation error
s	Ratio of the sediment density on the water density
$\boldsymbol{u}$	Depth-averaged velocity vector
u	Longitudinal component of the velocity
v	Transversal component of the velocity
w	Vertical velocity.
w	Coefficients of the Runge Kutta method
w_s	Settling velocity
z_b	Height of the bed sediment layer
$\ll f \gg_e$	Designs the integration of function f on the boundary of element e
$< f >_e$	Designs the integration of function f on the surface of element e

α	Coefficients of the Runge Kutta method
β	Coefficients of the Runge Kutta method
ϵ_0	Bed porosity
η	Elevation of the water surface above a reference level
μ	Dynamic viscosity
ν_t	Horizontal eddy viscosity
∇_h	Horizontal del operator $\left(\nabla_h = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}\right)$
Ω	Represents the whole domain
ρ	Water density
$\boldsymbol{\tau_s}$	Surface stress vector
$\boldsymbol{\tau_b}$	Bottom stress vector
τ_d	Critical value of the shear stress for deposition
τ_c	Critical value of the shear stress for erosion
τ_*	Non-dimensional bed shear stress
ϕ	Shape function
ϕ_i	Shape function associated to node i

List of Tables

Table **Page**

List of Figures

Figure **Page**

Chapter 1

Introduction

Rivers and seas have always played an important role in humankind. It provides food when a river deposits fertile ground. It provides an effective way of transport and it provides protection as borders are, until now, defined by water. This is why important cities are often found close to a waterway. But water can also bring destruction, for example, in periods of floods or during a tsunami.

Recently, with the global warming, the amount and the violence of the precipitations have significantly increased in certain regions. This exposes to a new problematic: the dams and the embankment built in the last century are not modelled to resist these new phenomena. An example is the break of the Chandora Dam on the Tapi River in India. It was caused by heavy rains in 1991 [18] and led to the total evacuation of 6 villages. It is thus highly important to have good understanding of the underlying physical mechanism in order to prevent future disasters.

Dam breaches are mainly due to an intensive erosion of the embankment. The usual model used to compute this situation combines a hydrodynamic module, usually the shallow water equations, with a bed updating equation based on the continuity of sediments. The latter equation can be implemented in two ways. One possibility is by solving it independently from the shallow water equations. This method is called the uncoupled or steady approach and works well when the morphological changes occur over a long time. Another possibility is to solve all the equations together, leading to higher computation time but adapted to fast transient flows. This approach is called the coupled approach and has proven to be effective to model dam breaches. [31]

The importance of correctly modelling morphodynamics is also seen in other domains. In estuaries and tidal inlets, a hot topic is the formation of shoals in the channels. It is crucial to have a good insight on the processes causing the morphological changes as a bad management

could make a waterway completely innavigable. The formation can be eliminated but at the price of costly activities like dredging. Various models have been proposed to predict the shoal formation [34]. A first possibility is to use linear models based on stability analysis [28]. A second possibility is by running a three dimensional model of the shallow water equations coupled to the Exner equation. [36] This model has proven to give good results and is capable of computing nonlinear interactions. It is nevertheless time consuming as shoal forming is happening at a long time scale. The idea arises to work with two dimensional, depth averaged equations [15]. This model produced very satisfying results and appealed for further research with a similar model.

The combination of the shallow water and the Exner equations are widely used with finite volumes and much less with finite element methods [32]. However, discontinuous finite elements methods can be very interesting in this domain [19]. It combines the high precision of the finite element method with the fluxes between the elements of the finite volume method.

This master thesis aims to include the morphological changes in the existing SLIM model. SLIM uses the discontinuous Galerkin method to compute the shallow water equations. During this master thesis, the Exner equation was implemented in an uncoupled way and details were given on how to implement the coupled way.

The structure of the thesis is detailed.

- In the first chapter, the model will be detailed. The equations currently implemented in SLIM will be exposed and an equation to include the bedload will be derived. Finally, a basic implementation of this equation in SLIM will be explained.

- In the second chapter three testcases are presented.

  - The first is a compound channel and was used to validate the model for this case. Particularly the suspended sediments were compared with Telemac and a first draft of varying bathymetry was implemented.

  - The second case is a rectangular channel where the bathymetry presents a bump. This case has an approximate analytical solution and the bump is supposed to evolve in a star-shaped pattern. The observed phenomena are due to the moving bedload and this case will validate the implementation of the new equation. The results are compared with the ones obtained by Hudson and Sweby [16].

  - The last case is a channel with a sudden narrowing. As the second case, the main changes are attributed to the moving bedload and are used as a second validation. The results are compared with the ones obtained by Bernard [2] in a flume.

- The third chapter details what should be changed in order to bring an accurate implementation of the equation in SLIM.

- An important part of the thesis has been to understand how to use SLIM. Currently no written material exists on that matter and the last chapter presents a starting guide to work with SLIM. The aim is to help future researcher who will have to use this tool.
- Finally, a conclusion is drawn and directions for future research are given.

Chapter 2

Model

2.1 About SLIM

The first computational model aiming to reproduce the flow in the oceans was initiated by Dr Kirk Bryan in 1969 and was based on a finite difference scheme. Since then, computation power has largely improved and new methods emerged. In 1999, the Université Catholique de Louvain started a research on a finite element ocean model and in 2004, the Second-generation Louvain-la-Neuve Ice-ocean Model (SLIM) was created.

The model solves the so called shallow water equations with the finite element method on an unstructured mesh. It is able to treat one-dimensional, two dimensional and three-dimensional elements but this master thesis will focus on the depth-averaged two dimensional module. The water height and the velocity of the flow are spatially discretized using the discontinous Galerkin method. This method has been chosen as it combines the flexibility of the finite element method and the flux at the interfaces of the finite volume method [3].

After applying the model to real cases as the Grand Barrier Reef [20], the model was tested on estuarine and coastal areas like the Mahakam Delta [4] and the Scheldt Estuary [11]. It produced solid results. Nevertheless, on those domains, morphological changes can play a big role, which is why it would be interesting to be able to take them into account.

2.2 Equations

2.2.1 Hydrodynamics

First of all, the hydrodynamics are solved using the Saint-Venant or shallow water equations. Those are derived from the conservation of mass and conservation of momentum as written by Navier-Stokes and are adapted to a two-dimensional model by averaging the values on the depth. An explanation on how these equations were derived can be found in [11].

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (H\boldsymbol{u}) = 0 \tag{2.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot (\nabla \boldsymbol{u}) = -g\nabla\eta - \frac{1}{\rho}\nabla p_{atm} + \frac{1}{H}\nabla \cdot (H\nu_t(\nabla \boldsymbol{u})) + \frac{\boldsymbol{\tau_s} - \boldsymbol{\tau_b}}{\rho H} \tag{2.2}$$

Where η is the elevation of the water surface above a reference level, h is the bathymetry under that same reference level and $H = \eta + h$ is the total water depth. t is the time, ∇ is the del operator and $\boldsymbol{u}$ is the depth-averaged velocity vector.
g is the gravitational acceleration, ρ is the water density which is assumed to be constant in accordance with the Boussinesq approximation, p_{atm} is the atmospheric pressure at the water surface, ν_t is the horizontal eddy viscosity. $\boldsymbol{\tau_s}$ is the surface stress vector and $\boldsymbol{\tau_b}$ the bottom stress vector.

To derive those equations the following assumptions were made [11]

- The flow is incompressible. (Boussinesq approximation)
- The vertical length scale is much smaller than the horizontal length scale.
- The baroclinic pressure gradient is neglected. This means that the salinity and the temperature have no influence on the density.
- The lateral momentum exchange due to the gradient of velocity on the depth is neglected.
- The bathymetry is constant over time.

The Eddy viscosity is calculated according the Smagorinsky parametrization [30].

$$\nu_t = (C_s\Delta)^2 \sqrt{2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2} \tag{2.3}$$

Where C_s is a non dimensional coefficient, Δ is the characteristic length of the mesh and u and v are respectively the longitudinal and the transversal component of the velocity.

The bottom stress vector is computed using the Chézy-Manning-Strickler formula which is originally developed for one-dimensional flows.

$$\tau_b = \rho g n^2 \frac{\|\boldsymbol{u}\|^2}{H^{1/3}} \tag{2.4}$$

Where n is the manning coefficient.

This equation can be adapted to two-dimensional flows [31]. If the direction of the stress is the same as the velocity, it reads

$$\begin{aligned} \tau_{b,x} &= \tau_b \frac{u}{\|\boldsymbol{u}\|} \\ \tau_{b,y} &= \tau_b \frac{v}{\|\boldsymbol{u}\|} \end{aligned} \tag{2.5}$$

In the scope of this master thesis, the Coriolis term is not calculated as it has no significant impact when using a small domain as a laboratory flume or theoretical cases.

2.2.2 Suspended Sediments

The suspended sediments are computed using a depth-averaged transport equation. The equation and its implementation in SLIM can be found in [11].

$$\frac{\partial (HC_{ss})}{\partial t} + \nabla \cdot (H\boldsymbol{u}C_{ss}) = \nabla (Hk\nabla C_{ss}) \tag{2.6}$$

Where C_{ss} is the weight concentration of suspended sediments in gram per liter of water and k is the horizontal diffusivity.

Eq. 2.6 computes how the sediment particles will propagate in the domain under the influence of the flow and the diffusion. This equation does not take into account that a suspended particle can settle and thus does not take part in the transport process anymore. Other particles, initially on the bed, could be suspended due to erosion and added to the sediment transport. A term is added to the equation to include this phenomena.

$$\frac{\partial (HC_{ss})}{\partial t} + \nabla \cdot (H\boldsymbol{u}C_{ss}) = \nabla (Hk\nabla C_{ss}) + (E - D) \tag{2.7}$$

Where E is the erosion rate and D the settling rate in $\left[kg\, s^{-1}\, m^{-2}\right]$.

The $(E - D)$ term is uncoupled from the rest of the equation. After the suspended sediment concentration has been calculated by means of the first equation (Eq. 2.6), the $(E - D)$ term is computed at each node of the mesh and added to C_{ss}. This has been done because the term is highly irregular over the mesh. Interpolating it with the finite element method could provoke a high erosion where there is little or no soil to be eroded. Uncoupling is made possible by the smaller time scale at which the erosion and settling is happening comparing to than the time scale of the advection and diffusion.

The settling is computed using the formula introduced by Einstein and Krone [7]

$$D = C_{ss} w_s \left(1 - \frac{\tau_b}{\tau_d}\right), \quad if \quad \tau_b < \tau_d \tag{2.8}$$

Where w_s is the settling velocity, τ_b is the bottom shear stress calculated as Eq. 2.4 and τ_d is the critical value of the shear stress for deposition. If the bottom friction is higher than the critical value, the deposition rate is zero.

According to Stoke's law, the settling velocity is

$$w_s = \frac{(\rho_s - \rho) g d_{50}^2}{18\mu} \tag{2.9}$$

Where d_{50} is the diameter of the particle greater then 50% of all the sediment particles and μ is the dynamic viscosity of water, which is supposed to be constant $(\mu = 10^{-3})$.

The erosion is computed using the Partheniades formula [25]

$$E = M\left(\frac{\tau_b}{\tau_c} - 1\right), \quad if \quad \tau_b > \tau_c \tag{2.10}$$

Where M is the erodibility coefficient. This parameter is used as a calibration parameter and is of the order of $1e-5 kg\, m^{-2}\, s^{-1}$ [26]. τ_c is the critical value of the shear stress for erosion and if the bed shear stress is lower, no erosion takes place.

The threshold shear stress for the erosion can be deducted from the Shields - van Rijn diagram. A sediment of grain size $d = 0.091mm$ (fine sand) gives a Shields parameter of $\tau_{ast} = 0.01$. The corresponding critical shear stress is $\tau_c = 0.15 N/m^2$. In this thesis, the same value is used for the threshold of deposition.

2.2.3 Bed updating equation

In the current version of SLIM, the bathymetry is constant. As a result, no real bed updating equation is implemented. A precursor of it can, however, be seen in the layer of available sediment for the erosion. This layer does not influence the hydrodynamics but is used to know if sediments are available to be suspended. The layer grows when settling occurs and decreases with the erosion.

Beside the suspended load, sediments move with the flow by rolling and saltating. The model can be improved by taking this moving bed load into account. It is also important to see that, when sediments move, the bed undergoes morphological changes. This will have an effect on the flow which has to be included in the model. In order to find the influence of the moving bed load on the bathymetry, a conservation of sediment mass is applied on a column of water. (Fig. 2.1). The column has a base of size $\Delta x \times \Delta y$ and includes three layers. The bottom layer holds the immobile sediments and the limit of this layer will define the bathymetry. A second layer comprises the sediments that are in motion, but to heavy to be suspended and the last layer is composed of suspended sediments. If the flow intensifies, initially immobile sediments can be moved, thus changing the limits of the bottom layer.

The flux of sediments on the bed is modelled by $\boldsymbol{q_{sb}}$. It is called the unit bedload transport rate and represents a sediment discharge per unit of width. The units are $\left[m^3\, s^{-1}\, m^{-1}\right]$.

The incoming mass of sediments in the x direction during a period of Δt is

$$\rho_s q_{sb,x}(x, y) \Delta y \Delta t \tag{2.11}$$

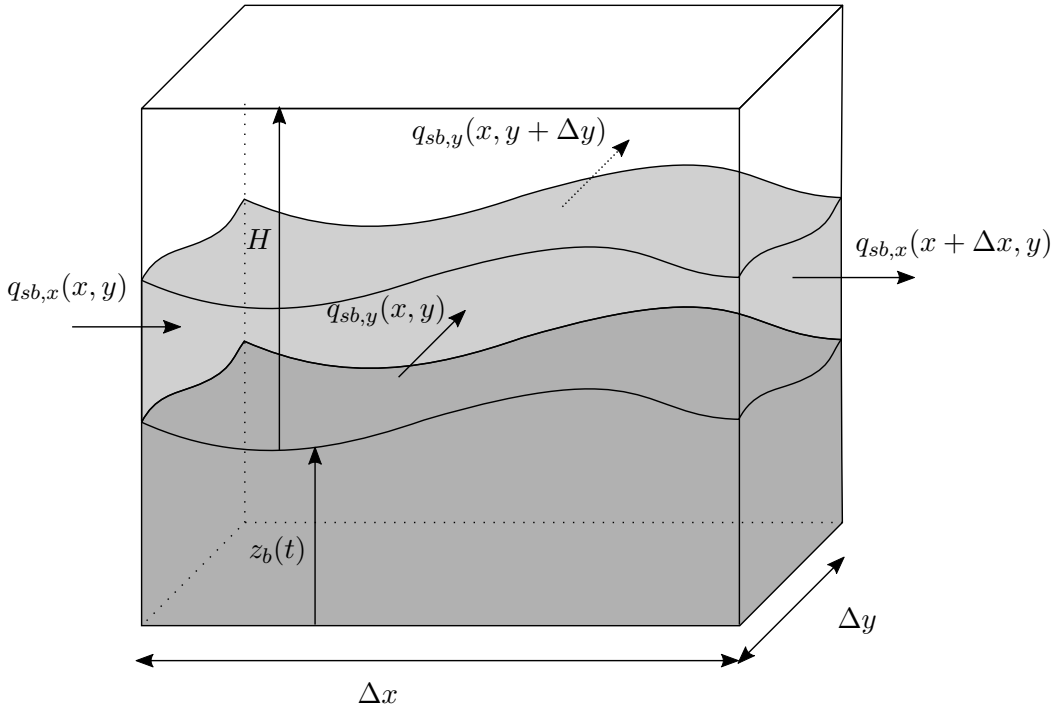

Figure 2.1 – Conservation of mass for the sediments

The outgoing mass of sediments in the same direction and period is

$$\rho_s q_{sb,x}(x + \Delta x, y)\Delta y \Delta t \tag{2.12}$$

Incoming mass of sediments in the y direction is

$$\rho_s q_{sb,y}(x, y)\Delta x \Delta t \tag{2.13}$$

The outgoing mass of sediments in the y direction is

$$\rho_s q_{sb,y}(x, y + \Delta y)\Delta x \Delta t \tag{2.14}$$

The mass of sediments in the bedrock layer at time t is

$$\rho_s z_b(t)\Delta x \Delta y(1 - \epsilon_0) \tag{2.15}$$

Where z_b is the height of the bed sediment layer and ϵ_0 is the bed porosity. After Δt, this becomes

$$\rho_s z_b(t + \Delta t)\Delta x \Delta y(1 - \epsilon_0) \tag{2.16}$$

The mass of sediments outside of the bedrock layer at time t is

$$\Delta x \Delta y H C_s(t) \tag{2.17}$$

Where C_s is the average concentration of suspended sediments and bedload in kilogram per cubic meter of water.

After Δt, this becomes

$$\Delta x \Delta y H C_s(t + \Delta t) \tag{2.18}$$

The conservation of mass can thus be written as

$$\begin{aligned} &\rho_s \Delta y \Delta t \left[q_{sb,x}(x + \Delta x, y) - q_{sb,x}(x, y)\right] \\ &+ \rho_s \Delta x \Delta t \left[q_{sb,y}(x, y + \Delta y) - q_{sb,y}(x, y)\right] \\ &+ \rho_s \Delta x \Delta y (1 - \epsilon_0) \left[z_b(t + \Delta t) - z_b(t)\right] \\ &+ \Delta x \Delta y \left[H C_s(t + \Delta t) - H C_s(t))\right] \\ &= 0 \end{aligned} \tag{2.19}$$

Multiplying this by $\frac{1}{\Delta x \Delta y \Delta t \rho_s}$ gives

$$\begin{aligned} &\frac{q_{sb,x}(x + \Delta x, y) - q_{sb,x}(x, y)}{\Delta x} + \frac{q_{sb,y}(x, y + \Delta y) - q_{sb,y}(x, y)}{\Delta y} \\ &+ (1 - \epsilon_0) \frac{z_b(t + \Delta t) - z_b(t)}{\Delta t} + \frac{1}{\rho_s} \frac{H C_s(t + \Delta t) - H C_s(t)}{\Delta t} = 0 \end{aligned} \tag{2.20}$$

Taking the limit of this equation with $\Delta x, \Delta y$ and Δt tending towards 0, we finally obtain

$$\nabla \cdot \boldsymbol{q_{sb}} + \frac{\partial z_b}{\partial t}(1 - \epsilon_0) + \frac{\partial H C_s}{\partial t} \frac{1}{\rho_s} = 0 \tag{2.21}$$

Usually, the average sediment concentration does not vary much over time. The term can thus be neglected and the equation becomes

$$\frac{\partial z_b}{\partial t}(1 - \epsilon_0) + \nabla \cdot \boldsymbol{q_{sb}} = 0 \tag{2.22}$$

Where $\boldsymbol{q_{sb}}$ is the unit bed transport rate, z_b is the height of the bed sediment layer and ϵ_0 is the bed porosity.

A widely used equation for the unit bedload transport rate is the empirical Meyer-Peter-Muller formula. [22]

$$\boldsymbol{q_{sb}} = 8\sqrt{g(s-1)d_{50}^3} \cdot \max\left\{0, (\boldsymbol{\tau_*} - 0.047)^{3/2}\right\} \tag{2.23}$$

Where s is the ratio of the sediment density on the water density, d_{50} is the particle size bigger then 50% of the sediment particles, τ_* is the non-dimensional bed shear stress and 0.047 is a threshold value. The bed shear stress is calculated as equation 2.4 and its dimensionless form is

$$\boldsymbol{\tau_*} = \frac{n^2 \boldsymbol{u} \|\boldsymbol{u}\|}{(s-1) d_{50} H^{1/3}} \tag{2.24}$$

An other equation that can be used to calculate the unit transport rate of bed sediment was found by Grass [12]

$$\boldsymbol{q_{sb}} = A\boldsymbol{u}\|\boldsymbol{u}\|^{m-1} \tag{2.25}$$

Where A is a dimensional constant determined from experimental data ($A = 0.001$) and m is also a constant such that $1 <= m <= 4$ (here $m = 3$).

2.3 Numerical Implementation

The numerical implementation of the shallow water equations (2.1 and 2.2) and the transport equation (2.6) in SLIM has been explained in various publications [11] [20] [26]. Their development will thus not be explained here. This section will mainly detail the newly implemented bed-updating equation (Eq. 2.22).

2.3.1 spatial discretization

As the main equations, the method used to solve the bed updating equation is a discontinuous finite element method using piecewise linear shape funtions. The odel is solved on triangular elements and the shape functions on are represented in Fig. 2.2. The value at the summits are different for two neighboring elements making the interpolation discontinuous at the interface and a flux will have to be calculated. The P_1^{DG} has already proven to be effective with the Exner equation as it can be used with a steep batyhmetry without producing spurious oscillations. Another advantage is its simplicity to implement as it is already being used for the shallow water equations.

