UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Improving the wetting-drying algorithm of a finite-element, shallow-water model

Dissertation presented by
Clément HANSEN

for obtaining the Master's degree in
Mathematical Engineering

Supervisor(s)
Eric DELEERSNIJDER, Sandra SOARES FRAZAO

Reader(s)
Jonathan LAMBRECHTS, Sigrun ORTLEB , Miltiadis PAPALEXANDRIS

Academic year 2017-2018

Acknowledgements

This thesis has seen the day thanks to the help of many people. First of all, I would like to thank Jonathan Lambrechts who was always available to answer my questions and help me with my code.
I am also grateful to my supervisors, Eric Deleersnijder who has given me important advice and Sandra Soares Frazao whose work gave me a lot of inspiration for my thesis.
I thank Sigrun Ortleb who answered my questions related to her article. Without her work, this thesis might have never existed.
I would like to express my gratitude toward Georges Médinger who kindly accepted to correct my text in spite of his little knowledge of the engineering field.

Contents

Chapter 1

Introduction

1.1 State of the art

The earth is covered of 71% of water including fluvial and estuarine regions that represent less than 1% of the hydrosphere. However, those regions are the main actors for their respective ecosystems and have a significant amount of biomass compared to the oceans. Thus, it is natural that we build software that allow us to make predictions about the behaviour of the fluid in the different locations of our planet.

For example, we can quote the SLIM model [1] (Second Generation Louvain-la-Neuve Ice-ocean Model) which is produced in UCL. This model focuses on the sea-land continuity and solves equations related to geophysical, groundwater and environmental problems. A lot of modules have been implemented to SLIM and they allow the representation of the transport of sediments and contaminants. SLIM has been applied successfully to a large variety of theoretical and realistic problems as for examples the Scheldt estuary ([7]), The Congo river ([11]) and also seas and lakes of Titan, a moon of Saturn ([16]).

The SLIM model uses an algorithm called the discontinuous Galerkin method. This method works pretty well for fluid simulations since it can fit quite easily any complex geometries oppositely to finite differences method. However as all the schemes with a fixed grid, this method has a lot of difficulties to manage the boundaries of the flow that changes throughout time. This is a considerable problem since a lot of applications are concerned by the tides, floods, storm surges, etc. For example, the Mont Saint-Michel bay located between Normandy and Brittany in France, is characterized by the effect of the tides as we can see in the photos below.

Figure 1.1: Mont Saint-Michel at low tide

Figure 1.2: Mont Saint-Michel at high tide

The lands depicted on the left picture, which are alternatively exposed to air or water, are called tidal flats. We see in the case of the Mont Saint-Michel bay that it is essential to predict the covering and the uncovering of these regions. The tidal flats will cause a lot of problems to the discontinuous galerkin method because of the following reasons :

- The algorithm can give invalid results that could contain negative water height.
- The fact that the water height is close to zero can perturb our algorithm

[1] `https://sites.uclouvain.be/slim/`

- The knowledge of resistance laws of a thin layer of fluid against a porous surface is not well known.

In order to solve those problems, we will use methods that are called *Wetting-drying* methods. A lot of those methods exist with their own advantages and disadvantages but all in all, those methods should meet the following specifications :

- The method must guarantee a positive water height.
- Mass must be conserved locally and globally.
- The method shouldn't produce excessive wiggles.
- For Adam Candy [6], the perfect wetting-drying method is the one that is flexible and that allows to make simulation at all ranges of physics and scales with inherited consistency and conservation.

How do we evaluate those methods ? Andrea Balzano proposed in his article ([1]) different tests that allow us to evaluate the quality of a wetting-drying method. Those tests correspond to tides for different type of bathymetries (a uniform slope, one with bottom convexity and one with a bassin) in a one dimensional channel and the Thacker test whose analytical solution is known ([15]). The Thacker test is a 2D problem with an ellipsoidal bathymetry and is depicted below 1.3. We could also consider the dam-break problem (Ritter test case) which also has an analytical solution, in order to check the precision of the method. This problem corresponds to a column of water that suddenly outflows as in 1.1.