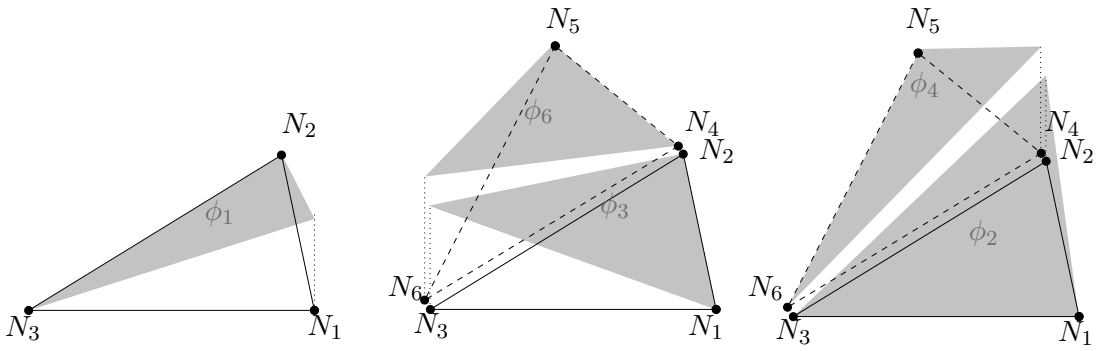

Figure 2.2 – P_1^{DG} shape functions on neighbouring elements

The finite element Galerkin methods consist in taking the interpolation error weighted with a shape function equal to zero. [21]

$$\sum_{e=1}^{N_e} \Big(< \phi r^h >_e \Big) = 0 \tag{2.26}$$

Where N_e is the number of elements on the domain, r^h is the interpolation error, ϕ is a shape function and the notation $< f >_e$ designs the integration of function f on the surface of element e.

To solve the equation 2.22, the weak formulation is used.

It is obtained as follow

$$\sum_{e=1}^{N_e}\left(< \phi\frac{\partial z_b}{\partial t} >_e (1-\epsilon_0) + < \phi\nabla\cdot\boldsymbol{q_{sb}} >_e\right) = 0 \tag{2.27}$$

Using the integration by parts

$$\sum_{e=1}^{N_e}\left(< \phi\frac{\partial z_b}{\partial t} >_e (1-\epsilon_0) + < \nabla\cdot(\boldsymbol{q}_{sb}\phi) >_e - < \boldsymbol{q}_{sb}\cdot\nabla\phi >_e\right) = 0 \tag{2.28}$$

Applying the divergence theorem

$$\sum_{e=1}^{N_e}\left(< \phi\frac{\partial z_b}{\partial t} >_e (1-\epsilon_0) + \ll q_{sb,n}\phi \gg_e - < \boldsymbol{q}_{sb}\cdot\nabla\phi >_e\right) = 0 \tag{2.29}$$

Where $q_{sb,n}$ is the component of the bed transport rate normal to the edge of the element and the notation $\ll f \gg_e$ designs the integration of function f on the boundary of element e.

The value of $\boldsymbol{q_{sb}}$ is calculated at each node and is approximated over the whole domain as

$$\boldsymbol{q}^h_{sb}(x,y) = \sum_{i=1}^{N_n} \boldsymbol{q_{sb,i}}\phi_i(x,y) \tag{2.30}$$

Where $\boldsymbol{q}^h_{sb}$ is the approximation of $\boldsymbol{q_{sb}}$, $\boldsymbol{q_{sb,i}}$ is $\boldsymbol{q_{sb}}$ evaluated at the node i, ϕ_i is the shape function associated to node i and N_n is the total number of nodes on the domain. Each element has three nodes and thus $N_n = 3N_e$

As the method produces discontinuities between two elements, the value of $q_{sb,n}$ is not the same at the right and the left side of the interface. Only one value can be accepted thus leading to a Riemann problem. The value on the interface will be computed as a combination of the $\boldsymbol{q^h_{sb}}$ on the edges of both elements. As the equation is implemented in an uncoupled way, the sediment transport does not depend on the bathymetry and is thus constant at each time step. Therefore, the value on the edge is computed as the mean between the left side term and the right side term.

$$q_{sb,n} = \frac{1}{2}\left(\boldsymbol{q^L_{sb}}\cdot\boldsymbol{n} + \boldsymbol{q^R_{sb}}\cdot\boldsymbol{n}\right) \tag{2.31}$$

When an element is situated on the boundary of the domain, the edge has only one neighbouring element. Two cases are possible. If the boundary is a wall, the value on the boundary is taken equal to zero. If the boundary is open, the value on the non-existing neighbouring element is set to the exact same value as the value on the existing element. This results in a value on the edge equal to the value inside of the domain.

The unknown in Equation 2.30 is $\frac{\partial z_b}{\partial t}$. It is possible to approximate this value in the same way as for q_{sb}.

$$\left(\frac{\partial z_b}{\partial t}\right)^h(x,y) = \sum_{i=1}^{N_n}\left(\frac{\partial z_b}{\partial t}\right)_i \phi_i(x,y) \tag{2.32}$$

Where $\left(\frac{\partial z_b}{\partial t}\right)^h$ is the approximation of $\frac{\partial z_b}{\partial t}$ and $\left(\frac{\partial z_b}{\partial t}\right)_i$ is $\frac{\partial z_b}{\partial t}$ evaluated at the node i. The aim is to find $\left(\frac{\partial z_b}{\partial t}\right)_i$ at each node. There are thus N_n unknown and the same amount of equations are needed to solve the problem. Equation 2.29 has to be true for each shape function ϕ_j associated to the node j, giving a total of N_n equations.

Replacing the approximated values (Eq. 2.30 and Eq. 2.32) in Equation 2.29

$$\sum_{e=1}^{N_e} < \left(\frac{\partial z_b}{\partial t}\right)^h \phi_j >_e (1-\epsilon_0) = \sum_{e=1}^{N_e} < \boldsymbol{q_{sb}^h} \cdot \nabla\phi_j >_e - \sum_{e=1}^{N_e} \ll q_{sb,n}\phi_j \gg_e \tag{2.33}$$

The left side term can be decomposed as follow

$$\begin{aligned}\sum_{e=1}^{N_e} < \left(\frac{\partial z_b}{\partial t}\right)^h \phi_j >_e &= \frac{1}{1-\epsilon_0} \sum_{i=1}^{N_n} \sum_{e=1}^{3N_e} < \left(\frac{\partial z_b}{\partial t}\right)_i \phi_i\phi_j >_e \\ &= \frac{1}{1-\epsilon_0} \sum_{i=1}^{N_n} \left(\frac{\partial z_b}{\partial t}\right)_i \sum_{e=1}^{3N_e} < \phi_i\phi_j >_e \\ &= \frac{1}{1-\epsilon_0} \sum_{i=1}^{N_n} \left(\frac{\partial z_b}{\partial t}\right)_i \iint_\Omega \phi_i\phi_j dxdy\end{aligned} \tag{2.34}$$

Where Ω represents the whole domain.

Equation 2.33 can be written in a matrix form

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{B} \tag{2.35}$$

Where A is a $3N_e \times 3N_e$ matrix with

$$A_{ij} = \iint_\Omega \phi_i\phi_j dxdy$$

x is a vector of size $3N_e$ with

$$x_i = \left(\frac{\partial z_b}{\partial t}\right)_i$$

B is a vector of size $3N_e$ containing the right side term of Eq. 2.33

$$B_j = \iint_\Omega \frac{\boldsymbol{q_{sb}^h} \cdot \boldsymbol{\nabla}\phi_j}{1-\epsilon_0} dxdy - \oint_{d\Omega} \frac{q_{sb,n}\phi_j}{1-\epsilon_0} ds$$

2.3.2 Temporal integration

Previous paragraph explains how $\frac{\partial z_b}{\partial t}$ is computed. In order to find the thickness of the sediment layer, it has to be integrated over the time. In the scope of this work, the bed updating equation is uncoupled from the shallow water equations. This means that first, the shallow water equations are computed. During this step the bathymetry is taken as a constant. The next step is to calculate the change in bathymetry with the Exner equation. This time, the

velocity and water height are constant. At the next time step, the process will repeat but using the new bathymetry as a constant to solve the shallow water equations.

This way of doing is not the most accurate but it makes sense as the bathymetry change is happening at a much slower rate than the change in velocity and water height. It could be interesting to implement the coupled equations in SLIM. A way of implementing it is detailed in Section 4.

Given that the equations are uncoupled. The bed transport rate q_{sb} does not change with z_b. The time integration is obtained by multiplying the result of the spatial discretization with the time step.

$$z_{b,i}^{t+1} = z_{b,i}^{t} + \Delta t \left(\frac{\partial z_b}{\partial t} \right)_i^t \tag{2.36}$$

Chapter 3

Test cases

To test the equations, three cases were used. The first is a compound channel. It is used to validate the model and implement a first draft of a variable bathymetry in SLIM. The second a sand bed presenting a bump and the third, a narrowing. The latter two test cases were used to validate the implementation of the Exner equation in the SLIM model.

3.1 Compound channel

A compound channel is a theoretic model simulating a river when the floodplain is submerged. It is composed of a deep section and two shallow sections at each side. The deeper part is modelling the main channel in which the river is continuously flowing and thus always submerged. The shallow parts are modelling the floodplain.

A flume reproducing a compound channel was built at the Hydraulics Laboratory of the Université Catholique de Louvain and several experiences have been conducted on this flume [8]. The aim of those experiments was to have a better understanding of diffusion processes and sediment settling on floodplains during overbank flow. The results were compared with numerical simulations using Telemac-2D.

Further research was conducted on the same flume by Pham Van. The purpose was to test if the Smagorinsky turbulent model could be used to calculate the horizontal eddy viscosity [27]. This time, the SLIM model was used and it was deduced that the Smagorinsky approach fitted the experiments better than the model used by Fraselle.

The flow in a compound channel is complex. The velocity in the floodplains is generally smaller than the velocity in the main channel, due to the increased roughness in the floodplains. There is a momentum transfer between the main channel and the floodplains creating a high shear layer. This causes turbulences that are not negligible [29]. This is why it is mostly

modelled by means of a three-dimensional model. Nevertheless, it is interesting to try to model those phenomena with a two-dimensional depth-averaged model. The advantage is a reduction of calculations. The three-dimensional effects are included in empirical formulas giving more approximate results.

The purpose of this section is to validate the SLIM two-dimensional model for compound channels, particularly for sediment transport. A first step will be to reproduce Pham Vans's results. The second step is to reproduce the diffusion of the solute tracer that Fraselle experienced in the flume and the last step is to find a formula for the erosion and the settling that fits the results obtained by Fraselle with Telemac.

3.1.1 Experiments

Fraselle made various experiments on the UCL flume. The laboratory setup and the methodology are detailed in his thesis [8]. Two sets of data were used in this thesis. The hydrodynamics and the solute tracer are based on the RBR20 experiments (Rigid bed, rough floodplains with a discharge of 20 l/s) and the results for the sediment transport comes from the RBS15 experiments (Rigid bed, smooth floodplains with a discharge of 15 l/s). Those experiment are briefly resumed.

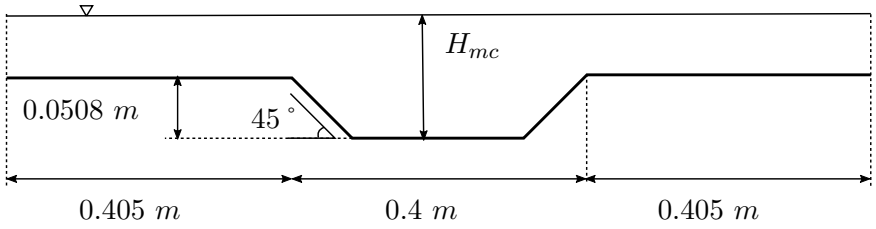

Figure 3.1 – Section of the compound channel

The flume has a 10 m long measurement section with a 2 m long inlet tank and a 2 m long outlet zone. The section of the flume is represented on figure 3.1. It has a total width of 1.21 m. The main channel has a width of 0.4 m and lies 0.058 m deeper than the floodplain. The transition between the floodplain and main channel is made by a 45 ° slope. The flume is made of plywood and has a manning coefficient of 0.0091 in the main channel and in the floodplain. The value is 0.011 on the slopes which are made of different wood.

A constant discharge is imposed at the inlet. The water is injected by three different supplies to ensure the same discharge over the whole section. In order to get an uniform flow, the longitudinal slope and the downstream water depth can be adjusted. The observed parameters are listed in table 3.1.

The injected water is clear. The experiment for a solute tracer consists in injecting NaCl at $x = 2$ m through a tube of 13 mm diameter at a rate of 1 g/s

For all experiments with sediments, a similar technique is used. The supplied water is sediment-free and sand of $d_{50} = 91\ \mu m$ is injected upstream. Very few experimental results are available in Fraselle's thesis. This last part is thus mainly a comparison between the results obtained with Telemac-2D and the ones obtained with SLIM.

	$Q\ [m/s]$	S_0	$\eta\ [m]$
Hydrodynamics	0.02	0.019	0.0727
Solute tracer	0.02	0.019	0.0727
Sediment transport	0.015	0.019	0.063

Table 3.1 – Flow parameters observed during experiments

3.1.2 Mesh

The mesh is generated using the GMSH mesh generator [9] and is represented in Fig. 3.2. The elements have a maximum edge length of 0.1 m resulting in a total of 3466 triangles.

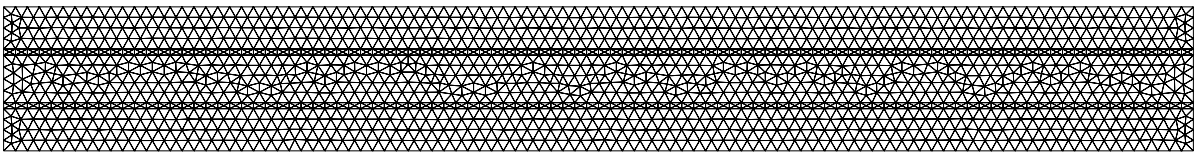

Figure 3.2 – Mesh used for the compound channel

3.1.3 Hydrodynamics

3.1.3.1 Initial and boundary conditions

The purpose is to get as fast and as close as possible to the uniform flow. It is, therefore, important to find a good set of initial and boundary conditions. It appears that the most effective way is by setting a constant discharge upstream and a constant water level downstream.

Other tests were carried, in particular by imposing the discharge at both boundaries or by imposing a water level at both boundaries. Results were not as good as the first option. Those were also the conditions used by Fraselle and Pham Van.

At the walls a partial slip condition is imposed. This choice is explained later. It has been implemented in the current version of SLIM for purpose of this master thesis.

The slip factor occurs at the calculation of the velocity on the boundary. In SLIM, the value on the boundary is calculated as a combination of the value inside the domain and a fictive value outside. The implementation of the slip factor consists in defining this external value.

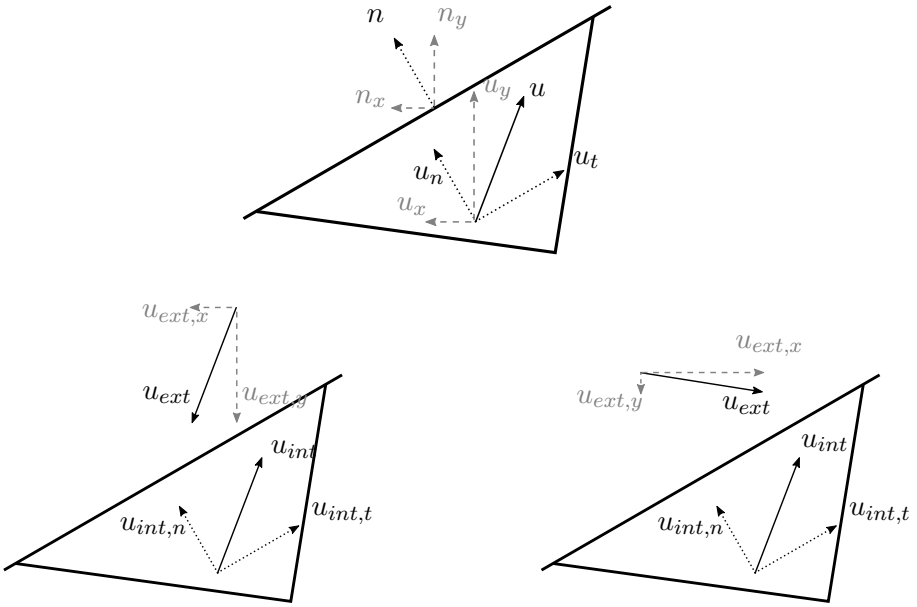

Figure 3.3 – Implementation of the slip factor

As showed in the upper part of Fig. 3.3, the values of the velocity normal and tangent to the boundary can be written as

$$\begin{aligned} u_n &= u_x n_x + u_y n_y \\ u_t &= -u_x n_y + u_y n_x \end{aligned} \tag{3.1}$$

When the slip factor is zero, the velocity at the boundary should be zero. In order to obtain this value, the external velocity should be equal, but in the opposite direction, to the internal velocity. This case is depicted in the lower left part of Fig. 3.3.

When the slip factor is one, the normal component of the velocity at the boundary should be zero and the tangential component at the boundary should be equal to the tangential component inside of the domain. In order to obtain this value, the external velocity should have an equal component in the tangential direction and an opposite component in the normal direction. This case is depicted in the lower right part of Fig. 3.3.

A solution to this system are the following equations

$$\begin{aligned} u_{ext,x} &= -u_{int,n} n_x - f_s u_{int,t} n_y \\ u_{ext,y} &= -u_{int,n} n_y + f_s u_{int,t} n_x \end{aligned} \tag{3.2}$$

Where f_s is the slip factor.

The slip factor matching Fraselle's results is 0.996. The values used at the boundaries for the hydrodynamics are resumed in Tab. 3.2.

It is hard to evaluate the exact manning coefficient. The fact that a depth averaged model is used does not take three-dimensional effects into account which can be implemented by

Upstream Condition	Downstream Condition	Walls
$Q = 0.02\ l/s$	$\eta = 0.0752\ m$	$f_s = 0.996$

Table 3.2 – boundary conditions

adapting the manning coefficient. The value leading to the uniform flow has to be found by trial and error. The last parameter that has to be set is the eddy viscosity. Pham Van found that the Smagorinsky model (Eq. 2.3) was a good approximation for the compound channel [27]. The Smagorinsky coefficient can vary and depends, for example, on the mesh. In this case, a coefficient of 0.4 is used. The final values of the parameters used in this simulation can be found in Tab. 3.3

n_{mc}	n_w	n_{fp}	C_s	S_0
0.01	0.001	0.016	0.4	0.0019

Table 3.3 – Parameters of the model

As the domain is small and the water level shallow, it is difficult to stabilize the flow. An instability can grow really fast, which can lead to a negative water depth. In order to avoid this phenomena, a time step of 1 s is used. The boundary conditions are also introduced progressively, starting with 0 and growing linearly towards the final value.

3.1.3.2 Results

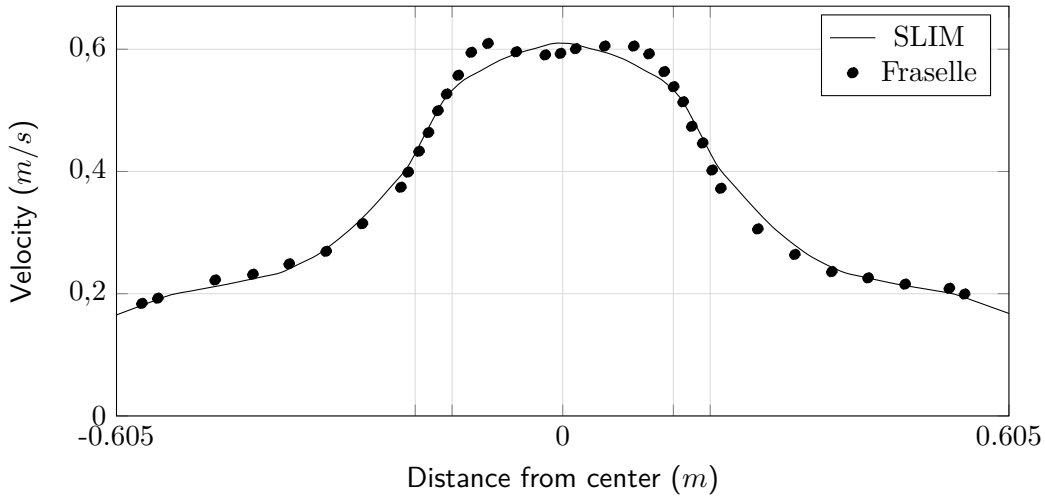

Figure 3.4 – Velocity profile in the section of the flume

Figure 3.4 depicts the results of the simulation with SLIM. The dotted line represents the experimental results of Fraselle. SLIM provides a good approximation. It must be kept in mind that a lot of parameters are empirical. Those parameters include the manning coefficient, the

slip factor and the smagorinsky coefficient. They where defined in order to reach a solution as close as possible to the results. A study of the influence of those parameters and a comparison with Telemac-2D is carried.

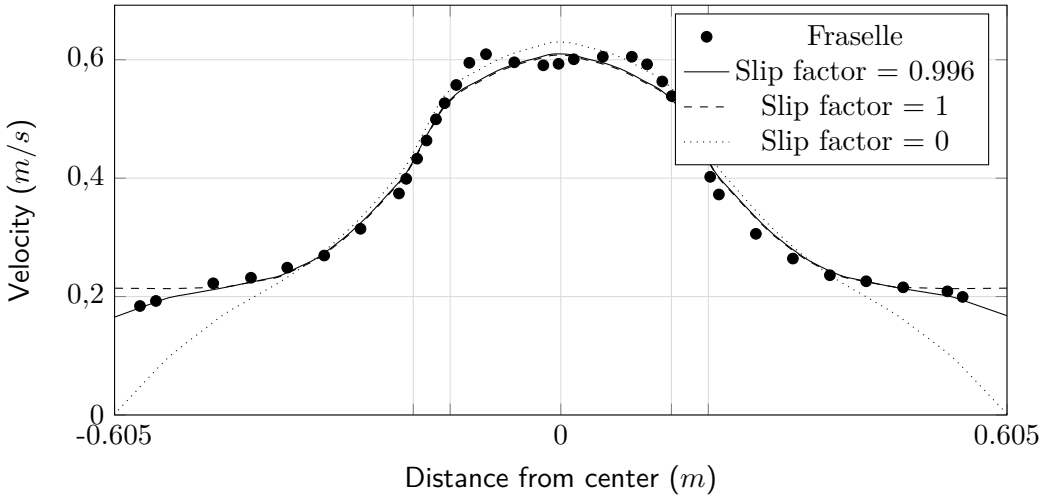

Figure 3.5 – Influence of the slip coefficient on the velocity profile

A first parameter that will be discussed is the influence of the partial slip condition. Figure 3.5 compares the results for a no slip condition ($f_s = 0$), for a partial slip condition ($f_s = 0.996$) and for a full slip condition ($f_s = 1$). The slip conditions act as expected confirming the good implementation. A significant difference between the total and partial slip condition is observed.

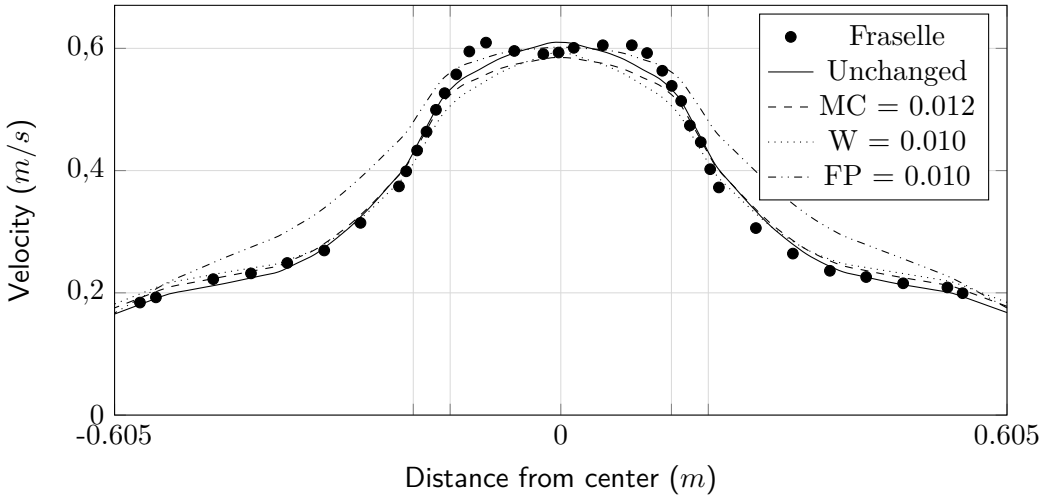

Figure 3.6 – Influence of the manning coefficient on the velocity profile

The manning coefficients are similar to the ones used by Fraselle and Pham Van. The exact values can be found in Tab. 3.4. In the other models, no information is given on the value used in the transition zone. As depicted in the graph, a change at the walls will have a similar

effect to a change in the main channel. Increasing it will slightly lower the velocity in the main channel and increase the velocity in the floodplains. The friction coefficient on the floodplains behaves as expected. A lower value will lead to a higher velocity in the floodplains.

	n_{mc}	n_w	n_{fp}
SLIM	0.010	0.001	0.016
Pham Van	0.0098	/	0.01492
Fraselle	0.010	/	0.014

Table 3.4 – Manning coefficients used in different simulations

It is difficult to compare the parameters with the ones of Fraselle and Pham Van. Fraselle used a different model for the Eddy viscosity and Pham Van described highly varying parameters in function of the mesh. The mesh used in this model is different from the one used by Pham Van, leading to a different set of parameters. Fig. 3.7 shows the results using Pham Van's parameters on the mesh used in this simulation. The values of the parameters can be found in Tab. 3.5. It is also important to keep in mind that, since Pham Van's research, SLIM underwent many changes which could also explain some differences.

n_{mc}	n_{fp}	C_s	η	f_s
0.0098	0.01492	0.1581	0.0732	1.0

Table 3.5 – Parameters used by Pham Van

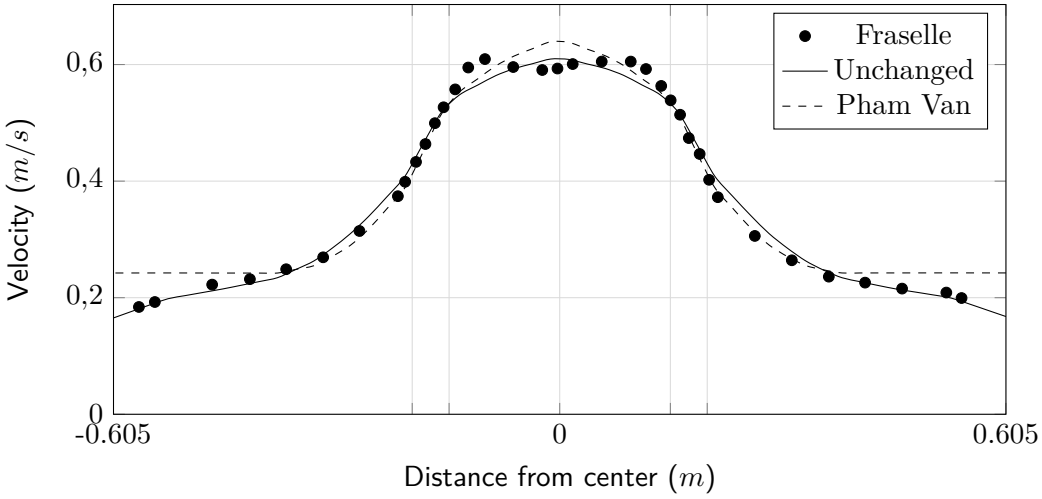

Figure 3.7 – Velocity profile using the same parameters as Pham Van

3.1.4 Solute tracer

The tests with a solute tracer where carried by injecting NaCl in the main channel at 2 m from the inlet. To compute the concentration of NaCl over the domain, Eq. 2.6 is used.

3.1.4.1 Initial and boundary conditions

Initially, no NaCl is present in the domain. The tracer injection is implemented with a source term at $x = 2\ m$. At the downstream boundary a symmetry condition is applied which means that the fictive value outside of the boundary is the same as the value inside.

As Fraselle mentioned in his thesis, the diffusivity is not taken constant over the whole domain. At the interface between the main channel and the floodplains, large scale turbulent structures exists [29]. These phenomena influence the diffusion of the sediments but are not computed with a two-dimensional model. In order to take them into account, the diffusivity coefficient is modified.

Fraselle tried different types of repartitions of the diffusivity on the domain. This analysis is based on the one that led to the best results. The values, however, are not the same. The left figure of Fig. 3.8 represents the repartition of the diffusivity coefficient over the section with the values used in SLIM. The right figure represents the ones used by Fraselle in Telemac.

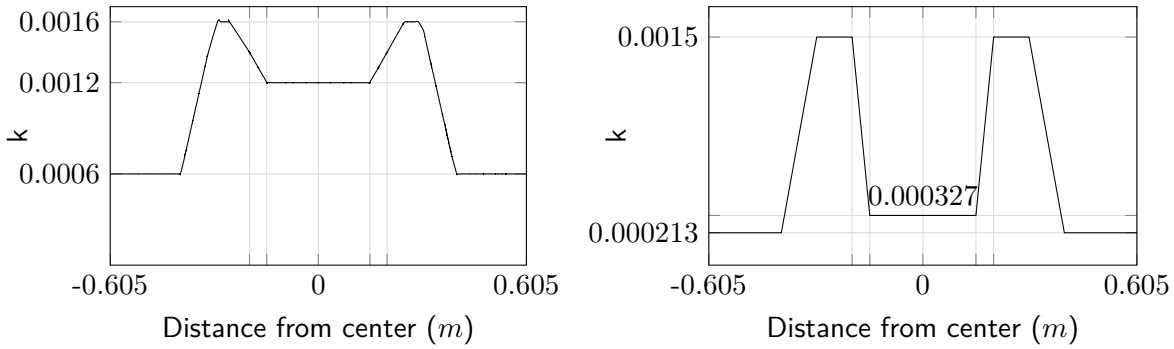

Figure 3.8 – Distribution of the diffusivity coefficient in the section, using SLIM (Left), using Telemac (right)

3.1.4.2 results

The NaCl concentration profile at $x = 9\ m$ is shown in Fig. 3.9. Due to a lack of experimental data provided in Fraselle's thesis, the results can only be compared with the simulation of Telemac. The aim is thus to validate the method in order to calculate the sediment transport. By adapting the diffusivity coefficients, a solution close to the one obtained by Fraselle is found.

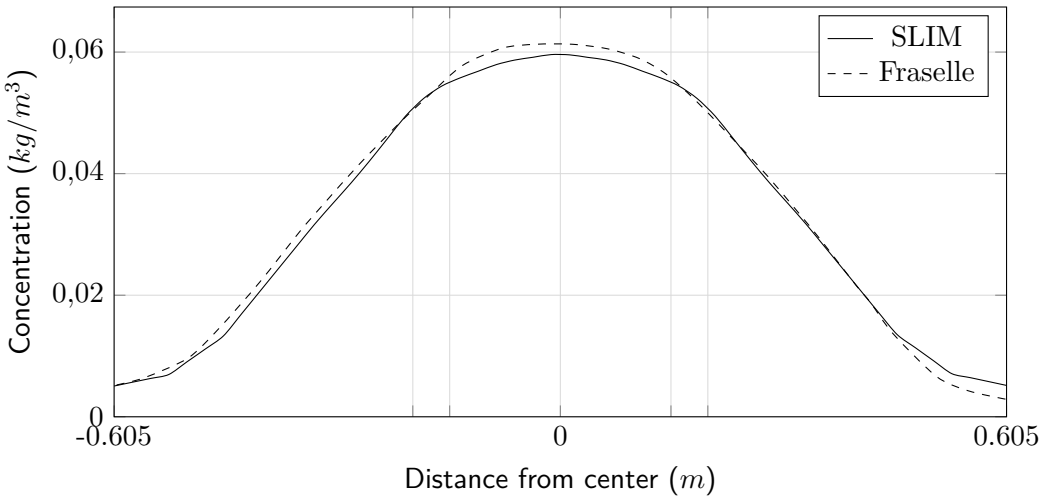

Figure 3.9 – Concentration of the solute tracer at $x = 9\ m$

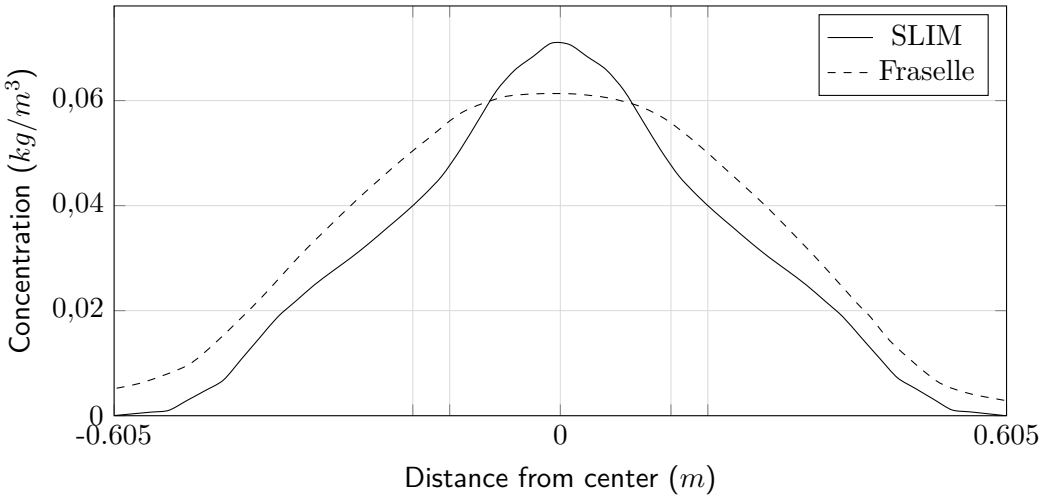

Figure 3.10 – Concentration of the solute tracer using Fraselle's parameters

As the diffusivity coefficients are dependent on the model used, it is interesting to compare SLIM with Telemac-2D. Fig. 3.10 depicts the influence of the diffusivity on SLIM and Telemac. The dashed line shows the results obtained by Telemac-2D using the diffusivity coefficients presented in the left part of Fig. 3.8, while the solid line shows the results in SLIM using the same parameters.

3.1.5 Sediment transport

The experiments on the transport of sediments were realised with a lower discharge of $q = 0.015$ m^3/s and smoother floodplains. The parameters of the flow had thus to be adapted and the new set is presented in Tab. 3.6. Sediments are injected at the upstream boundary in the main

channel and the flood plain is initially sediment-free. The details of the setup can be found in Fraselle's thesis [8].

$q\ [m^3/s]$	C_s	$\eta\ [m]$	n_{mc}	n_w	n_{fp}
0.015	0.3	0.063	0.010	0.01	0.012

Table 3.6 – Parameters for $q = 0.015\ m^3/s$

3.1.5.1 Transfer to the floodplains

The initial and boundary conditions on the concentration are the same as for the solute tracer. A concentration of 1 g/l of sediments is injected upstream in the main channel. The experiments are realised on a rigid bed which in SLIM can be modelled by setting an initial bed layer containing zero sediment. No erosion will thus occur.

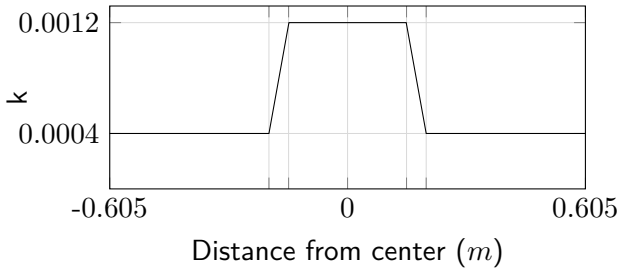

Figure 3.11 – Distribution of the diffusivity coefficient used for the sediments

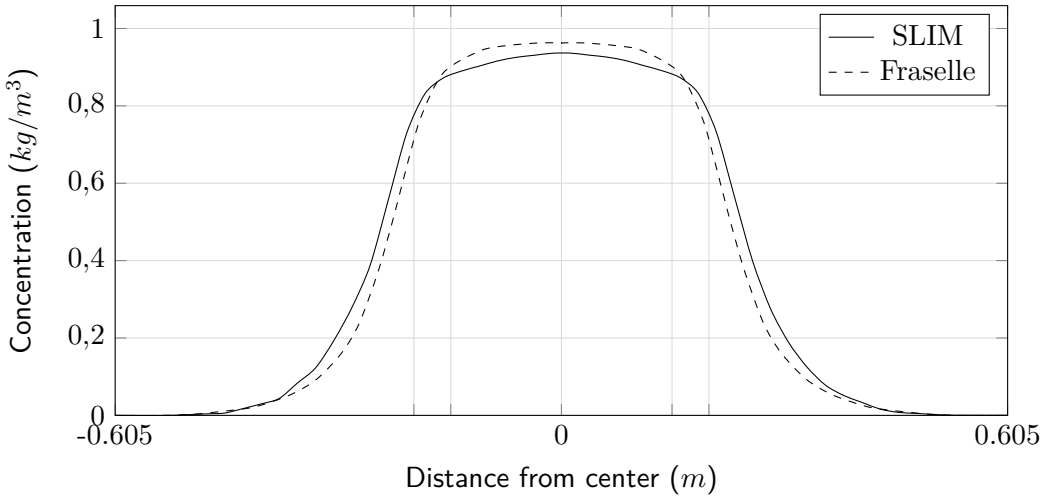

Figure 3.12 – Concentration profile of sediments without erosion and settling at $x = 6\ m$

In order to visualize the sediment transfer from the main channel to the floodplains, Fraselle plotted the theoretical case in which the sediments do not erode neither settle. All the sediments are suspended and it allows to calibrate the diffusivity coefficients. The sediment transport is

indeed different from a solute tracer and the diffusivity has to be adapted. Fig. 3.12 shows the concentration profile without settling. The transverse repartition of the diffusivity coefficients used for that matter are depicted in Fig. 3.11.

3.1.5.2 Erosion and settling

Now all the parameters of the flow and the sediment transport are known, the erosion and the settling can be included. Eq. 2.7 is computed and the parametrization of the erosion and the settling is detailed in section 2.2.2. The parameters of the modelisation are the same as previous simulation except for the sediment injection. 0.8 g/l of sediments are now injected upstream in the main channel. The transverse concentration profile is shown in Fig. 3.13 for $x = 2, 4, 6$ and 8 m.

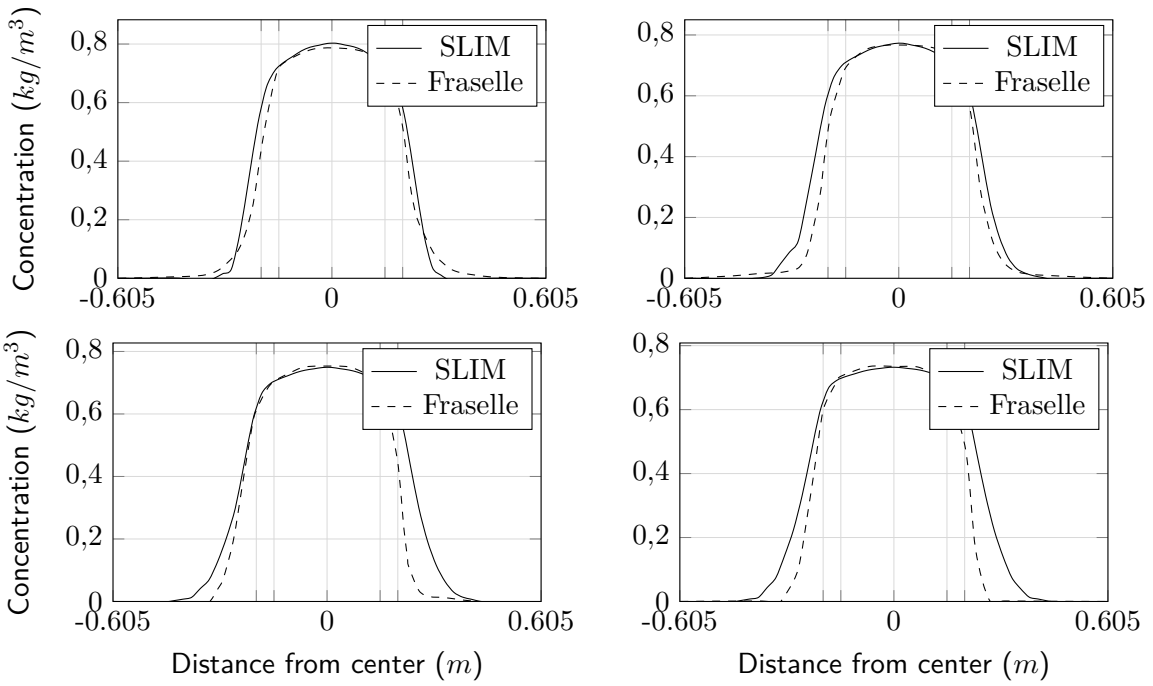

Figure 3.13 – Concentration of sediments after settling at $x = 2$ m (Top left), $x = 4$ m (Top right), $x = 6$ m (Bottom left,) $x = 8$ m (Bottom right).