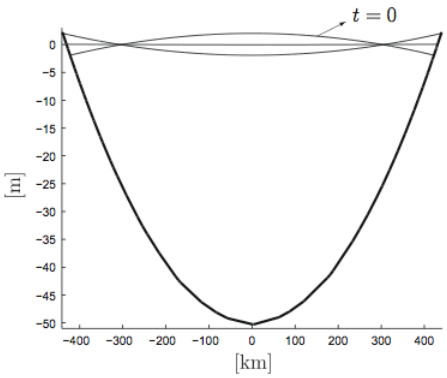


Figure 1.3: Thacker test case

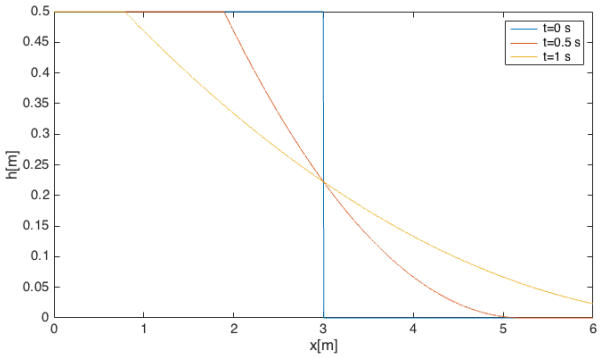


Figure 1.4: Ritter test case

Taking care of the tidal flats is not an easy task. This is why a large variety of solutions are proposed. Over the years, we've been able to make a classification of those wetting-drying methods. We identify two main categories.

1. Methods with an adaptive mesh. In this case, the frontier of the mesh becomes the wet/dry interface and the distinction between the dry and wet areas is clear. This kind of method is the one that is closer to real physics but its main drawback is that it is computationally demanding. For example, we can note the algorithm proposed by L.Zhao *et al* in [17] in which they proposed an algorithm that uses level sets. Those algorithms, that are called level sets methods, change the mesh around the separation between water and air. Another algorithm that uses an adaptive mesh for one dimensional problems is proposed by Onno Bokhove in [3].

2. Methods with a fixed grid. In this case, the mesh remains the same and we can divide this category into three subcategories :

- wet/dry cells : In this case the nodes and elements can be considered wet or dry. Under different criteria, those can be considered dry and in this case are removed from the computation. The drawback of those methods is that the abrupt removal or addition of a node or an element can bring numerical instabilities or perturb the mass and momentum conservation. One of the oldest method was proposed by Leendertse in 1970 ([12]) . We can also note the one proposed by Stelling in [14] . Both methods exclude or include cells or boundaries based on their respective definition of the water mean for a cell or a boundary.

  In order to reduce those perturbations, some methods introduce some transitive states with special treatments. In this way, the removal of a cell from the computation becomes less abrupt. One method that has transitive states is proposed for example by Bates and Hervouet in [2].

- Thin layer : Those algorithms keep a small layer of fluid in the dry regions. By this way, the dry elements are always in the computational domain. These methods also ensure the continuity between the elements. The difficulty in this case, is to maintain this layer everywhere on the domain even though the gravity will tend to dry certain elements. In order to avoid this problem, we will cancel the gravity force in different cases. We could cite for example the studies done by O.Gourgue [9] or Bunya [4].

- Porosity method : Certain methods allow negative water depth solutions so that the fluid flows in a porous layer. The main drawbacks of these methods are the fact that there might be inconsistency with mass and momentum conservations and also there is an issue about the definition of this porous layer. Karna in [10] proposed for example an implicit method that allows the bathymetry to move.

The most spread methods are the thin layer ones since they are easier to implement than a level set method and are more precise than a porosity method. Besides, they seem more attractive than the methods in which we add states for the cells. Another attractive feature for a wetting-drying method is to work with implicit time integration. Indeed, this will allow to make simulations over a long period of time unlike explicit time integration. Some categories of wetting-drying methods (unfixed mesh and wet/dry cells) are less suited for an implicit integration since they deal with discrete variables or states.

Currently, there are not a lot of methods that have such requirements. For example, we will explain one of these methods in this thesis by S.Ortleb and A.Meister ([13]). This method seems to be a step in the right direction in order to solve implicitly this issue. This method doesn't particularly fit in one of the five categories. It is a method that uses a Patankar trick when negative heights appear. Such a trick is based on the well known production-destruction equations. For methods that use explicit time integration, we have already many solutions like the one proposed by O.Gourgue ([9]).

To summarize, due to the tides, the discontinuous Galerkin method meets with experiencing difficulties due to the movable frontier of the fluid. The solutions to this problem are the wetting-drying methods. Those methods should conserve the mass and give consistent results (no excessive wiggles and no negative heights) . A lot of wetting-drying methods, that work with explicit time integration, fulfill those requirements. However, It is not really the case in implicit time integration even though this is more practical if simulations over a large period of time were to be made. From all the the various works that exist, a classification stood out and each of the categories have their own advantages and defaults.