The parametrization of the erosion and the settling used by Fraselle is different than the one used in SLIM. Replicating the approach of Fraselle is difficult as not every parameter is detailed. It was tried to reproduce his method in SLIM but the results were not satisfying, leading to think that different formulas were used. As the formulas are empirical and adapted to specific cases, big variations exists between them. Final results can thus be very different depending on which is used. Another possibility is that the numerical scheme of the erosion and diffusion term is different in Telemac which could aggravate the difference.

The results presented in Fig. 3.13 give a good approximation from what was computed with Telemac. The settling on the floodplain is underestimated leading to a higher concentration of sediments in that zone. It was tried to modify the calibration parameters to reach the same rate of settling as Fraselle, but in order to reach it, their value must be highly altered. The parameters are then out of the range prescribed for fine sand. Fraselle reported an overestimation of the sediment settling in the Telemac simulation, which could explain why lower values are found.

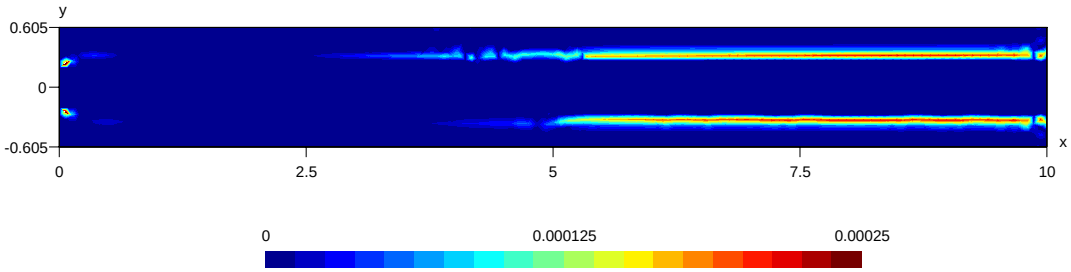

Figure 3.14 – Deposition in $[m]$ calculated with SLIM after 1620 s

The deposition over the domain obtained with SLIM is represented in Fig. 3.14. The pattern can be compared with Fig. 3.15, which shows the results obtained by Fraselle. The top figure presents the experimental result and shows where the sediment settled. It is very similar to the pattern obtained with SLIM. The bottom figure presents the results obtained with Telemac by Fraselle. A difference of a factor 10 can be observed for the maximum deposition height. This can be explained by the lower deposition rate of SLIM.

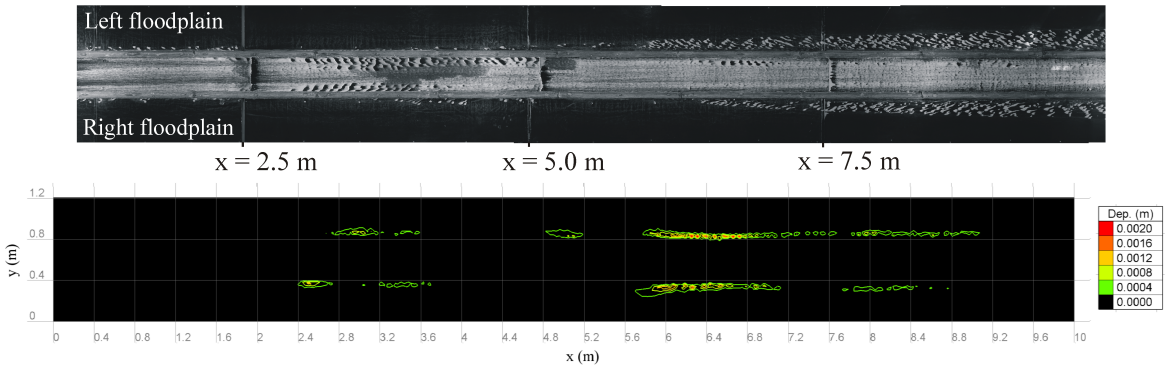

Figure 3.15 – Deposition experienced experimentally (Top) and with Telemac (Bottom) after 1620 s [1]

[1] Figures coming from (Fraselle, 2011 [8])

3.1.6 Discussion

Section 3.1.4 and 3.1.5 compared the results obtained by Fraselle with the Telemac-2D model and the results obtained with SLIM. They showed that Telemac was more diffusive than SLIM and that some parameters had to be modified in order to obtain the same results. This is probably due to the use of a different scheme in both models. In SLIM, discontinuous Galerkin is used, which is known to be less diffusive than other finite elements methods. In Telemac, the streamline upwind Petrov-Galerkin scheme is used [5]. This scheme is also known to reproduce a good diffusion. An in depth-study of both scheme is out of the scope of this thesis but could be interesting to perform.

Finally, the first version of a variable bathymetry was implemented in SLIM. The deposition and erosion calculated with Eq. 2.8 and 2.10 are added to the bathymetry with

$$\frac{\partial z_b}{\partial t} = \frac{D - E}{\rho_s(1 - \epsilon_0)} \tag{3.3}$$

As depicted in Fig. 3.14 the deposition process is very slow and at the time scale of the experiments no changes where observed in the flow.

3.2 Dome

The case used to test the moving bed load is a flat bathymetry with a bump in the middle. This case has an approximate analytical solution developed by De Vriend [33], which is known to make the bump evolve into a star-shaped pattern. It is a theoretical case of which results are not meant to model a real life situation. This analysis is used as a first validation of the implementation of the Exner equation (Eq. 2.22) in SLIM. The validation is thus purely based on the results of previous simulation and has never been experimented in a flume.

As the problem is solely mathematical, the shallow water equations are reduced to their basic form and do not take the dissipative terms into account.

$$\begin{aligned} &\frac{\partial \eta}{\partial t} + \nabla \cdot (H\boldsymbol{u}) = 0 \\ &\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot (\nabla \boldsymbol{u}) = -g\nabla\eta \end{aligned} \tag{3.4}$$

The test is based on the research conducted by Sweby and Hudson [16]. Their study focuses on comparing the steady and the unsteady approach to solve the equations. The steady approach is the resolution of the shallow water equations separately from the bed updating equation. The unsteady approach is thus the resolution in a coupled way. The study shows that the unsteady approach should be preferred even if the computation time is bigger. The equations were solved by means of the finite volume method with a high resolution scheme based on a Roe's scheme.

In the scope of this thesis, only the unsteady approach will be tested. In order to validate the implementation, the results will be compared to the ones obtained by Sweby and Hudson.

3.2.1 Domain and mesh

The domain is represented in Fig. 3.16. It has a width of 1000 m and a length of 1500 m. The dome is initialy comprised between $x = 500$ m and 700 m and $y = 400$ m and 600 m. The bathymetry has a maximum height is 1 m and is defined by a sinusoidal function.

$$z_b(x,y) = \begin{cases} \sin^2\left(\dfrac{\pi\,(x-500)}{200}\right)\sin^2\left(\dfrac{\pi\,(y-400)}{200}\right) & if \quad 500 \leq x \leq 700, \quad 400 \leq y \leq 600 \\ 0 & elsewhere \end{cases} \tag{3.5}$$

The problem is solved on a mesh of triangular elements. The mesh is represented on figure (Fig. 3.17). The elements have edges of approximately 20 m. In comparison, the mesh used by Sweby and Hudson consists of rectangular cells of 20 $m \times 20$ m

3.2.2 Initial and boundary conditions

Initially, the water height is flat and has a value of $10m$ where the bathymetry is zero. The initial velocity is set to

$$u(x,y) = \frac{10}{H(x,y)}$$
$$v(x,y) = 0 \tag{3.6}$$

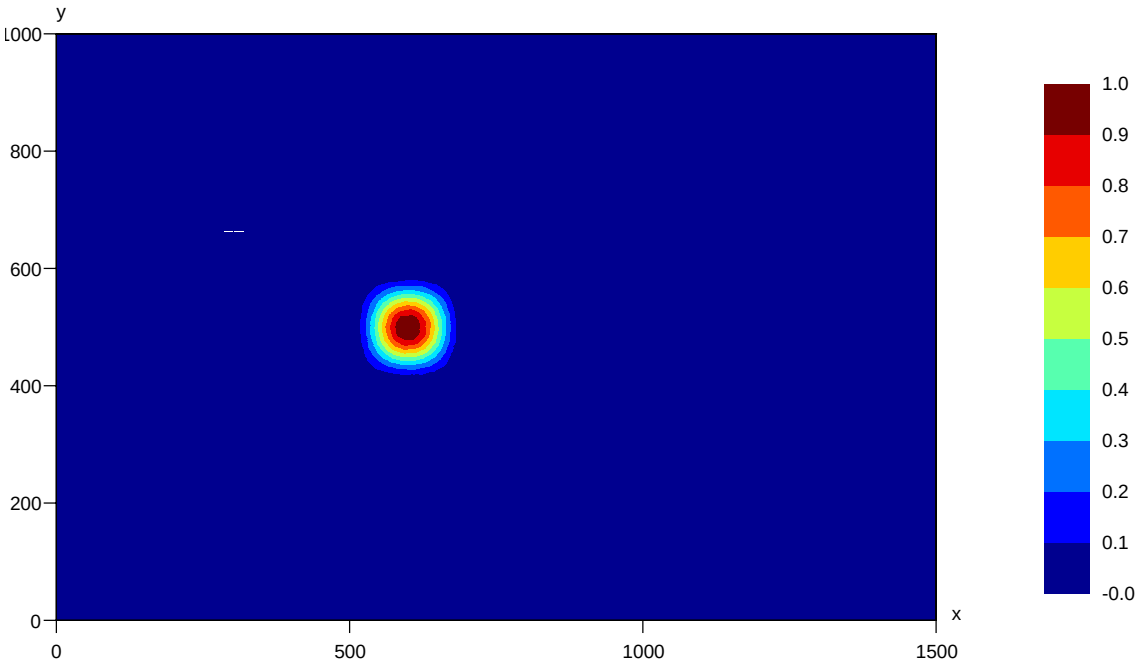

Figure 3.16 – Initial bathymetry showing a bump at $x = 600\ m$ and $y = 500\ m$

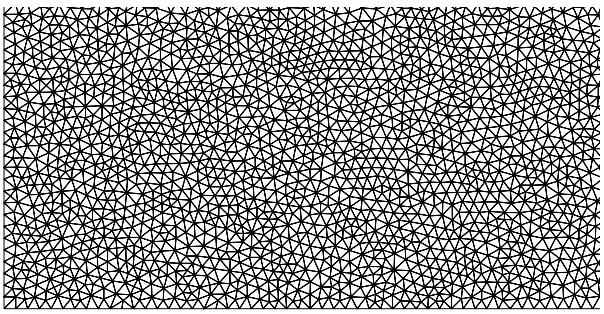

Figure 3.17 – Mesh of the dome case

The boundary conditions at $x = 0\ m$ and $x = 1500\ m$ are imposed on both the water level and the velocity. In this case, a constant water level of 10 m and a constant velocity of 1 m/s are used. At $y = 0\ m$ and $y = 1000\ m$ wall conditions are imposed with a slip condition.

To calculate the sediment transport rate, Hudson and Sweby used the Grass formula 2.25. The sediment is fine sand ($d_{50} = 0.25\ mm$) resulting in $A = 0.001$ and a porosity $\epsilon = 0.4$. Using the same model as Hudson, a value of $m = 3$ is chosen.

Like for the compound channel model, the test case is running first, only computing the hydrodynamic equations to allow the flow to settle and avoid the bathymetry to be influenced by the initial conditions. When the flow is steady, the bath updating equation is added.

3.2.3 Results

When trying to compute this case, an error occurred. The discontinuous Galerkin scheme causes the bathymetry to be discontinuous and for now, SLIM can not work with it. The solution that was chosen is to make it continuous by setting the value at each node equal for all the neighboring elements.

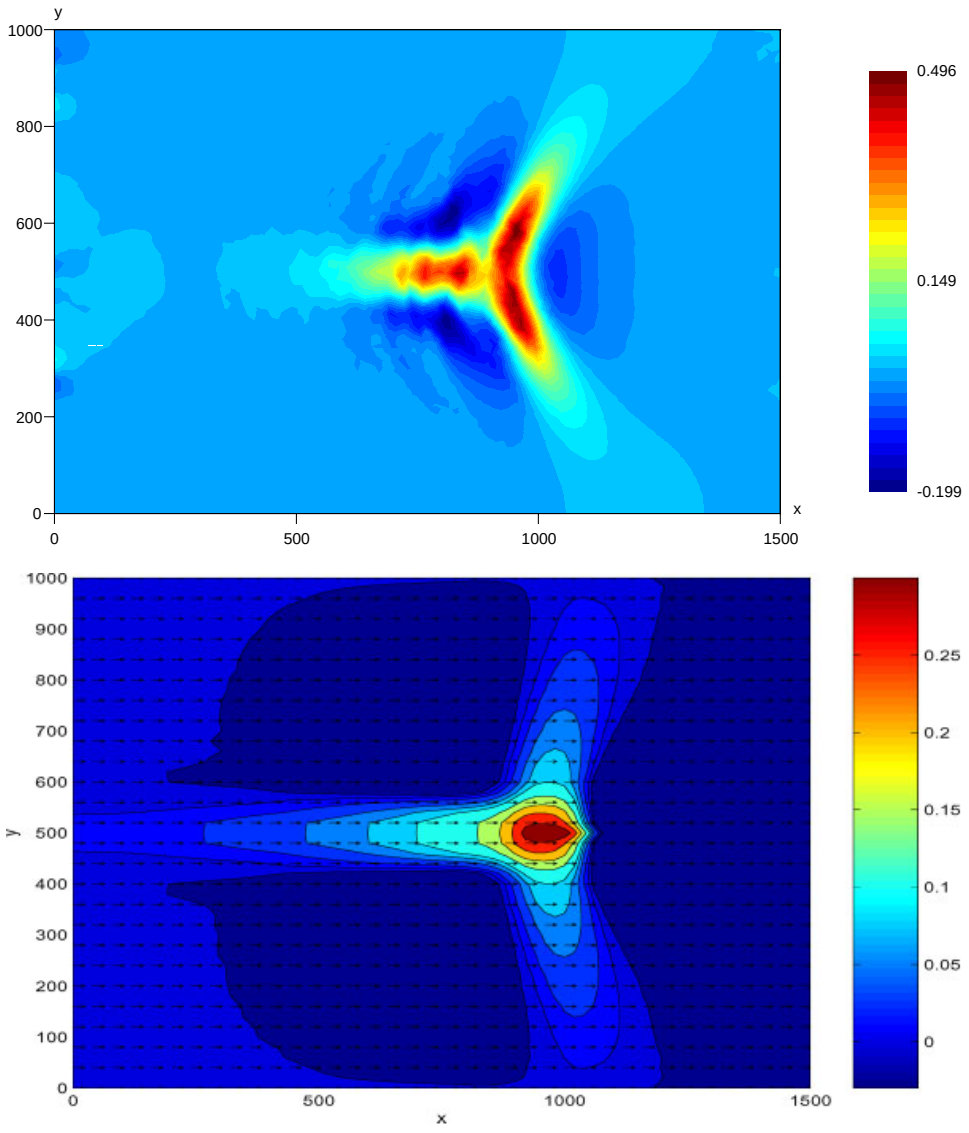

Figure 3.18 – Result of the bump model after 200 hours using SLIM (Top) and obtained by Hudson and Sweby (Bottom) [2]

[3]

The simulation is stopped after 200 h and the results are depicted in Fig. 3.18. SLIM gives coherent results and the same star shape appears. In the center, the dome has moved to $x = 1000$ m which is equivalent to Hudson's results. Nevertheless, the solution seems to be less diffusive. The highest point in SLIM is 0.5 m while it is 0.3 m in Hudson's simulation. Also the front legs of the star are further in the top figure.

A study was carried by Hudson where he compared the use of different schemes on a shallow water-Exner model [17]. Results show a high variability in diffusivity and angle of the legs. After 100 h, the highest point varies between 0.45 m and 0.9 m. In this case, a value of 0.81 m is found. The approximate analytical solution provides an equation for the angle of spread. This is the angle at which the legs spread when the bump moves forward. Fig 3.19 shows the evolution of the bump after: 25 h, 50 h, 75 h and 100 h with the spread angle represented. For a parametrization of the bedload transport rate of the form $q_{sb} = a\|\boldsymbol{u}\|^m$, it has been deduced that

$$\tan\alpha = 3\sqrt{3}\,\frac{m-1}{9m-1} \tag{3.7}$$

and thus for the Grass parametrization (Eq. 2.25) $\alpha = 21.78$ $^\circ$. In the simulation, a spread angle of 33 $^\circ$ has been found, which is in the range of other simulations [6]. It actually shows that a less diffusive solution is closer to the analytical solution thus confirming the good result of SLIM.

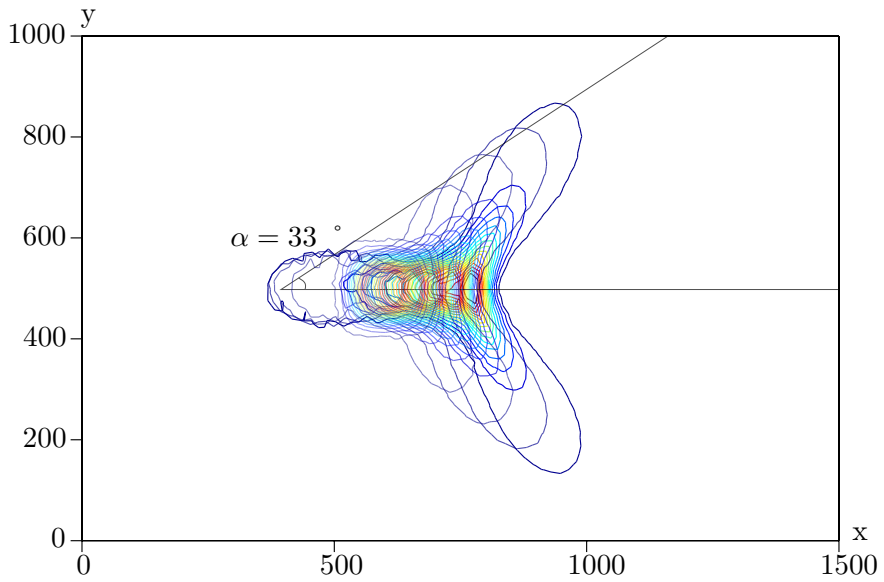

Figure 3.19 – Evolution of the bump after 25 h, 50 h, 75 h and 100 h with the spread angle

[3] Figure coming from (Fraselle, 2011 [8])

Finally, some instabilities are noticeable in the SLIM simulation. A limiter has been applied to avoid overshoots but the solution remained disturbed. Hudson in Sweby reported the presence of kinks at the back, when using the steady approach (Fig. 3.18). Those were not visible with the unsteady approach. It would be interesting to implement the unsteady approach in SLIM and see if the instabilities disappear. Another possibility is that those instabilities rise from the used scheme. A deeper study would have to be carried with the implementation of more advanced schemes.

A widely used equation to calculate the sediment transport rate is the Meyer-Peter-Muller equation (Eq. 2.23). It is tempting to try to use this second model. The same sediment is used $d_{50} = 0.25\ mm$ and $\rho_s = 2650\ kg/m^3$ and a manning coefficient of 0.010 is chosen. Given that this is a theoretical case with very fine sediments and that the formula was empirically deduced for natural rivers, the results are deceiving. The bump barely moves.

Introducing the characteristic values in Eq. 2.24, the bottom shear stress is

$$\boldsymbol{\tau_*} = \frac{0.010^2 1^2}{0.00025(2.65-1)10^{1/3}} = 0.1125$$

The transport rate is than

$$q_{sb} = 8\sqrt{9.81(2.65-1)0.00025^3} \cdot (0.1125 - 0.047)^{3/2} = 2.1e^{-6} m^2/s$$

.

In comparison, the sediment transport rate computed with the Grass formula (Eq. 2.25) is

$$q_{sb} = 1^3 0.001 = 0.001 m^2/s$$

Which is 500 times more. This can be explained by a very low d_{50} that will not be found in natural rivers.

3.3 Narrowing

The last test case is based on the research conducted by Marilyn Bernard for her master thesis. The purpose was to evaluate the influences of secondary currents in an abrupt narrowing (Fig. 3.20). She observed a lowering of the bathymetry at the narrowing showed in Fig. 3.21. This phenomena is mainly due to the sediment transport in the bulk layer. And is modelled with the Exner equation (Eq. 2.22). Her results are used in this thesis to validate, on a real case, the implementation of the Exner equation in SLIM.