1.2 This thesis

The aim of the thesis is to assess a thin layer method which is based on the algorithm proposed by S.Ortleb and A.Meister ([13]) in one dimension. Usually, the wetting-drying methods use an implicit scheme of order 1. The goal of S.Ortleb and A.Meister is to be able to use implicit schemes with higher order to be able to have more precise results. We will apply this method with the one-dimensional test cases of Balzano ([1]) and evaluate the results we obtained. The most important aspect to check is the implicit time integration since it is a must-have for simulations with a large period of time. The other important aspect is the use of scheme of order 2 and to generalize the method for higher order.

The thesis will be divided into four chapters. The first one deals with the derivation of the Saint-Venant equations and their properties. We will get to know the equations, we solve, better and to introduce the compatibility equations that are used in several numerical scheme. The second one explains the discontinuous Galerkin algorithm. This chapter introduces the theory of finite elements methods when we don't impose continuity between elements. Finally, the last two chapters correspond to the core of the thesis : the wetting-drying algorithms. The first one is devoted to an explicit method with a thin layer of fluid. The second one will use an implicit time integration using the algorithm of S.Ortleb and A.Meister with a thin layer.

Chapter 2

The shallow-water model

Before deriving the discontinuous Galerkin method, we will recall how to obtain the Shallow-water model and its fundamental properties. This will allow us to have a better grasp of the terms we integrate. Besides, in order to write the discontinuous Galerkin method, we will need to introduce some of the properties of those equations.

The Shallow-water equations, also called Saint-Venant equations, are a set of hyperbolic partial differential equations that describes the depth-averaged terms of a thin layer of fluid. In this model, we suppose that there is no vertical velocity term such that the vertical pressure profile is hydrostatic. Those equations are mainly used for numerical simulations in coastal regions, estuaries, rivers and channels and could describe for example, the progression of a tsunami.

First of all, we will derive the equations for a one-dimensional problem from the principles of conservation of mass and momentum. Afterwards, from this 1D model, we will define the characteristics and write the compatibility equations. This chapter is mainly based on the work done by Sandra Soares Frazao in the first chapter of her Phd thesis.

2.1 One-dimensional model

In the following pages, we will consider channels with arbitrary cross-section for the 1D problems in order to be more general. To write the Saint-Venant, we need to make the following assumptions :

- We only consider incompressible flow.
- The flow is parallel to the bed slope. This assumption allows us to define the average velocity and implies that the pressure distribution is hydrostatic.
- The flow is considered turbulent. It implies that the friction losses are proportional to the squared fluid velocity. This is the case for almost all the applications of the Saint-Venant equations as in the theory of river hydraulics, for example.

Before deriving the continuum equation, we will first introduce the notations that will be used throughout the chapter. Consider the figure in 2.1 that describes our channel, a channel is defined by :

- $P[m]$: the wetted perimeter.
- $z[m]$: the bed elevation.

The flow is characterized by the following :

- $\rho[kg/m^3]$: the density of the fluid.
- $A[m^2]$: area contained at a channel section.
- $Q[m^3/s]$: the flow.
- $\mathbf{V}[m/s]$: the flow velocity.

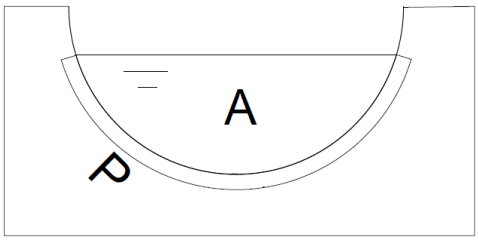


Figure 2.1: Section of the channel

2.1.1 Continuum equation

The continuum equation depicts the fact the mass, m, is conserved in our system. This can be written as :

$$\frac{Dm}{Dt} = 0$$

where $\frac{D\cdot}{D\cdot}$ is the material derivative. We can rewrite the material derivative over a control-volume, Ω with its corresponding control-surface, $\partial\Omega$:

$$\frac{DN}{Dt} = \frac{\partial}{\partial t}\int\int\int_{\Omega} \rho\eta \,\mathrm{d}\Omega + \int\int_{\partial\Omega} \eta\rho(\mathbf{V}\cdot\mathbf{n})\,\mathrm{d}s \tag{2.1}$$

The expression in 2.1 states that the time rate of any quantity $N = m\eta$ is equal to the time rate of the quantity N within the control volume Ω plus the rate of flux of N through the control surface $\partial\Omega$ where n is a unit vector that points outside the surface.