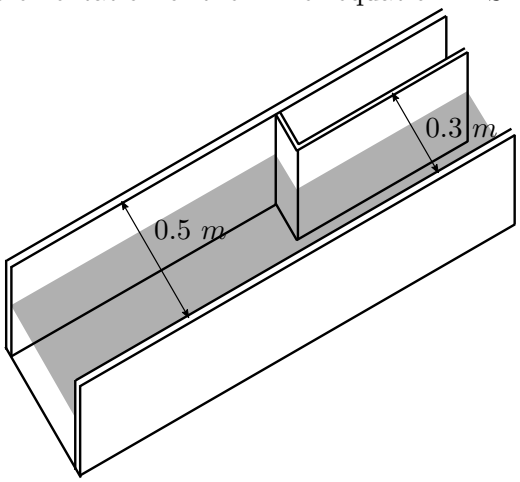

Figure 3.20 – conceptual view of the narrowing

Figure 3.21 – Experimental observation in the narrowing [4]

3.3.1 Experiment

The experiments were realised on a flume in the UCL Laboratory. The details of the setup and the measurements can be found in Bernard's thesis [2]. The details that are needed to build the test case are resumed here.

Fig. 3.22 presents a top view of the flume. It has a length of 7.4 m and a width of 0.5 m at its largest section. At 2 m from the endpoint, the channel narrows to a width of 0.3 m. Water is injected upstream at a discharge of 0.010 m^3/s.

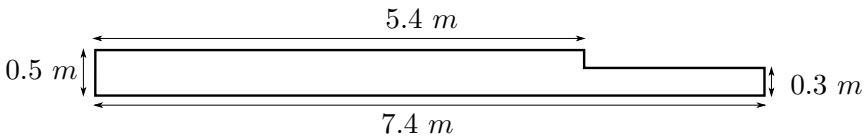

Figure 3.22 – Dimensions of the narrowing flume

The bed of the channel consists of a sediment layer of 0.15 m. It is initially horizontal and is made of sand. The sand has a median grain diameter $d_{50} = 0.00165$ m. The manning coefficient due to this sand was estimated to 0.018. The parameters are resumed in Tab. 3.7.

[4] Figure coming from (Bernard, 2009 [2])

The numerical results of SLIM will be compared with the experimental data, but also with the numerical results obtained by Bernard. She solved the shallow water equations coupled with the Exner equation using a finite volume model with a Harten Lax van Leer sheme [14]. In the equation for the conservation of momentum, she neglects the shear stress as it is much smaller than the turbulent dissipation and her dissipative term uses another parametrization than SLIM. More details can be found in her thesis [2].

$Q\ [m^3/s]$	$d_{50}\ [m]$	n	$\rho_s\ [kg/m^3]$	ϵ_0
0.01	0.00165	0.018	2650	0.4

Table 3.7 – Parameters of the sediments used in the narrowing

3.3.2 Implementation

Unlike the previous case, the model now tries to reproduce a real situation. This is why the complete shallow water equations described in Eq. 2.1 and Eq. 2.2 are used. They are combined to the Exner equation (Eq.2.22) and are solved in an uncoupled way. The sediment transport rate is calculated with the Meyer-Peter-Muller formula (Eq. 2.23).

The equations are solved on the mesh represented in Fig. 3.23. In the coarse zone, the edges of the elements have a size of 0.1 m. The mesh has been refined at the narrowing where most of the phenomena occurs. The edges in this zone are 0.05 m long.

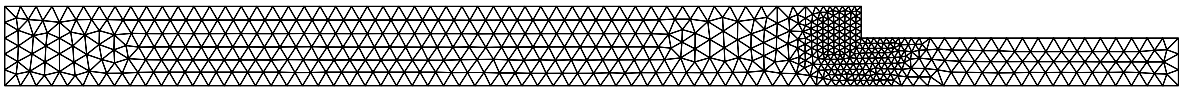

Figure 3.23 – Mesh used to model the narrowing

At the upstream boundary, a constant discharge of 0.01 m^3/s is imposed and at the downstream boundary, a constant water height is set. As Bernard does not mention an imposed water height in her thesis, the value is deduced from the experimental results and is fixed to 0.05 m.

Like the two first cases, the simulation is first ran with the hydrodynamics only. When the flow has reached its equilibrium, the Exner equation is added.

3.3.3 Results

When the flow stabilizes, the resulting velocity acts like expected. Fig. 3.24 represents the depth averaged velocity where the section reduces. In the smaller section, a same discharge must pass which leads to a higher velocity. A zone of recirculation arises behind the angle.

Fig. 3.25 represents the bathymetry after 600 s. The reference bathymetry is taken at 0.15 m as it is the thickness of the sediment layer. No change is observed in the wider part

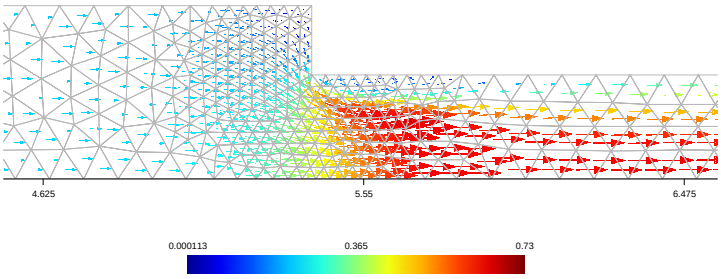

Figure 3.24 – Velocity in the narrowing before the bathymetry change

of the channel. At the narrowing, sand has been eroded to a depth of 10 *cm*. At the zone of recirculation, a little ridge of 2 *cm* was formed.

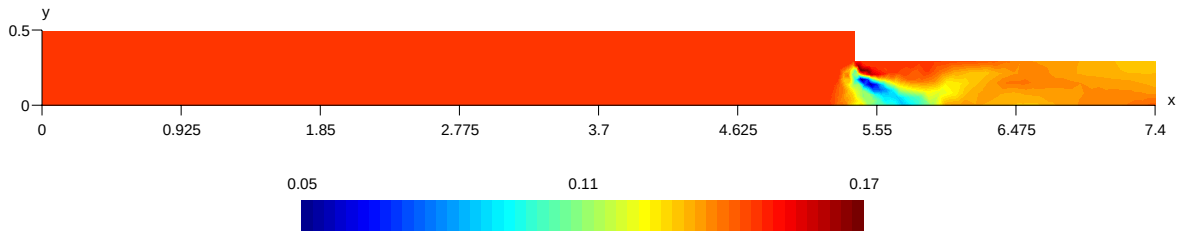

Figure 3.25 – Bathymetry of the narrowing in SLIM after 600 *s*

The results Bernard experimented in the flume is depicted in Fig. 3.26 (Top). The scale is slightly different than the SLIM simulation. While the deepest point is at 5 *cm* in SLIM, it was at 7.5 *cm* in the flume. Also the height of the ridge is overestimated in SLIM.

In the flume, the deepest point is situated at the angle while it lies further in the numerical model. It is interesting to compare these results with the numerical results of Bernard (Fig. 3.26 Bottom). They are similar to the ones obtained with SLIM. This strengthen the conclusion of Bernard saying that the formation of the pit is influenced by a turbulent three-dimensional structure that can not be modelled with a 2D model. It could also be due to the difficulty of a numerical model to predict a phenomena happening on the boundary.

Fig. 3.27 shows a longitudinal cut at $y = 0.15\ m$. The same observations are made. The SLIM model overestimates the depth of the pit and the prediction is globally correct elsewhere. The numerical results of Bernard are shallower. This is because she integrated the fact that the river can not dig infinitely in the bed layer. When the bathymetry is to steep, sediments will glide under the influence of the gravity.

The water height is represented in Fig. 3.28 and a longitudinal cut is plotted with the experimental results in Fig. 3.29. There is a slight underestimation of the water height upstream.

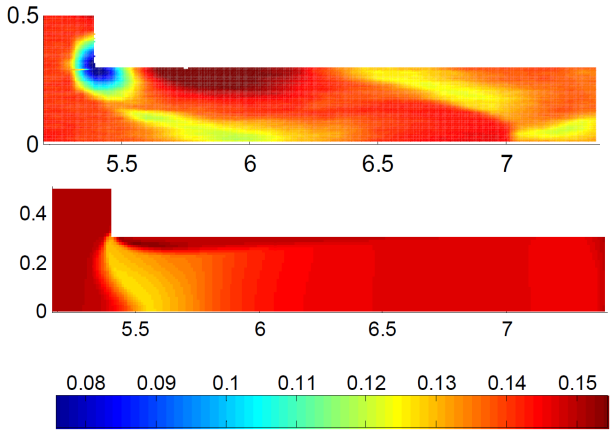

Figure 3.26 – Bathymetry of the narrowing found by Bernard after 600 s. Experimentally (Top) , Numerically (Bottom)

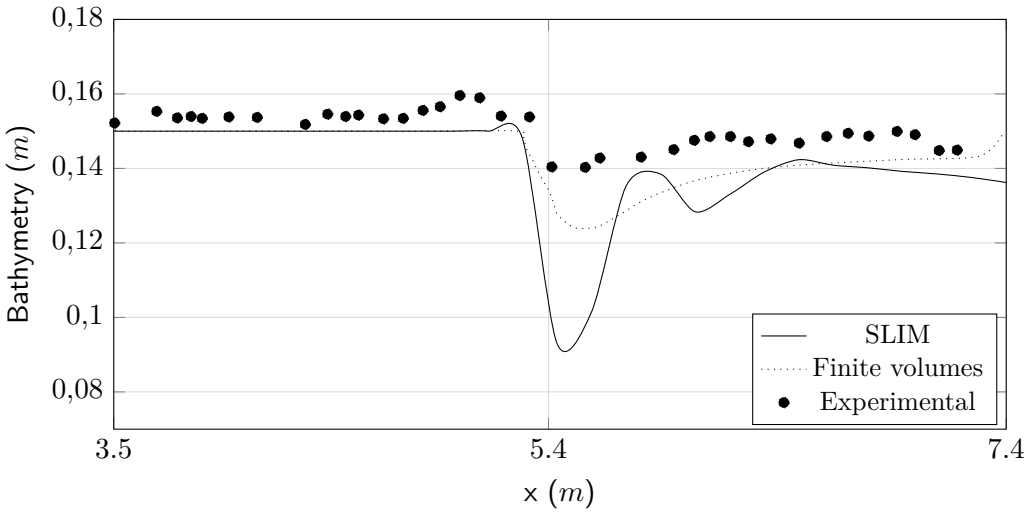

Figure 3.27 – Longitudinal bathymetry at $y = 0.15\ m$ after 600 s

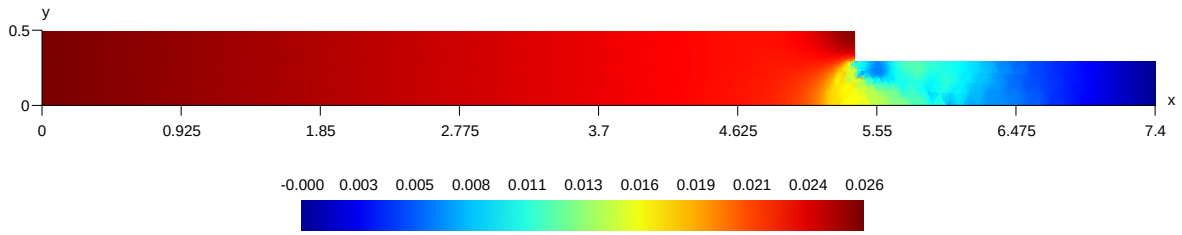

Figure 3.28 – Water height computed in SLIM after 600 s

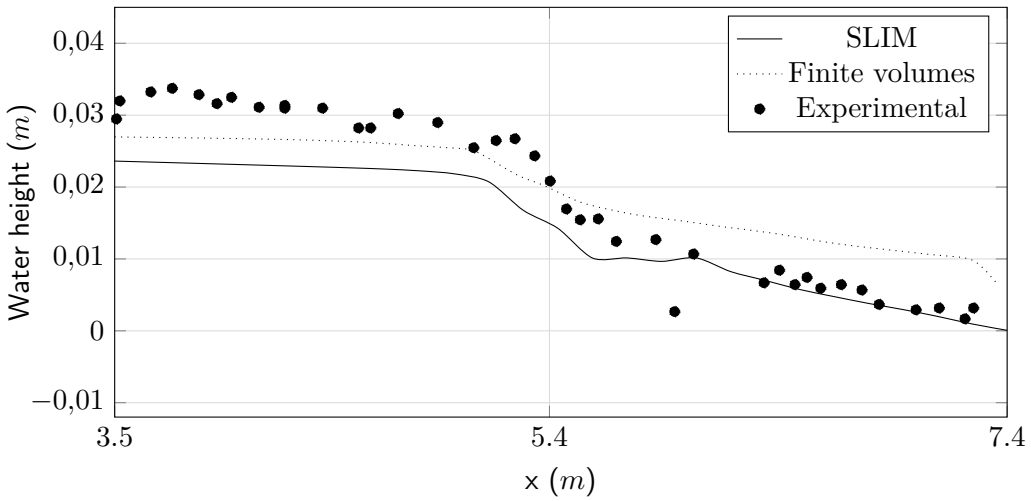

Figure 3.29 – Longitudinal water height at $y = 0.15\ m$ after 600 s

Chapter 4

Improvement

Throughout the text, it has been clear that the implementation of the Exner equation in SLIM could be improved. This chapter will go trough the main features that must be changed.

4.1 Adapted shallow water equations

When SLIM was designed, it was done assuming that the bathymetry would not vary over time. Particularly in the derivation of the shallow water equations, this assumption was made. This section will detail the changes that have to be carried out in order to take a variable bathymetry into account.

The complete development of the shallow water equations will not be done here. It has been detailed many times in the context of SLIM and the reader can refer, for example, to the thesis of Courgue [11] for an exhaustive explanation. The hypothesis of a constant bathymetry only appears when the equations are averaged over the depth. The development will thus start from the three dimensional shallow water equations.

The conservation of mass reads

$$\boldsymbol{\nabla}_h \cdot \boldsymbol{u} + \frac{\partial w}{\partial z} = 0 \tag{4.1}$$

Where ∇_h is the horizontal del operator $\nabla_h = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}$ u and v are the horizontal velocities and w is the vertical velocity.

These equations will be integrated over the depth. Starting with the conservation of mass

and applying the Leibniz integration rule, it becomes

$$\begin{aligned}\int_{-h}^{\eta} \boldsymbol{\nabla}_h \cdot \boldsymbol{u} dz + \int_{-h}^{\eta} \frac{\partial w}{\partial z} dz &= \boldsymbol{\nabla}_h \cdot \int_{-h}^{\eta} \boldsymbol{u} dz - \boldsymbol{u}(\eta) \cdot \boldsymbol{\nabla}_h \eta + \boldsymbol{u}(-h) \cdot \boldsymbol{\nabla}_h(-h) + w(\eta) - w(-h) \\ &= \boldsymbol{\nabla}_h \cdot (H\boldsymbol{U}) - \boldsymbol{u}(\eta) \cdot \boldsymbol{\nabla}_h \eta + \boldsymbol{u}(-h) \cdot \boldsymbol{\nabla}_h(-h) + w(\eta) - w(-h)\end{aligned} \tag{4.2}$$

Where $\boldsymbol{U}$ is the depth averaged velocity.

To find $w(\eta)$ and $w(-h)$, boundary conditions have to be written. Imagining a material particle with coordinates (x, y, z). The distance between that particle and the bottom is

$$\xi = -h - z \tag{4.3}$$

If that particle is on the bottom and always stays there, the distance between the particle and the bottom will always be zero

$$\frac{D\zeta}{Dt} = 0 \tag{4.4}$$

With $\frac{D\zeta}{Dt}$ the derivative over time following that particle. Per definition of the material derivative

$$\frac{D\zeta}{Dt} = \frac{\partial \zeta}{\partial t} + u(-h)\frac{\partial \zeta}{\partial x} + v(-h)\frac{\partial \zeta}{\partial y} + w(-h)\frac{\partial \zeta}{\partial z} = 0 \tag{4.5}$$

In local derivatives, $\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} = 0$ and $\frac{\partial z}{\partial z} = 1$. Injecting Eq. 4.3 in Eq. 4.5 thus gives

$$\frac{\partial(-h)}{\partial t} = \boldsymbol{u}(-h) \cdot \boldsymbol{\nabla}_h(-h) - w(-h) \tag{4.6}$$

The same can be done for the condition at water surface

$$\frac{\partial \eta}{\partial t} + \boldsymbol{u}(\eta) \cdot \boldsymbol{\nabla}_h(\eta) = w(\eta) \tag{4.7}$$

Replacing $w(\eta)$ and $w(-h)$ of equations 4.6 and 4.7 in Eq. 4.2, the conservation of mass becomes

$$\boldsymbol{\nabla}_h \cdot (H\boldsymbol{U}) + \frac{\partial H}{\partial t} = 0 \tag{4.8}$$

This is the first difference with SLIM where $\frac{\partial(-h)}{\partial t} = 0$ and thus omitted.

The three dimensional equation for the conservation of momentum is

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}_h \cdot \boldsymbol{u}\boldsymbol{u} + \frac{\partial w\boldsymbol{u}}{\partial z} + f\boldsymbol{e}_z \times \boldsymbol{u} = -\frac{1}{\rho_0}\boldsymbol{\nabla}_h p_{atm} - g\boldsymbol{\nabla}_h \eta + \frac{\partial}{\partial z}\left(\nu_t \frac{\partial \boldsymbol{u}}{\partial z}\right) \tag{4.9}$$

Where $f\boldsymbol{e}_z \times \boldsymbol{u}$ is the Coriolis term.

There are only two terms that will produce a time derivative when integrated over the depth. As a variable bathymetry only influences those terms, the other ones will not be developped here. The first term is the time derivative of the velocity. Applying the Leibniz rule, it gives

$$\int_{-h}^{\eta} \frac{\partial \boldsymbol{u}}{\partial t} dz = \frac{\partial H\boldsymbol{U}}{\partial t} - \boldsymbol{u}(\eta)\frac{\partial \eta}{\partial t} - \boldsymbol{u}(-h)\frac{\partial(-h)}{\partial t} \tag{4.10}$$

The second term is the vertical advection.

$$\int_{-h}^{\eta} \frac{\partial w\boldsymbol{u}}{\partial z} dz = w(\eta)\boldsymbol{u}(\eta) - w(-h)\boldsymbol{u}(-h) \tag{4.11}$$

Replacing $w(\eta)$ and $w(-h)$ of equations 4.6 and 4.7 in Eq. 4.11, this term becomes

$$\int_{-h}^{\eta} \frac{\partial w\boldsymbol{u}}{\partial z} dz = \boldsymbol{u}(\eta)\frac{\partial \eta}{\partial t} - \boldsymbol{u}(\eta)\boldsymbol{u}(\eta)\cdot\boldsymbol{\nabla}_h\eta + \boldsymbol{u}(-h)\frac{\partial(-h)}{\partial t} - \boldsymbol{u}(-h)\boldsymbol{u}(-h)\cdot\boldsymbol{\nabla}_h(-h) \tag{4.12}$$

When Eq. 4.10 is summed to Eq. 4.12 the terms $\boldsymbol{u}(-h)\frac{\partial -h}{\partial t}$ are cancelled. A variable bathymetry will thus have no effect on the conservation of momentum.

The equations to be implemented are

$$\begin{aligned} &\frac{\partial H}{\partial t} + \nabla\cdot(H\boldsymbol{u}) = 0 \\ &\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot(\nabla\boldsymbol{u}) + f\boldsymbol{e}_z\times\boldsymbol{u} = -g\nabla\eta - \frac{1}{\rho}\nabla p_{atm} + \frac{1}{H}\nabla\cdot(H\nu_t(\nabla\boldsymbol{u})) + \frac{\boldsymbol{\tau_s}-\boldsymbol{\tau_b}}{\rho H} \end{aligned} \tag{4.13}$$

The conservation of mass now computes the total water height H instead of the surface water height η. Fig. 4.1 shows the two references that can be taken to calculate the equation. Now that the Exner equation is implemented, the (H, z_b) reference makes more sense.