Let's consider a control volume of length dx as depicted in 2.2. Since we are in a one dimensional problem, we only consider the horizontal velocity, u. In order to write the continuum equation, we use the formula in 2.1 with $\eta = 1$:

$$\begin{aligned}\frac{Dm}{Dt} &= \frac{\partial}{\partial t}\int\int\int_{\Omega} \rho \,\mathrm{d}\Omega + \int\int_{A_{out}} \rho u \,\mathrm{d}s - \int\int_{A_{in}} \rho u \,\mathrm{d}s \\ &= \frac{\partial}{\partial t}(\rho A \,\mathrm{d}x) + (\rho Q + \frac{\partial}{\partial x}(\rho Q)\,\mathrm{d}x) - (\rho Q)\end{aligned}$$

with $Q = AV$. Since $\frac{Dm}{Dt} = 0$, we finally obtain the continuity equation :

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \tag{2.2}$$

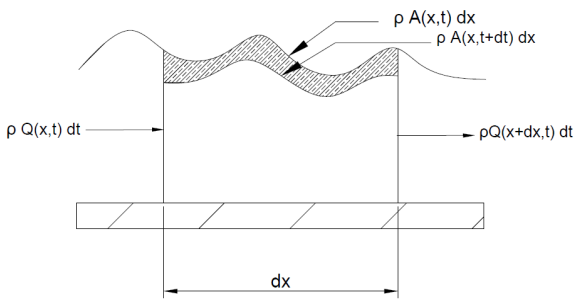


Figure 2.2: Mass conservation for a control volume

2.1.2 Momentum equations

The momentum equations are derived from the second law of Newton. Since we work on a one dimensional problem, we only consider the horizontal momentum such as :

$$\frac{D(mu)}{Dt} = F_x \tag{2.3}$$

where F_x is the sum of horizontal forces that are acting on the fluid. Those forces correspond to :

- F_g : the gravity force

  We take the horizontal component of the gravity force that corresponds to

  $$F_g = g\rho A \,\mathrm{d}x\, S_0$$

  with $S_0 = -\frac{\partial z}{\partial x}$ the bed slope

- F_f : the friction force

  This force is expressed as $F_f = -\tau P dx$ where τ is the shear stress along the bed. In order to proceed, we need a model for the shear stress.

  Since we supposed that the flow is turbulent, we have that $\tau = \mathcal{O}(\mathbf{V}^2)$. Besides, we can imagine that the bigger the wetted perimeter P is, the bigger the friction force will be and the bigger the area of the section A is, the less important the friction force will be. Let's define $R = A/P$, the hydraulic radius. From the statement above, we can guess that $\tau = \mathcal{O}(R^{-1})$.

  All in all, we can express the shear stress through the friction slope S_f that meets all the above requirements. We express friction using the Manning formula :

  $$S_f = \frac{n^2 V^2}{R^{4/3}} \tag{2.4}$$

  with n the Manning coefficient. We can thus now express the shear stress τ and the friction force F_f :

  $$\begin{aligned} \tau &= g\rho R S_f \\ F_f &= -g\rho A S_f \end{aligned} \tag{2.5}$$

- F_h : hydrostatic force at the sections of the control volume

  The hydrostatic force at section is equal to $\rho g I_1$ where I_1 is the cross-section momentum. We defined I_1 as follows with terms that are defined in the figure 2.3 :

  $$I_1 = \int_0^h b(\eta)(h-\eta)d\eta \tag{2.6}$$

  Thus, the hydrostatic force corresponds to the difference between the force we obtain at the sections of the control volume :

  $$F_h = \rho g I_1 - (\rho g I_1 + \frac{\partial}{\partial x}(\rho g I_1)\,\mathrm{d}x) = -\frac{\partial}{\partial x}(\rho g I_1)\,\mathrm{d}x$$

  Let's rewrite $\frac{\partial I_1}{\partial x}$ in terms of the other variables by using the Leibniz theorem.