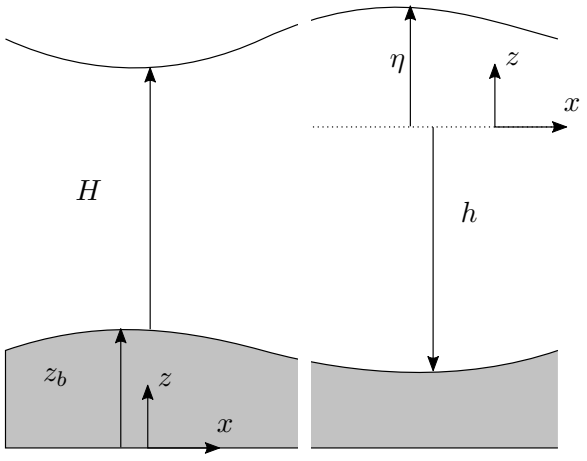

Figure 4.1 – References

The pressure term in the equation for the conservation of momentum has to be converted to the new reference. Going back to the pressure term in the three dimensional shallow water equations, it writes

$$-\frac{1}{\rho_0}\boldsymbol{\nabla}_h p \tag{4.14}$$

The assumptions about the pressure term that were made in SLIM are

- The pressure is hydrostatic $p = p_{atm} + \rho g(z_b + H - z)$
- The horizontal and vertical variation of the density is negligible

Integrating Eq.4.14 over the depth gives

$$\begin{aligned}\frac{1}{\rho}\int_{z_b}^{z_b+H} \boldsymbol{\nabla}_h \left[p_{atm} + \rho g(z_b + H - z)\right] dz &= \frac{H}{\rho}\boldsymbol{\nabla}_h p_{atm} + g\int_{z_b}^{z_b+H} \boldsymbol{\nabla}_h (z_b + H - z))dz \\ &= \frac{H}{\rho}\boldsymbol{\nabla}_h p_{atm} + g\boldsymbol{\nabla}_h \int_{z_b}^{z_b+H} (z_b + H - z))dz + gH\boldsymbol{\nabla}_h z_b \\ &= \frac{H}{\rho}\boldsymbol{\nabla}_h p_{atm} + g\boldsymbol{\nabla}_h \frac{H^2}{2} + gH\boldsymbol{\nabla}_h z_b\end{aligned} \tag{4.15}$$

In SLIM, the conservation of momentum is implemented in the non-conservative form. Dividing Eq. 4.15 by H and replacing the pressure term in Eq. 4.13 , the final equations are obtained. The equations are given in two dimensions, ∇_h can be replaced by ∇.

$$\begin{aligned}&\frac{\partial H}{\partial t} + \nabla \cdot (H\boldsymbol{u}) = 0 \\ &\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot (\nabla \boldsymbol{u}) + f\boldsymbol{e}_z \times \boldsymbol{u} = -g\nabla H - g\nabla z_b - \frac{1}{\rho}\nabla p_{atm} + \frac{1}{H}\nabla \cdot (H\nu_t(\nabla \boldsymbol{u})) + \frac{\boldsymbol{\tau_s} - \boldsymbol{\tau_b}}{\rho H} \\ &\frac{\partial z_b}{\partial t}(1 - \epsilon_0) + \nabla \cdot \boldsymbol{q_{sb}} = 0\end{aligned} \tag{4.16}$$

However, it could be interesting in SLIM to keep the same reference level. All the formulas are based on the (η, h) reference. Changing it could create confusion and lead to errors. It is possible to keep the same reference at the cost of computation time. The conservation of mass can be rewritten as

$$\frac{\partial \eta}{\partial t} + \frac{\partial h}{\partial t} + \nabla \cdot (H\boldsymbol{u}) = 0 \tag{4.17}$$

The time derivative of h can be calculated with Exner and the set of equations to be implemented is

$$\begin{aligned}
&\frac{\partial \eta}{\partial t} + \nabla \cdot (H\boldsymbol{u}) + \frac{1}{(1-\epsilon_0)} \nabla \cdot \boldsymbol{q_{sb}} = 0 \\
&\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot (\nabla \boldsymbol{u}) + f \boldsymbol{e}_z \times \boldsymbol{u} = -g \nabla \eta - \frac{1}{\rho} \nabla p_{atm} + \frac{1}{H} \nabla \cdot (H \nu_t (\nabla \boldsymbol{u})) + \frac{\boldsymbol{\tau_s} - \boldsymbol{\tau_b}}{\rho H} \\
&\frac{\partial h}{\partial t} (1-\epsilon_0) - \nabla \cdot \boldsymbol{q_{sb}} = 0
\end{aligned} \tag{4.18}$$

The computation time will be affected as the Exner equation has to be computed twice.

4.2 Unsteady approach

Models in which the Exner equation is coupled to the shallow water equations are known to produce more accurate results. [16]. As deduced in section 3.2, it would be interesting to see the results of a coupled model on the dome case. This section explains how it could be implemented in SLIM. As in the previous section, this study will focus on the terms that were modified. A detailed explanation of the implementation of the shallow water equations in SLIM can be found in [11].

4.2.1 Spatial discretization

Previous section proposed two sets of equations depending on which reference was taken. In this section, the implementation in the (H, z_b) reference (Eq. 4.16) will be detailed. This model has been chosen because it requires less computation. Anyhow, the second set of equations do not differ much from the first one and the implementation is similar.

Starting with the conservation of mass, the weak formulation writes

$$\sum_{e=1}^{N_e} \left(< \phi \frac{\partial H}{\partial t} >_e + < \phi \nabla \cdot Hu >_e \right) = 0 \tag{4.19}$$

As a reminder, N_e is the number of elements on the domain, ϕ is a shape function, the notation $< f >_e$ designs the integration of function f on the surface of element e and $\ll f \gg_e$ the integration of the function on the boundary of the element.

An integration by parts on the second term gives

$$\sum_{e=1}^{N_e} \left(< \phi \frac{\partial H}{\partial t} >_e + < \nabla \cdot (H\boldsymbol{u}\phi) >_e - < H\boldsymbol{u} \cdot \nabla \phi >_e \right) = 0 \tag{4.20}$$

Applying the divergence theorem

$$\sum_{e=1}^{N_e} \left(< \phi \frac{\partial H}{\partial t} >_e + \ll H u_n \phi \gg_e - < H\boldsymbol{u} \cdot \nabla \phi >_e \right) = 0 \tag{4.21}$$

Where u_n is the the component of the velocity, perpendicular to the boundary of the element.

The same can be done for the equation of momentum. The terms that have not been changed will be taken from Gourgue [11]. The new terms are

$$-g\nabla H - g\nabla z_b \tag{4.22}$$

Developing them in the same way, it gives

$$\begin{aligned}
&\sum_{e=1}^{N_e} (-g < \boldsymbol{\phi}\nabla \cdot H >_e -g < \boldsymbol{\phi}\nabla \cdot z_b >_e) \\
=&\sum_{e=1}^{N_e} (-g < \nabla \cdot (H\boldsymbol{\phi}) >_e +g < H \cdot \nabla\boldsymbol{\phi} >_e -g < \nabla \cdot (z_b\boldsymbol{\phi}) >_e +g < z_b \cdot \nabla\boldsymbol{\phi} >_e) \\
=&\sum_{e=1}^{N_e} (-g \ll H\phi_n \gg_e +g < H \cdot \nabla\boldsymbol{\phi} >_e -g \ll z_b\boldsymbol{\phi} \gg_e +g < z_b \cdot \nabla\phi_n >_e)
\end{aligned} \tag{4.23}$$

$\boldsymbol{\phi}$ is now a vector because the momentum equation has two components. One in the x direction that will compute $\frac{\partial u}{\partial t}$ and the other in the y direction that will compute $\frac{\partial v}{\partial t}$. In the first equation, the vector $\boldsymbol{\phi}$ will thus be $(\phi, 0)$ and in the second equation $(0, \phi)$. ϕ_n is the component of $\boldsymbol{\phi}$ perpendicular to the boundary.

Including the new terms in the weak formulation of the conservation of momentum gives

$$\begin{aligned}
&\sum_{e=1}^{N_e} \left(< \boldsymbol{\phi} \cdot \frac{\partial \boldsymbol{u}}{\partial t} >_e + \ll u_n \boldsymbol{u} \cdot \boldsymbol{\phi} \gg_e - < \boldsymbol{u} \cdot \nabla \cdot (\boldsymbol{\phi}\boldsymbol{u}) >_e + < \boldsymbol{\phi} \cdot f(\boldsymbol{e_z} \times \boldsymbol{u}) >_e \right) \\
&= \sum_{e=1}^{N_e} \left(\frac{1}{\rho} < \boldsymbol{\phi}\nabla p_{atm} >_e -g \ll H\phi_n \gg_e +g < H \cdot \nabla\boldsymbol{\phi} >_e -g \ll z_b\boldsymbol{\phi} \gg_e +g < z_b \cdot \nabla\phi_n >_e \right. \\
&\left. + \ll \nu \frac{\partial \boldsymbol{u}}{\partial n} \cdot \boldsymbol{\phi} \gg_e - < \nu(\nabla \boldsymbol{u}) : (\nabla\boldsymbol{\phi})^T >_e + < \frac{\nu}{H}(\nabla H) \cdot (\nabla \boldsymbol{u}) \cdot \boldsymbol{\phi} >_e + < \frac{\boldsymbol{\tau}_s - \boldsymbol{\tau}_b}{\rho H} \cdot \boldsymbol{\phi} >_e \right)
\end{aligned} \tag{4.24}$$

Where $\frac{\partial \boldsymbol{u}}{\partial n}$ is the derivative of the velocity on the boundary.

The last equation to implement is the Exner equation. As q_{sb} is dependent on the velocity, it has to be included in the matrix. Two formulas were proposed in this thesis. The Grass model (Eq. 2.25) and the Meyer-Peter-Muller model (Eq. 2.23) For the sake of simplicity, the Grass model will be implemented. The Exner equation than reads

$$\frac{\partial z_b}{\partial t} + \frac{1}{(1-\epsilon_0)} \nabla \cdot (A\boldsymbol{u}\|\boldsymbol{u}\|^{m-1}) = 0 \tag{4.25}$$

Repeating the same process

$$\begin{aligned}
&\sum_{e=1}^{N_e} \left(< \phi \frac{\partial z_b}{\partial t} >_e + \frac{A}{1-\epsilon_0} < \phi \nabla \cdot \boldsymbol{u}\|\boldsymbol{u}\|^{m-1} >_e \right) = 0 \\
=&\sum_{e=1}^{N_e} \left(< \phi \frac{\partial z_b}{\partial t} >_e + \frac{A}{1-\epsilon_0} < \nabla \cdot (\boldsymbol{u}\|\boldsymbol{u}\|^{m-1}\phi) >_e - \frac{A}{1-\epsilon_0} < \boldsymbol{u}\|\boldsymbol{u}\|^{m-1} \cdot \nabla\phi >_e \right) = 0 \\
=&\sum_{e=1}^{N_e} \left(< \phi \frac{\partial z_b}{\partial t} >_e + \frac{A}{1-\epsilon_0} \ll \|\boldsymbol{u}\|^{m-1} u_n \phi \gg_e - \frac{A}{1-\epsilon_0} < \boldsymbol{u}\|\boldsymbol{u}\|^{m-1} \cdot \nabla\phi >_e \right) = 0
\end{aligned} \tag{4.26}$$

The values of H, u, v and z_b are evaluated at the nodes of each element and approximated over the whole domain as follow

$$\begin{aligned} H^h(x,y) &= \sum_{i=i}^{N_n} H_i \phi_i(x,y) \\ \boldsymbol{u}^h(x,y) &= \sum_{i=i}^{N_n} \boldsymbol{u_i} \phi_i(x,y) \\ z_b^h(x,y) &= \sum_{i=i}^{N_n} z_{b,i} \phi_i(x,y) \end{aligned} \tag{4.27}$$

This is exactly the same procedure as in Section 2.3.1. Replacing the unknowns by their approximations in Eq. 4.21, Eq. 4.24 and Eq. 4.26 and using them with the shape functions ϕ_j associated to each node gives $4N_n$ equations.

$$\begin{aligned} \sum_{i=i}^{N_n} \frac{\partial H_i}{\partial t} \iint_\Omega \phi_i \phi_j dx dy \, \hat{\boldsymbol{e}}_{\boldsymbol{j}} &= f_H(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_j) \\ \sum_{i=i}^{N_n} \frac{\partial u_i}{\partial t} \iint_\Omega \phi_i \phi_j dx dy \, \hat{\boldsymbol{e}}_{\boldsymbol{j}} &= f_u(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_j) \\ \sum_{i=i}^{N_n} \frac{\partial v_i}{\partial t} \iint_\Omega \phi_i \phi_j dx dy \, \hat{\boldsymbol{e}}_{\boldsymbol{j}} &= f_v(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_j) \\ \sum_{i=i}^{N_n} \frac{\partial z_{b,i}}{\partial t} \iint_\Omega \phi_i \phi_j dx dy \, \hat{\boldsymbol{e}}_{\boldsymbol{j}} &= f_{z_b}(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_j) \end{aligned} \tag{4.28}$$

In the matrix form, this is

$$\boldsymbol{A} \cdot \frac{\partial \boldsymbol{x}}{t} = \boldsymbol{b} \tag{4.29}$$

Where x is a vector of size $4N_n$

$$\boldsymbol{x} = \begin{bmatrix} H_1 \\ \vdots \\ H_{N_n} \\ u_1 \\ \vdots \\ u_{N_n} \\ v_1 \\ \vdots \\ v_{N_n} \\ z_{b,1} \\ \vdots \\ z_{b,N_n} \end{bmatrix}$$

A is a $4N_n \times 4N_n$ matrix

$$\begin{bmatrix}
\iint \phi_1\phi_1 dxdy & \dots & \iint \phi_1\phi_{N_n} dxdy & & & & & & & & \\
\vdots & \ddots & \vdots & & 0 & & & 0 & & 0 & \\
\iint \phi_{N_n}\phi_1 dxdy & \dots & \iint \phi_{N_n}\phi_{N_n} dxdy & & & & & & & & \\
 & & & \iint \phi_1\phi_1 dxdy & \dots & \iint \phi_1\phi_{N_n} dxdy & & & & & \\
 & 0 & & \vdots & \ddots & \vdots & & 0 & & 0 & \\
 & & & \iint \phi_{N_n}\phi_1 dxdy & \dots & \iint \phi_{N_n}\phi_{N_n} dxdy & & & & & \\
 & & & & & & \cdot & \dots & \cdot & & \\
 & 0 & & & 0 & & \vdots & \ddots & \vdots & 0 & \\
 & & & & & & \cdot & \dots & \cdot & & \\
 & & & & & & & & \iint \phi_1\phi_1 dxdy & \dots & \iint \phi_1\phi_{N_n} dxdy \\
 & 0 & & & 0 & & & 0 & \vdots & \ddots & \vdots \\
 & & & & & & & & \iint \phi_{N_n}\phi_1 dxdy & \dots & \iint \phi_{N_n}\phi_{N_n} dxdy
\end{bmatrix}$$

And b is a vector of size $4N_n$ containing all the other terms

$$\boldsymbol{B} = \begin{bmatrix}
f_H(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_1) \\
\vdots \\
f_H(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_{N_n}) \\
f_u(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_1) \\
\vdots \\
f_u(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_{N_n}) \\
f_v(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_1) \\
\vdots \\
f_v(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_{N_n}) \\
f_{z_b}(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_1) \\
\vdots \\
f_{z_b}(H_i, u_i, v_i, z_{b,i}, \phi_i, \phi_{N_n})
\end{bmatrix}$$

Due to the discontinuities between the elements inherent to the discontinuous Galerkin method, H, z_b and u_n are defined two times on the interface. To compute the integrals on the boundary of the elements, one single value is needed. Fluxes will thus have to be implemented to find the physical value on the interface. Various possible schemes exists. Particularly the approximate Riemann solver proposed by Audusse et al. [1] could be used as it is very close the method currently used in SLIM.

4.2.2 Temporal integration

The last step is to integrate Eq. 4.29 over the time. As $\boldsymbol{A}$ is not time dependent, the equation can be written as

$$\frac{\partial \boldsymbol{A} \cdot \boldsymbol{x}}{\partial t} = \boldsymbol{b} \tag{4.30}$$

The first method that comes to mind to solve this equation, is by applying the derivative for Δt.

$$\boldsymbol{A} \cdot \boldsymbol{x}^{t+\Delta t} - \boldsymbol{A} \cdot \boldsymbol{x}^{t} = \Delta t \boldsymbol{b}^{t} \tag{4.31}$$

Where the superscript t denotes that the matrix is evaluated at time t, which has been previously computed and $t + \Delta t$ that it is evaluated one time step further.

This is the Euler explicit method which is commonly used in SLIM. This method does not require much computing time but is unstable if Δt is too big. This comes because $\boldsymbol{b}$ depends on $\boldsymbol{x}$ and can be enhanced by using higher order explicit methods or implcit methods. In SLIM, the Runge-Kutta methods are used. The general expression is

$$\begin{aligned} \boldsymbol{A} \cdot \boldsymbol{x}^{t+1} =& \boldsymbol{A} \cdot \boldsymbol{x}^{t} + \Delta t \sum_{j=1}^{n} (w_j K_j) \\ & K_1 = \boldsymbol{b}^{t}(\boldsymbol{x}) \\ & \vdots \\ & K_n = \boldsymbol{b}^{t+\alpha_n \Delta t}(\boldsymbol{x} + \sum_{m=1}^{n-1} (\beta_{nm} \Delta t K_m)) \end{aligned} \tag{4.32}$$

Where n is the order of the method, α, β and w are coefficients chosen to reach a precision. If $n = 1$ and $w_1 = 1$, expression 4.31 is found.

In SLIM, the most used explicit method is from the second order and is also called the Heun method. The coefficients are

$$\begin{aligned} w_i &= 0.5 \\ \alpha_i &= 1 \\ \beta_{i,j} &= 1 \end{aligned}$$

The most used implicit method is also from the second order and the coefficients are

$$w_1 = \frac{\sqrt{2}}{2} \qquad w_2 = 1 - \frac{\sqrt{2}}{2}$$

$$\alpha_1 = 1 - \frac{\sqrt{2}}{2} \qquad \alpha_2 = 1$$

$$\beta_{1,1} = \beta_{2,2} = 1 - \frac{\sqrt{2}}{2} \qquad \beta_{1,2} = 0 \qquad \beta_{2,1} = \frac{\sqrt{2}}{2}$$

CHAPTER 5

Using SLIM

I spent an important part of my master thesis on understanding the structure of SLIM and how to use it. For now, no document exists on the functioning of SLIM and learning everything by reading code and asking questions can take a lot of time. This chapter aims to explain the basics of SLIM and help to understand how it works. However, it should be taken with caution. As the program is still in development, it is always changing. New features may be added thus outdating some information contained in this document.

Throughout the chapter, the following notation will be used. A grey background without frame represents a command line execution.

```
example of a command line execution
```

A grey background with a black frame represents code lines writed in a file.

```
example of code
```

The <> delimits a parameter that will be introduced by the user.

5.1 Structure of SLIM

Currently SLIM is made of two layers. The core program is written in c++. The functions of the core program will basicaly : assemble the terms of the equations using the discontinous Galerkin (dg) procedure, solve the system and integrate the result over time. The second layer is written in python and acts as a user interface.

The advantage of this implementation is that the most time consuming functions are programmed in c++ which is known to be fast. Python on the other hand, offers a user friendly

interface. It is easier to use and as an interpreted language it does not need to be compiled. This makes it more effective when the program is modified on regular basis.

If the purpose is to build a test case and solve the shallow water equations with or without a tracer, working with the python interface will be enough. The interface even allows interaction with the parameters making it possible to solve new equations in an uncoupled way. Limitations will start to appear when a variable has to be spatially integrated or derived. In that case the discontinuous Galerkin scheme is needed and the core program will have to be modified.

SLIM exists for Linux and for Windows. When working with the python interface only, the Windows version will do the job. Anyway, for non Linux users, the Windows version is a good starting point for using SLIM and familiarize with the python interface.

When the c++ code has to be modified it can be interesting to go over to the Linux version. A first advantage is that the compiler is natively present in Linux making the code easy to modify and to compile. However, this could also be done on Windows using a third party software. A second advantage is that the whole SLIM team is working with Linux. Having the same tools makes it easier to get help or to work together. SLIM works perfectly on a virtual machine.

5.2 Installation

5.2.1 Windows

To use SLIM on Windows, the following programs are needed

- A finite element mesh generator
- The Windows version of SLIM
- A post precessing software to see the results on the mesh

For the mesh generation and the post processing, the GMSH software developped by Christophe Geuzaine et Jean-François Remacle [9] can be used.

Optionality, the following programs might be useful as well

- A good python development environment
- A software capable of reading NETcdf files

On Windows, Spyder can be used as IDE and MATLAB can load NETcdf files by using the `ncread` function.

5.2.2 Linux

On Linux, the python interpreter is natively integrated. Nevertheless, some packages have to be installed. This is done by opening a terminal and run

```
sudo apt-get install cmake-curses-gui
sudo apt-get install gfortan
sudo apt-get install libblas-dev
sudo apt-get install liblapack-dev
sudo apt-get install git
sudo apt-get install swig
sudo apt-get install petsc
sudo apt-get install libnetcdf-dev
sudo apt-get install libopenmpi-dev
```

The source code can now be downloaded [1]. In the main folder, a new directory can be created and named `build`. This folder will contain all the information that the computer needs to compile the SLIM code. Particularly, a 'make' file has to be generated.

The 'make' file is generated by opening a terminal in the `build` directory (`Documents/<installation-directory>/build/`) and run

```
ccmake ..
```

An interface will open with the message: EMPTY CACHE. The file has now to be configured by typing c. An error will probably occur. This is because more components are necessary to run all features of SLIM but the ones that are now installed are enough to run the basic features. The previous interface can be reached by pressing e. The components causing the error can now be desactivated by navigating with the arrow keys and by by pressing enter to put them on OFF. This has to be done for `ENABLE_GEOTIFF, ENABLE_GOTM, ENABLE_LIM, ENABLE_METIS, ENABLE_NUMPS, ENABLE_PNETCDF and ENABLE_UNREF`. It is now necessary to configure (c), exit (e), configure a **second time** (c) and exit again (e). When this is done, a new option has appeared in the interface: generate (g). By pressing g, the make file will finally be generated.

The code can now be compiled by running in the same terminal

(`Documents/<installation-directory>/build/`)

```
make
```

It can be interesting to modify the .bashrc file to run SLIM without having to reach the build directory. Therefore, open a terminal in the home directory and run

```
sudo gedit .bashrc
```

The next lines can be copy pasted in that file. If the path to the build directory is different (in this case `Documents/dg/build`), it has to be replaced.

[1] The SLIM source code is freely available on gitHub: `https://git.immc.ucl.ac.be/dg/dg`

```
#SLIM
export PATH=$HOME/Documents/dg/build:$PATH
DGBUILD=$HOME/Documents/dg/build
export LD_LIBRARY_PATH=$DGBUILD:$LD_LIBRARY_PATH
export PYTHONPATH=.:$DGBUILD:$DGBUILD/dgpy/scripts:$PYTHONPATH
```

The file can finally be saved and closed.

Eventually installing Spyder for a good python development environment and GMSH as mesh generator and for the post-processing can be interesting

```
sudo apt-get install spyder
sudo apt-get install gmsh
```

5.3 Use

First of all, a mesh should be generated. If GMSH is used, all the information can be found on their website `http://gmsh.info/`. The .geo file used to generate the mesh of the compound channel can be found in the annexes. When the mesh is created, it is important to apply a physical tag to the boundaries. This will be used by SLIM to know which boundary corresponds to what (inlet, outlet, wall, ...).

The next step is to create the python script. To build a simple testcase, all the necessary functions can be found in the `slim.py` or `slimPre.py`. Under Windows, these files are located in the `lib/` directory of the installation files. Under Linux, they are in the `dgpy/scripts/` directory. An example of the script used to build the compound channel test case can be found in the annex. As said in the beginning of this chapter, the functions are constantly changing and the parameters that are given in the example are probably out of date. The example is given to provide the structure of the code and the parameters taken by those functions should always be checked in `slim.py`.

In the script, NETcdf files will often be used. A little word of explanation is needed in order to understand how they work and how to create them. They allow to store vectors in a file so that they can be reused later. A NETcdf file in SLIM is created as follow

```
slimPre.write_file('<output_NETcdf_file.nc>', region=<region>, time=<time>,
    data=[('<NETcdf_variable_1>', <vector_1>),('<NETcdf_variable_2>',
    <vector_2>)])
```

As the NETcdf files can store various vectors, two names have to be given. `<output_NETcdf_file.nc>` is the name of the file and has a .nc extension, whereas `<NETcdf_variable>` is the name under which the vector is stored in the NETcdf file. `<region>` and `<time>` are objects created by SLIM and are explained further. When a function takes a NETcdf variable as parameter the, syntax is the following

```
<function>(('<output_NETcdf_file.nc>', '<NETcdf_variable>'))
```

5.3.1 Making a script

An example of a script can be found in the Appendice. The script will generally follow the next structure:

Importing the necessary libraries In order to use the functions defined in `slim.py` and `slimPre.py` these files have to be imported.

```
import slimPre
import slim
import numpy
```

numpy is a python library providing scientific computing [2] (e.g. matrices). [24]

Reading the mesh The program will read the mesh and create a *mesh* object usable by SLIM. The mesh file is the one generated by the mesh generator and has the .msh extension.

```
<mesh> = slimPre.Mesh(<meshfile.msh>)
```

Creating a region object This will create a *region* object containing, among others, the coordinates of the nodes.

```
<region> = slimPre.Region(<mesh>)
```

Creating a time object This will create a *time* object containing, among others, the initial time, the final time and the time step.

```
<time_vector> =
    numpy.arange(<t_initial>,<t_final>+2*<dt>,<dt>,numpy.float64)
<time> = slimPre.Time(time_vector=<time_vector>)
```

The initial time, the final time and the time step must be given in seconds.

Defining the bathymetry Now that the program has the coordinates of each node of the mesh, the bathymetry can be defined. This is done by creating a vector of size n, where n is the number of nodes. The vector contains the depth from a reference level at each node. This depth is the h of Fig. 4.1. A good practice is to take the reference level at the water surface to have a biggest possible h and a smallest possible η. Normally the result should not be affected but if the developer forgot to sum η to the bathymetry to represent the total water level, a bigger h will give better results.

[2] More information available on the numpy website: `http://www.numpy.org/`

```
<bathymetry> = <function_defining_the_bathymetry>(<region>.coordinates)
```

Where `<region>.coordinates` is a $n \times 3$ array containing the x, y and z coordinates of the nodes. The bathymetry is than stored in a NETcdf file using the write file function described at the beginning of this section. As the bathymetry depends on the region and not on the time, only the *region* will be given and *time* will be set to `None`.

Creating the domain for the equations To solve the equations, SLIM needs on object of type *domain*. This is created from the meshfile and the NETcdf file containing the bathymetry.

```
<domain> = slim.Domain(<meshfile.msh>,
    ('<output_NETcdf_bathymetry.nc>','<bathymetry_variable>'))
```

Defining the initial conditions Defining the initial conditions is exactly the same process as the bathymetry but instead of giving the depth at each point, it is now the initial state of the flow that will be given. If the shallow waters equations are computed, an initial condition has to be given for η, and for both velocities. If a tracer is added, it will also need an initial condition. Once again, a NETcdf file will be created containing the different vectors.

Defining the boundary conditions A vector will be created containing the values of the parameter associated to each time step. The boundary conditions can be set on: the discharge, η, the velocities or a combination of the last two. In the latter case, SLIM will take a physically possible mean of given η, u and v. Once again, a NETcdf file will be created containing the different vectors. The *region* will now be set to `None`, while the *time* will be the one created at the fourth step

Defining the shallow water equations An object of type *ShallowWater2D* will now be created. This will initialize the system of the basic shallow water equations.

$$\begin{aligned} &\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla \cdot (Hu)} = \mathbf{0} \\ &\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{t}} + \boldsymbol{u \cdot (\nabla u)} + f e_z \times u = \boldsymbol{-g \nabla \eta} - \frac{\mathbf{1}}{\boldsymbol{\rho}} \boldsymbol{\nabla p_{atm}} + \frac{1}{H} \nabla \cdot (H \nu_t (\nabla u)) + \frac{\tau_s - \tau_b}{\rho H} \end{aligned} \tag{5.1}$$

```
<equation_1> = slim.ShallowWater2d(domain = <domain>, temporal_scheme =
    "<temporal_scheme>", initial_time=<t_initial>, final_time=<t_final>)
```

Doing this will only include the bold terms of Eq. 5.1. The domain is the *domain* object that was created previously and the temporal scheme can be `"implicit"` (second order Runge-Kutta) or `"explicit"` (Euler explicit).

To add the coriolis term ($fe_z \times u$) the following function must be used

```
<equation_1>.set_coriolis(<coriolis>)
```

Where `<coriolis>` is a NETcdf variable with the coriolis term at each point of the domain.

The viscous term $\left(\frac{1}{H}\nabla \cdot (H\nu_t(\nabla u))\right)$ is added with

```
<equation_1>.set_viscosity(<mode>, <smagorinsky_coefficient>,
    <constant_viscosity>)
```

Where mode will define wether the Eddy viscosity ν is constant over the domain (`<mode> = "constant"`) or calculated with the Smagorinsky [3] model (`<mode> = "smagorinsky"`)

The surface stress τ_s is added with

```
<equation_1>.set_wind_stress(<mode>, <wind_x>, <wind_y>, <air_density>)
```

Where mode defines if the speed of the wind is given (`<mode> = "speed"`) or the surface stress due to the wind is given (`<mode> = "stress"`). The other terms are NETcdf variables with the wind at each point of the domain. Wind is given in $m\,s^{-1}$ and stress in $kg\,m^{-1}\,s^{-1}$. If the speed mode is chosen, the density of the air also has to be defined. It is a constant.

The bottom friction τ_b is added with

```
<equation_1>.set_dissipation(<mode>, <coefficient>)
```

Three parametrization of the bottom shear stress are possible: `<mode> = "linear"`, `<mode> = "quadratic"` and `<mode> = "manning"`. The linear mode computes $\tau_b = \gamma\rho H\boldsymbol{u}$, where γ is the coefficient that has to be introduced. The quadratic mode evaluates $\tau_b = C_d\frac{\rho\|\boldsymbol{u}\|\boldsymbol{u}}{H}$, where C_d is the coefficient and the last mode is the Chezy-Manning-Strickler parametrization : $\tau_b = \rho g n^2\frac{\|\boldsymbol{u}\|^2}{H^{1/3}}$, where n is the Manning coefficient.

Setting the initial and the boundary conditions The NETcdf files containing the initial and boundary conditions generated previously will now be applied to the equations.

```
<equation_1>.set_initial_condition(<eta_initial>, <ux_initial>,
    <uy_initial>)
<equation_1>.set_boundary_open('<physical_tag>', discharge=<discharge>,
    eta=<eta>, ux=<ux>, uy=<uy>)
<equation_1>.set_boundary_coast('<physical_tag_wall>', <slip>)
```

[3] See Pham Van [27]

`physical tag`, designs the name that was applied to the boundaries when the mesh was generated. The `<slip>` is a boolean. If it is true, the tangential velocity will not be affected at the wall. If it is false, the tangential velocity will be set to zero at the wall. All the other parameters, are the NETcdf variables that were created earlier. Various open boundaries can be created. If the user wants, for example, to impose a discharge upstream and a water height downstream, he needs to apply a different physical tag to those boundaries and set two open boundary conditions. Like explained before, the discharge cannot be set together with the velocities or the water height. This would impose a double condition as these variables are already included in the discharge. A good practice is to set the water height and the velocities together if both parameters are known.

Defining the tracer equation Defining the tracer is exactly the same process as for the shallow water equations. The equation that will now be implemented is

$$\frac{\partial(\boldsymbol{HC_{ss}})}{\partial \boldsymbol{t}} = -\boldsymbol{\nabla \cdot (HC_{ss})} + \nabla(Hk\nabla C_{ss}) + (E - D) \tag{5.2}$$

Therefore, the following function is used.

```
<equation_2> = slim.ShallowWaterTracer2d(domain = <domain>,
    temporal_scheme = "<temporal_scheme>", hydro_sol = <equation_1>,
    initial_time = <t_initial>, final_time = <t_final>)
```

The domain, the temporal scheme and the times are the same as for the shallow water. Note that the initial time for the tracer can be later than the initial time for the shallow water. This is interesting if the flow has to reach an equilibrium before injecting the tracer. Hydro sol refers to the shallow water equation.

Like with the shallow water equation, this will only implements the basic terms that are bolted in Eq. 5.2. Adding the diffusive term $\nabla(Hk\nabla C_{ss})$ is done with

```
<equation_2>.set_diffusivity(<mode>, <okubo_coefficient>,
    <constant_diffusivity>)
```

There are two possible parametrization for the diffusivity coefficient k: `<mode>="okubo"` and `<mode>="constant"`. The constant mode will set the value of `<constant_diffusivity>` $[m^2/s]$ over the whole domain. The okubo parametrization [4] will set the diffusivity as

$$k = c_k \Delta^{1.15} \tag{5.3}$$

Where c_k is the okubo coefficient in $[m^{0.85}/s]$ and Δ is the characteristic length of the mesh.

[4] more information is available in Okubo [23]

If the tracer is a sediment, erosion and diffusion will occur. The $(E - D)$ term is added with

```
<equation_2>.set_sediment(<initial_bottom_concentration>, ... )
```

The set sediment equation is not available in all the SLIM versions and can take a lot of different parameters, depending on how the erosion is calculated. If this function has to be used, the user should read the signature of the function in `slim.py`. The only parameter that will always be asked is the `<initial_bottom_concentration>`. It is a NETcdf variable which defines how much sediment is available at each point of the mesh. The values are in $[kg\,m^{-2}]$ and it represents the total mass of sediment per unit surface of the bed.

Setting the initial and the boundary conditions The boundary conditions are set the exact same way as for the shallow water equation.

```
<equation_2>.set_initial_condition(<initial_concentration>)
<equation_2>.set_boundary_open('<physical_tag_wall>', <concentration>)
<equation_2>.set_boundary_coast('<physical_tag_wall>')
```

The concentrations are NETcdf variables and their unit is $[g\,l^{-1}]$. A source term can also be added if, for example, the tracer is injected in the middle of the domain.

```
<equation_2>.set_source(tracer_source = <source>)
```

Source is a NETcdf variable with the injected concentration per second $[gl^{-1}s^{-1}]$ at each node of the mesh.

Initializing the temporal solver The temporal solver is initialized by creating a *loop* object

```
<loop> = slim.Loop(maximum_time_step=<dt>, export_time=<export_time_step>)
```

The maximum time step will be the time step set in the *time* object. In an explicit scheme, SLIM could use a smaller time step for the computations so that the temporal scheme remains stable. The `<export_time_step>` is the time step at which a file with the results will be produced. Those files will be used by the post-processing software to represent the result. Its value will generally be the same as the normal time step. It can be interesting, however, to have a bigger export time step if the normal time step is really small. There will be less result files, making it easier for the post processing software to show them.

Adding the equations to the solver The equations that have to be solved can now be added to the solver.

```
<loop>.add_equation(<equation_1>)
<loop>.add_equation(<equation_2>)
```

Running the simulation Finally, the simulation is started with

```
<loop>.run()
```

To end the program properly and avoid an error message, the following line has to be added

```
slimPre.exit(0)
```

The results will be stored in a directory called output. To view the results, open (file → open) the .msh file with GMSH and merge (file → merge) the .idx file of the needed variable.

5.3.2 Windows

Running SLIM consists in running the .exe file and referring to the test-case script.

5.3.3 Linux

On Linux, running SLIM is done by opening a terminal in the same directory as the test-case script and executing :

```
rundgpy  <script-name>
```

If the c++ code has changed since the last execution, priory the make command has to be run in the build directory.

5.4 Working with DOF's and functors

If the user wants to go further. It is possible to adapt the SLIM code from the python interface. If, for example, the parametrization of the bottom friction has to be changed or a new parametrization for the diffusivity wants to be added, this can be done by changing the `slim.py` file. In order to do that, it is important to understand how the python interface interacts with the core program. This section explains the basics of this interaction.

All the data transferred from the python interface to the c++ interface are objects called functors. Functors are functions defined over the whole domain. In order to create them DOF's will be used. A DOF can be seen as an array containing the value of a variable at the nodes of the mesh.

To make the examples hereunder work, some scripts have to be imported.

```
import dgpy
import slim_private
import numpy
```

5.4.1 Creating a DOF

A DOF is really similar to an array of size $(M \times N)$, where M are the number of nodes of the domain and N the number of variables that the DOF carries. For example, a DOF with the results of the shallow water equations would have 3 columns. One for the water level and one for each component of the velocity.

Creating a DOF in the python interface goes trough two steps. The first step consists in initiating the DOF

```
<dof> = dgpy.dgDofContainer(<group>, <int>)
```

Where int is the integer with the number of variables that the DOF will hold and group is an object that provides the information of which node corresponds to which element.

The second step consists in filling it with data. This can be done by different means. A first possibility is by filling it with an existing DOF

```
<dof> = <existing dof>
```

another possibility by loading data from a netcdf file

```
slim_private._load(<dof>, <NETcdf-variable>)
```

a last way to fill the DOF is by interpolating a functor.

```
<dof>.interpolate(<functor>)
```

5.4.2 Creating a functor

A functor is created from a DOF by calling.

```
<functor> = <dof>.getFunction()
```

It can also be set constant

```
<functor> = dgpy.functionConstant(<constant>)
```

or by loading data from a netcdf file

```
<functor> = slim_private._load_function(<NETcdf_variable>, <group>)
```

5.4.3 Operations

What differentiates a DOF from an array is that the value of a DOF cannot be called easily. An operation like adding two DOF's is quite complicated. That does not mean it is impossible. The way to do it, is by creating a lambda function and realizing the operations inside of this lambda function. Here is an example on how to add two DOF's

```
def <function>(cm, <funtor 1>, <funtor 2>):
    <array 1> = cm.get(<funtor 1>)
    <array 2> = cm.get(<funtor 2>)
    <solution> = numpy.zeros(<array 1>.shape)
    <solution>[:] = (<array 1>[:]+<array 2>[:])
    return <solution>
<functor of the solution> = dgpy.functorNumpy(lambda cm : <function>(cm, <dof
    1>.getFunction(), <dof 2>.getFunction()))
```

Doing this will initiate the function but not yet calculate it. The function will only be calculated when the functor is interpolated on a DOF.

```
<dof of the solution>.interpolate(<functor of the solution>)
```

This is important to notice as it has no sense to put the lambda function in a if condition. Doing so will not initiate the lambda function if the condition is not respected. The same holds in a for loop.

Other operations are already implemented in SLIM

```
<dof1>.axpy(<dof2>,<a>)
```

Adds $a \times dof2$ to $dof1$ where a is a number.

```
<dof>.scale(<a>)
```

Multiplies the DOF with a.

```
<new dof>.multiply(<dof1>,<dof2>)
```

Multiplies dof1 with dof2 term by term and saves it in a new dof.

5.5 Common mistakes

If an error occurs while running the test-case python file, it could probably be one of the following common made errors

- Most of the parameters are taken on the whole domain and not as a constant. A common made error is to give a scalar as parameter where it should be a vector. This error is made while creating the NETcdf file.

- If a parameter of the equations has not been defined, it will be equal to zero. (e.g. Forgetting to define the dissipation will cause a model really difficult to stabilize.)
- The mesh generated in Windows is not exactly the same as the one generated in Linux. If an error occurs, it can be interesting to try to regenerate the mesh.

Chapter 6

Conclusion

The aim of this thesis was to include a module capable of computing morphological changes of the bed in SLIM. SLIM would thus be able to model a new type of problems like dam breaches or the formation of shoals in estuaries. In order to reach this objective, different steps were taken.

A first step was made by updating the bathymetry in function of the erosion and settling of suspended sediments. The model was tested on a compound channel to validate this approach. SLIM reproduces well the settling pattern of sediments but as the changes on the bathymetry are minor, the influence on the flow is not visible. The results were compared with data from previous computations using Telemac. It showed an important difference in diffusivity between both models, probably due to the use of a different scheme. Interesting data was collected for future comparisons.

The second consisted in including the effects of the bedload on the morphology of the bed. This was done by implementing the Exner equation in SLIM. As a basic implementation, the uncoupled approach was chosen. It was validated using two cases: one purely mathematical and the second experimented in a flume. Both cases showed promising results.

The first case reproduced well the characteristic star pattern but developed oscillations. In previous studies, small kinks were reported when using the uncoupled approach. These instabilities disappeared with a coupled approach. Other studies showed that the results are highly dependent on the used scheme. It has been deduced that the oscillations could probably be eliminated by modifying the scheme and by using the coupled approach.

The second case lacked to give a good representation of the reality. Particularly, the phenomena happening at the angle are not well represented. Bernard's model showed the same inadequacy letting to think that the problem lies in the model more than in the implementation. It was also observed that SLIM overestimated the highest and the deepest point of the

bathymetry. This could be avoided by implementing a slope limiter.

SLIM should now be able to represent morphological changes occurring at a much slower rate than the hydrodynamics. In order to simulate processes like a dam breach, however, the coupled approach should be introduced. The last part of the thesis described how it could be implemented in SLIM.

By the view of the encouraging results and the actual problematic, it would be interesting to implement this feature in SLIM. Very few models currently solve the coupled shallow water - Exner model with the discontinuous Galerkin method. However, investigation has found that it has a lot of potential. Further research could imply a study of different schemes to compute the flux at the interface between two elements.

Bibliography

[1] Audusse, E., Chalons, C., & Ung, P. (2016). A simple three-wave Approximate Riemann Solver for the Saint-Venant–Exner equations.

[2] Bernard, M. (2009). Etude de l'influence des courants secondaires sur l'écoulement dans un rétrécissement brusque sur fond mobile (Master's thesis, université Catholique de Louvain, Louvain-la-Neuve, Belgium).

[3] Blaise, S., De Brye, B., De Brauwere, A., Deleersnijder, E., Delhez, E. J., & Comblen, R. (2010). Capturing the residence time boundary layer-application to the Scheldt Estuary. Ocean Dynamics, 60(3), 535-554.

[4] de Brye, B. (2011). Multiscale Finite-Element Modelling of River-Sea Continua (Doctoral dissertation, Ph. D. thesis. Institut de Mécanique, Matériaux et génie Civil, Université Catholique de Louvain).