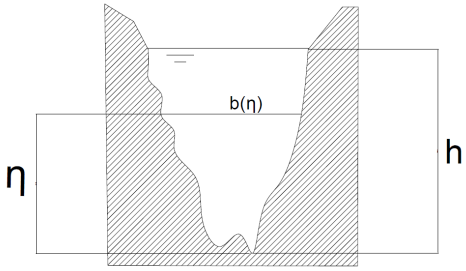


Figure 2.3: Cross-section of our channel

$$\begin{aligned}\frac{\partial I_1}{\partial x} &= \int_0^h \left(\frac{\partial b}{\partial x}(h-\eta) + b\frac{\partial h}{\partial x}\right) \mathrm{d}\eta + \frac{\partial h}{\partial x} b(h)(h-h) - 0 \\ &= A\frac{\partial h}{\partial x} + \int_0^h \frac{\partial b}{\partial x}(h-\eta)\,\mathrm{d}\eta\end{aligned}$$

We define I_2 the effect of a change in the section width.

$$I_2 = \int_0^h \frac{\partial b}{\partial x}(h-\eta)d\eta$$

All in all, we can express the hydrostatic force exerted on the control volume :

$$F_h = -\rho g A \frac{\partial h}{\partial x}\,\mathrm{d}x - \rho g I_2\,\mathrm{d}x \tag{2.7}$$

- $F_{h,l}$: the longitudinal component of the hydrostatic force

  This occurs when the width section changes provoking a hydrostatic force on the walls. This force depends thus directly on I_2 so that :

$$F_{h,l} = g\rho I_2\,\mathrm{d}x \tag{2.8}$$

All in all, the sum of the horizontal forces are equal to :

$$F_x = \rho g A\,\mathrm{d}x\,(-\frac{\partial h}{\partial x} + S_0 - S_f) \tag{2.9}$$

Now that we have written the right-hand side of the equation in (2.3), let's compute the material derivative $\frac{D(mu)}{Dt}$ using the relation in (2.1).

$$\begin{aligned}\frac{D(mu)}{Dt} &= \frac{\partial}{\partial t}\int\int\int_\Omega \rho u\,\mathrm{d}\Omega + \int\int_{A_{out}} \rho u^2\,\mathrm{d}s - \int\int_{A_{in}} \rho u^2\,\mathrm{d}s \\ &= \frac{\partial}{\partial t}\left[\rho\,\mathrm{d}x\int\int_A u\,\mathrm{d}s\right] + \frac{\partial}{\partial x}\left[\rho\int\int_A u^2\,\mathrm{d}s\right]\mathrm{d}x\end{aligned}$$

The first term in the sum is equal to $\rho\,\mathrm{d}x\,\frac{\partial Q}{\partial t}$ since $\mathrm{d}Q = u\,\mathrm{d}s$. For the second one, we will express the horizontal speed in term of the mean velocity, V and a variable e so that $u = V(1+e)$. The variable e has the property that $\int e\,\mathrm{d}A = 0$ since $\int u\,\mathrm{d}A = VA = Q$. It follows that :

$$\begin{aligned}\int\int u^2 \,\mathrm{d}A &= \int\int V^2(1+e)^2 \,\mathrm{d}s \\ &= V^2 \int\int (1+e^2) \,\mathrm{d}s \\ &= \beta A V^2\end{aligned}$$

where $\beta = 1 + \frac{\int\int e^2 \mathrm{d}A}{A}$ is the Boussinesq approximation. All in all, we can write the material derivative of the momentum :

$$\frac{D(mu)}{Dt} = \rho dx \left(\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}(\beta A V^2) \right)$$

Finally, we can write the momentum equation based on (2.3) and the assumption that $\beta \approx 1$:

$$\rho dx \left(\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}(AV^2) \right) = \rho g A \,\mathrm{d}x \,(-\frac{\partial h}{\partial x} + S_0 - S_f)$$

In order to have a conservative form, we can rewrite $A\frac{\partial h}{\partial x}$ as :

$$A\frac{\partial h}{\partial x} = \frac{\partial I_1}{\partial x} - I_2$$

Which gives us the final equation :

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A} + gI_1 \right) = gA(S_0 - S_f) + gI_2 \tag{2.10}$$

An important hypothesis of the shallow water equations hasn't been covered yet. Indeed, we talked about the conservation of the horizontal momentum but what about the vertical momentum. Since the water height is small compared to the length of the domain, a lot of terms can be neglected. This is what Saint-Venant did, claiming that the pressure is hydrostatic. The vertical velocity is thus 0 and the conservation of the vertical momentum is reduced to :

$$\frac{\partial p}{\partial y} = 0$$

with p the pressure and y the vertical component.