[5] Brooks, A. N., & Hughes, T. J. (1982). Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. Computer methods in applied mechanics and engineering, 32(1-3), 199-259.

[6] Dı, M. C., Fernández-Nieto, E. D., Ferreiro, A. M., & Parés, C. (2009). Two-dimensional sediment transport models in shallow water equations. A second order finite volume approach on unstructured meshes. Computer Methods in Applied Mechanics and Engineering, 198(33), 2520-2538.

[7] Einstein, H. A., & Krone, R. B. (1962). Experiments to determine modes of cohesive sediment transport in salt water. Journal of Geophysical Research, 67(4), 1451-1461.

[8] Fraselle, Q. (2010). Solid transport in flooding rivers with deposition on the floodplain: experimental and numerical investigations (Doctoral dissertation, Ph. D thesis, université Catholique de Louvain, Louvain-la-Neuve, Belgium).

[9] Geuzaine, C., & Remacle, J. F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. International Journal for Numerical Methods in Engineering, 79(11), 1309-1331.

[10] Gourgue, O., Baeyens, W., Chen, M. S., de Brauwere, A., de Brye, B., Deleersnijder, E., ... & Legat, V. (2013). A depth-averaged two-dimensional sediment transport model for environmental studies in the Scheldt Estuary and tidal river network. Journal of Marine Systems, 128, 27-39.

[11] Gourgue, O. (2011). Finite element modeling of sediment dynamics in the Scheldt (Doctoral dissertation, Université catholique de Louvain).

[12] Grass, A. J. (1981). Sediment transport by waves and currents. University College, London, Dept. of Civil Engineering.

[13] Hanert, E., Legat, V., & Deleersnijder, É. (2003). A comparison of three finite elements to solve the linear shallow water equations. Ocean Modelling, 5(1), 17-35.

[14] Harten, A., Lax, P. D., & Leer, B. V. (1983). On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. SIAM review, 25(1), 35-61.

[15] Hibma, A., De Vriend, H. J., & Stive, M. J. F. (2003). Numerical modelling of shoal pattern formation in well-mixed elongated estuaries. Estuarine, Coastal and Shelf Science, 57(5), 981-991.

[16] Hudson, J., & Sweby, P. K. (2005). A high-resolution scheme for the equations governing 2D bed-load sediment transport. International Journal for Numerical Methods in Fluids, 47(10-11), 1085-1091.

[17] Hudson, J. (2001). Numerical techniques for morphodynamic modelling (Doctoral dissertation, University of Reading).

[18] Kale, V. S., Ely, L. L., Enzel, Y., & Baker, V. R. (1994). Geomorphic and hydrologic aspects of monsoon floods on the Narmada and Tapi Rivers in central India. Geomorphology, 10(1-4), 157-168.

[19] Kubatko, E. J., Westerink, J. J., & Dawson, C. (2006). An unstructured grid morphodynamic model with a discontinuous Galerkin method for bed evolution. Ocean modelling, 15(1), 71-89.

[20] Lambrechts, J. (2011). Finite element methods for coastal flows: application to the Great Barrier Reef (Doctoral dissertation, PhD thesis, Université catholique de Louvain).

[21] Legat, V. (2004). Introduction aux éléments finis. Université catholique de Louvain.

[22] Meyer-Peter, E., & Müller, R. (1948). Formulas for bed-load transport. In IAHSR 2nd meeting, Stockholm, appendix 2. IAHR.

[23] Okubo, A. (1971, August). Oceanic diffusion diagrams. In Deep sea research and oceanographic abstracts (Vol. 18, No. 8, pp. 789-802). Elsevier.

[24] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1, p. 85). USA: Trelgol Publishing.

[25] Partheniades, E. (1965). Erosion and deposition of cohesive soils. Journal of the Hydraulics Division, 91(1), 105-139.

[26] Pham Van, C. (2014). Development of a finite element model simulating flow and sediment dynamics: application to the Mahakam land-sea continuum (Indonesia) (Doctoral dissertation, UCL).

[27] Pham Van, C., Deleersnijder, E., Bousmar, D., & Soares-Frazao, S. (2014). Simulation of flow in compound open-channel using a discontinuous Galerkin finite-element method with Smagorinsky turbulence closure. Journal of Hydro-environment Research, 8(4), 396-409.

[28] Schuttelaars, H. M., & De Swart, H. E. (1999). Initial formation of channels and shoals in a short tidal embayment. Journal of fluid mechanics, 386, 15-42.

[29] Shiono, K., & Knight, D. W. (1991). Turbulent open-channel flows with variable depth across the channel. Journal of Fluid Mechanics, 222, 617-646.

[30] Smagorinsky, J. (1963). General circulation experiments with the primitive equations: I. The basic experiment. Monthly weather review, 91(3), 99-164.

[31] Soares-Frazão, S., & Zech, Y. (2011). HLLC scheme with novel wave-speed estimators appropriate for two-dimensional shallow-water flow on erodible bed. International Journal for Numerical Methods in Fluids, 66(8), 1019-1036.

[32] Soares-Frazão, S., Canelas, R., Cao, Z., Cea, L., Chaudhry, H. M., Die Moran, A., ... & Greco, M. (2012). Dam-break flows over mobile beds: experiments and benchmark tests for numerical models. Journal of Hydraulic Research, 50(4), 364-375.

[33] de Vriend, H. N. (1987). 2DH mathematical modelling of morphological evolutions in shallow water. Coastal Engineering, 11(1), 1-27.

[34] de Vriend, H. J., & Ribberink, J. S. (1996). Mathematical modeling of meso-tidal barrier island coasts part II: process-based simulation models. In Advances in coastal and ocean engineering (pp. 151-197).

[35] Wu, W. (2004). Depth-averaged two-dimensional numerical modeling of unsteady flow and nonuniform sediment transport in open channels. Journal of hydraulic engineering, 130(10), 1013-1024.

[36] Wang, Z. B., Louters, T., & de Vriend, H. J. (1995). Morphodynamic modelling for a tidal inlet in the Wadden Sea. Marine geology, 126(1-4), 289-300.

APPENDIX A

Generating a mesh with GMSH

The .geo file used to generate the mesh of the compound channel using GMSH is presented hereunder as an example on how to use SLIM.

```
start = 0;
end = 10;

//Create the points to draw the contour
Point(1) = {start, 0.605, 0, 1.0};
Point(2) = {start, -0.605, 0, 1.0};
Point(3) = {start, 0.2, 0, 1.0};
Point(4) = {start, 0.1492, 0, 1.0};
Point(5) = {start, -0.1492, 0, 1.0};
Point(6) = {start, -0.2, 0, 1.0};
Point(7) = {end, 0.605, 0, 1.0};
Point(8) = {end, -0.605, 0, 1.0};
Point(9) = {end, 0.2, 0, 1.0};
Point(10) = {end, -0.1492, 0, 1.0};
Point(11) = {end, 0.1492, 0, 1.0};
Point(12) = {end, -0.2, 0, 1.0};

//Create the lines between the points
Line(1) = {7, 1};
Line(2) = {2, 8};
Line(3) = {1, 3};
Line(4) = {3, 4};
Line(5) = {4, 5};
Line(6) = {5, 6};
Line(7) = {6, 2};
Line(8) = {9, 7};
```

```
27 Line(9) = {11, 9};
28 Line(10) = {10, 11};
29 Line(11) = {12, 10};
30 Line(12) = {8, 12};
31 Line(13) = {3, 9};
32 Line(14) = {4, 11};
33 Line(15) = {5, 10};
34 Line(16) = {6, 12};
35
36 //Defines the boundary of every surface. The elements will be generated
       separately on each surface
37 Line Loop(17) = {13, 8, 1, 3};
38 Plane Surface(18) = {17};
39 Line Loop(19) = {14, 9, -13, 4};
40 Plane Surface(20) = {19};
41 Line Loop(21) = {15, 10, -14, 5};
42 Plane Surface(22) = {21};
43 Line Loop(23) = {16, 11, -15, 6};
44 Plane Surface(24) = {23};
45 Line Loop(25) = {2, 12, -16, 7};
46 Plane Surface(26) = {25};
47
48 //Defines the physical tags that will be used to impose the boundary conditions
49 Physical Line("Wall") = {1, 2};
50 Physical Line("In") = {3, 4, 5, 6, 7};
51 Physical Line("Out") = {8, 9, 10, 11, 12};
52 Physical Surface("Domain") = {18, 20, 22, 24, 26};
53
54 //Other parameters
55 Mesh.CharacteristicLengthFactor = 0.1;
56 Mesh.Algorithm = 6;
```

APPENDIX B

Running a simulation with SLIM

The python script used for the compound channel is presented hereunder as an example on how to use SLIM.

```
1  import slimPre
2  import mySlim as slim
3  from dgpy.scripts import slim_private
4  import numpy
5
6  ###################
7  ##  Parameters  ##
8  ###################
9
10 #File parameters
11 meshfile = "CompoundChannel.msh"
12 output_bath = "output/bath"              #Output folder for the bathymetry
13 output_nc = "netcdf/"                    #Output folder for NETcdf files
14 mesh_start = 0
15 mesh_end = 10
16
17 #Time parameters
18 dt = 1                            #Time step
19 tstart = 0                        #Starting time for the hydrodynamics
20 tend = 160                        #End time of the simulation
21 tinit = 80                        #Starting time for the tracer equation
22 tstab = 50                        #Time at which the boundary conditions reach
       their final value
23 export = dt
24
25 #Bathymetry parameters
```

```
26 b = 0.605
27 bfp = 0.405
28 h = 0.0752
29 hfp = 0.0508
30 s0 = 0.0019
31 level = 0.0119
32
33 #Parameters boundary conditions
34 q = 0.015
35 slip = 0.996
36
37 #Flow parameters
38 smagorinsky_coefficient = 0.4
39 manning_mc = 0.010
40 manning_w = 0.001
41 manning_fp = 0.016
42
43 #Parameters tracer
44 concentrationIn = 0.0
45 concentration = 0.0
46 source = 1.0
47
48 #Parameters sediment
49 initial_bottom_concentration = 0.0
50 windU = 1e-4
51 windV = 1e-4
52 rhosediment = 2650
53 porosity = 0.4
54 rhosedimentbottom = rhosediment*(1.-porosity)
55 d50 = 91e-6
56 d90 = 0.0004
57 shields = 0.0119
58 diffusivity = 1e-6
59 g = 9.81
60 fluxMc = 1
61
62 diffusivity_mode_constant = False
63 diffusivity_mc = 1.2e-3
64 diffusivity_w = 1.0e-3
65 diffusivity_fp = 0.4e-3
66 smagorinsky_coefficient = 0.3
67 h = 0.063
68 manning_mc = 0.010
69 manning_w = 0.01
70 manning_fp = 0.012
71 q = 0.015
72 level = 0.005
```

```
73
74 #The functions to calculate various parameters are defined. They will be used
       later
75 def bathf(x,y) :
76     bathi = 0.
77     if abs(y) >= (b-bfp) :
78         bathi = h-hfp
79     elif abs (y) >= (b-bfp-hfp) :
80         bathi = h-(abs(y)-(b-bfp-hfp))
81     else :
82         bathi = h
83     return bathi+x*s0-level
84
85 def calcEta(x,y) :
86     return -x*s0+level
87
88 def calcUInit(x,y) :
89     return 0.
90
91 def calcManning(x,y):
92     n = manning_mc
93     if abs(y) >= (b-bfp) :
94         n = manning_fp
95     elif abs (y) >= (b-bfp-hfp) :
96         n = manning_w
97     else :
98         n = manning_mc
99     return n
100
101 def calcDiffusivity(x,y):
102     if abs(y) >= (b-bfp) :
103         d = diffusivity_fp
104     elif abs (y) >= (b-bfp-hfp) :
105         d = diffusivity_mc +
                (abs(y)-(b-bfp-hfp))/(hfp)*(diffusivity_fp-diffusivity_mc)
106     else :
107         d = diffusivity_mc
108     return d
109
110
111 def calcC(x,y) :
112     c=concentration
113     return c
114
115 def calcSource(x,y):
116     c = 0
117     if abs(y-0)<=(b-bfp-hfp)+0.0001 and abs(x)<0.1:
```

```
118             c=0.8*source/(0.1*(b-bfp-hfp)*2*h*1000)*700*q
119         return c
120
121     def calcSed(x,y) :
122         sed = initial_bottom_concentration
123         return sed
124
125     def calcQ(t0, i,dt):
126         t = t0 + i*dt
127         discharge = q
128         if t<tstab:
129             discharge = q/tstab*t
130         return discharge
131
132     def calcEtaIn(t0,i,dt):
133         return calcEta(mesh_start,0)
134
135     def calcEtaOut(t0,i,dt):
136         return calcEta(mesh_end,0)
137
138
139
140     #Parameters for the sediment
141     factorW= 0.01
142     wMax=0.0001
143     u0erosion = 0.4
144     u0deposition = u0erosion
145     w0=10
146     a1=0.000001
147     n=4
148     a01=0.028
149     a02=0.144
150     omega2=2.4538
151     eros0Fact=0.245
152
153
154     if not slim_private.path.exists(output_nc):
155         slim_private.makedirs(output_nc)
156
157     ####################
158     ### pre process ###
159     ####################
160
161
162     #Create the object of type mesh
163     mesh = slimPre.Mesh(meshfile)
164     #Create the slimPre region
```

```
region = slimPre.Region(mesh)
#array of shape (n,3), n being the number of nodes in the region, with the coordinates of the mesh nodes
xyz = region.coordinates

#The x and y coordinates are stored in a NETcdf file
slimPre.write_file(output_nc + 'coordinates.nc', region=region, time=None, data=[('x',xyz[:,0]),('y',xyz[:,1])])

########################
### Flow parameters ###
########################

#Initiates the vectors containing the flow parameters on the whole domain
bath = numpy.zeros(xyz.shape[0])
manning_vector = numpy.zeros(xyz.shape[0])
diffusivity_vector = numpy.zeros(xyz.shape[0])

#Fills the flow parameters as defined in the functions previously
for i in range(xyz.shape[0]) :
    bath[i] = bathf(xyz[i,0], xyz[i,1])
    manning_vector[i] = calcManning(xyz[i,0], xyz[i,1])
    diffusivity_vector[i] = calcDiffusivity(xyz[i,0], xyz[i,1])

#The flow parameters are saved in NETcdf files, the standard input for SLIM simulations
slimPre.write_file(output_nc + 'bath.nc', region=region, time=None, data=[('bath',bath)])
slimPre.write_file(output_nc + 'manning.nc', region=region, time=None, data=[('manning',manning_vector)])
slimPre.write_file(output_nc + 'sediment.nc', region=region, time=None, data=[('diffusivity',diffusivity_vector)])

#Export a field in the netcdf format to msh format and gives the same name as the original mesh file but in the folder "output/bath"
slimPre.netcdf_to_msh(meshfile, output_nc + "bath.nc", "bath", output_bath)

######################
### open boundary ###
######################

#The object of type time which stores each time step is created
time_vector = numpy.arange(0,tend+2*dt,dt,numpy.float64)
time = slimPre.Time(time_vector)

#The vectors that will hold the boundary conditions are set to zero
#They have the same size as the time. There is one value for each time step
```

```
205 etaOut = numpy.zeros_like(time_vector)
206 discharge_vector = numpy.zeros_like(time_vector)
207
208 #Fills the boundary conditions as defined previously
209 for i in range(time_vector.shape[0]) :
210     etaOut[i] = calcEtaOut(tstart,i,dt)
211     discharge_vector[i] = calcQ(tstart,i,dt)
212
213 concentrationIn_vector=concentrationIn*numpy.ones_like(time_vector)
214 concentrationOut_vector=concentration*numpy.ones_like(time_vector)
215
216 #The boundary conditions are saved in two NETcdf files. One for the
       hydrodynamics, one for the sediments
217 #The NETcdf files are applied to the time (time=time)
218 slimPre.write_file(output_nc + 'boundaryIn.nc', region=None, time=time,
       data=[('dischargeIn', discharge_vector)])
219 slimPre.write_file(output_nc + 'boundaryOut.nc', region=None, time=time,
       data=[('etaOut', etaOut)])
220 slimPre.write_file(output_nc + 'sedimentBoundaryIn.nc', region=None,
       time=time, data=[('c', concentrationIn_vector)])
221 slimPre.write_file(output_nc + 'sedimentBoundaryOut.nc', region=None,
       time=time, data=[('c', concentrationOut_vector)])
222
223 ##########################
224 ### initial condition ###
225 ##########################
226
227 #Initiates the vectors for the initial conditions to zero
228 etaInit = numpy.zeros(xyz.shape[0])
229 uInit = numpy.zeros(xyz.shape[0])
230 vInit = numpy.zeros(xyz.shape[0])
231 cInit = numpy.zeros(xyz.shape[0])
232 bottomConcentration_vector = numpy.zeros(xyz.shape[0])
233 source_vector = numpy.zeros(xyz.shape[0])
234
235 #Fills the vector the way the initial conditions were previously defined
236 for i in range(xyz.shape[0]) :
237     etaInit[i] = calcEta(xyz[i,0], xyz[i,1])
238     uInit[i] = calcUInit(xyz[i,0], xyz[i,1])
239     cInit[i] = calcC(xyz[i,0], xyz[i,1])
240     bottomConcentration_vector[i] = calcSed(xyz[i,0], xyz[i,1])
241     source_vector[i] = calcSource(xyz[i,0], xyz[i,1])
242
243 windU_vector = windU*numpy.ones(xyz.shape[0])
244 windV_vector = windV*numpy.ones(xyz.shape[0])
245
```

```
246 #The initial conditions are saved in two NETcdf files. One for the
        hydrodynamics, one for the sediments
247 #The NETcdf file is now applied at the domain (region=region)
248 slimPre.write_file(output_nc + 'init.nc', region=region, time=None,
        data=[('eta', etaInit),('u', uInit),('v', vInit)])
249 slimPre.write_file(output_nc + 'sedimentInit.nc', region=region, time=None,
        data=[('c', cInit),('initial_bottom_concentration',
        bottomConcentration_vector),('source', source_vector),('windU',
        windU_vector),('windV', windV_vector)])
250
251 ############
252 ### run ###
253 ############
254
255 #The object of time domain is created by giving it the NETcdf file containing
        the bathymetry
256 domain = slim.Domain(meshfile,(output_nc + 'bath.nc','bath'))
257
258 #The system of the shallow water equations is created and the optional terms
        are added
259 eq = slim.ShallowWater2d(domain, "implicit",initial_time=tstart,
        final_time=tend, export_every_sub_time_step = False)
260 eq.set_viscosity("smagorinsky",smagorinsky_coefficient)
261 eq.set_dissipation("manning", coefficient=(output_nc + 'manning.nc','manning'))
262
263 #The initial and boundary conditions are applied
264 eq.set_initial_condition((output_nc + 'init.nc','eta'),(output_nc +
        'init.nc','u'),(output_nc + 'init.nc','v'))
265 eq.set_boundary_open('In', discharge=(output_nc +
        'boundaryIn.nc','dischargeIn'))
266 eq.set_boundary_open('Out', sse=(output_nc + 'boundaryOut.nc','etaOut'))
267 eq.set_boundary_coast('Wall', slip_factor=slip)
268
269 #The system of the tracer equation is created and the optional terms are added
270 eq2 = slim.ShallowWaterTracer2d(domain, "implicit",eq, name="tracer", offline
        = False, initial_time=tinit, final_time=tend)
271 eq2.set_diffusivity("variable", variable_diffusivity=(output_nc +
        'sediment.nc','diffusivity'))
272 eq2.set_source((output_nc + 'sedimentInit.nc','source'))
273 eq2.set_sediment((output_nc +
        'sedimentInit.nc','initial_bottom_concentration'),(output_nc +
        'sedimentInit.nc','windU'),(output_nc + 'sedimentInit.nc','windV'),
        rhosediment, porosity, d50, d90, shields, False, dt, factorW, wMax,
        u0erosion, u0deposition, w0, a01, a02, a1, n, omega2, eros0Fact)
274
275 #The initial and boundary conditions are applied
276 eq2.set_initial_condition((output_nc + 'sedimentInit.nc','c'))
```

```
277 eq2.set_boundary_open('In', (output_nc + 'sedimentBoundaryIn.nc','c'))
278 eq2.set_boundary_open('Out', (output_nc + 'sedimentBoundaryOut.nc','c'))
279 eq2.set_boundary_coast('Wall')
280
281 #Create an object of type Loop = temporal Solver
282 loop = slim.Loop(maximum_time_step=dt, export_time=export)
283
284 #Add the equations to solve
285 loop.add_equation(eq)
286 loop.add_equation(eq2)
287
288 #The simulation starts
289 loop.run()
290
291 slimPre.exit(0)
```

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**