2.1.3 Summary

From the conservation of mass and the second law of Newton, we were able to derive the shallow water model which consists of two hyperbolic partial differential equations. One particularity of that type of equations is that the solution can contain shocks and discontinuities for any initial conditions. This is one of the main reason we will consider discontinuous Galerkin as we will see in the next chapter. Those non-linear relations can be written in this form for clarity :

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{s} \tag{2.11}$$

Where :

$$\mathbf{U} = \begin{pmatrix} A \\ Q \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} Q \\ \frac{Q^2}{A} + gI_1 \end{pmatrix} \qquad \mathbf{s} = \begin{pmatrix} 0 \\ gA(S_0 - S_f) + gI_2 \end{pmatrix}$$

The aim of the next chapter is to write a numerical method that solves the equation above

which is not so easy as we will see. To make our job simpler, we will only consider the case where we have a rectangular channel with a constant width, B. This implies the following :

$$I_1 = \frac{Bh^2}{2} \qquad I_2 = 0$$

We can rewrite our equations based on the following variables $h = A/B$ and $q = Q/B$. The relations will have the same form as (3.1) with :

$$\mathbf{U} = \begin{pmatrix} h \\ q \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} q \\ \frac{q^2}{h} + g\frac{h^2}{2} \end{pmatrix} \qquad \mathbf{s} = \begin{pmatrix} 0 \\ gh(S_0 - S_f) \end{pmatrix} \tag{2.12}$$

2.2 Properties of the shallow-water equations

Now that we obtained the shallow-water equations, we can raise some of their properties that will be handy for later. We will show that an information travels along trajectories called *characteristics*. We will derive the *compatibility equations* that are used in several numerical schemes.

Fist of all, let's apply the method of characteristics for the shallow-water equations. In other words, we compute the linear combination of the mass (2.2) and momentum conservation (2.10) in order to get an ordinary differential equation.

$$\begin{aligned} L_1 &\equiv \frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \\ L_2 &\equiv \frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} - gA(S_0 - S_f) = 0 \end{aligned}$$

We rewrite the second equation by noting that $\mathrm{d}A = B\,\mathrm{d}h$ which corresponds to the cross-section area variation :

$$L_2 \equiv \frac{\partial Q}{\partial t} + 2\frac{Q}{A}\frac{\partial Q}{\partial x} - \frac{Q^2}{A^2}\frac{A}{x} + g\frac{A}{B}\frac{\partial A}{\partial x} - gA(S_0 - S_f) = 0$$

The method of the characteristics consists in finding the parameters μ soi that the following equation yields a system of ODE.

$$\mu L_1 + L_2 = 0$$

Reorganizing the terms, we obtain this expression :

$$\left[\frac{\partial Q}{\partial t} + \frac{\partial Q}{\partial x}\left(2\frac{Q}{A} + \mu\right)\right] + \mu\left[\frac{\partial A}{\partial t} + \frac{\partial A}{\partial x}\left(-\frac{Q^2}{\mu A^2} + \frac{g}{\mu}\frac{A}{B}\right)\right] = gA(S_0 - S_f)$$

The above equation becomes an ODE if and only if $\frac{\mathrm{d}x}{\mathrm{d}t}$ is equal to the terms in parenthesis. Thus, we need to check the following equality.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = 2\frac{Q}{A} + \mu = -\frac{Q^2}{\mu A^2} + \frac{g}{\mu}\frac{A}{B}$$

Solving for μ, we obtain $\mu = -V \pm c$ with $V = Q/A$ and $c = \sqrt{gA/B}$ which is called the

relative celerity of an infinitesimal perturbation.The set of ODE becomes :

$$\begin{aligned} \frac{\mathrm{d}Q}{\mathrm{d}t} + (-V-c)\frac{\mathrm{d}A}{\mathrm{d}t} &= gA(S_0 - S_f) \text{ on } C^- \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V - c \\ \frac{\mathrm{d}Q}{\mathrm{d}t} + (-V+c)\frac{\mathrm{d}A}{\mathrm{d}t} &= gA(S_0 - S_f) \text{ on } C^+ \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V + c \end{aligned} \tag{2.13}$$

Those equations are called the compatibility equations. C^- and C^+ are the characteristics and express the condition under which the Saint-Venant equations become ODE. They can be drawn in a x-t plane and represent the way the information is propagating. Based on the Froude number $Fr = \frac{V}{c}$, we can distinguish two different types of flow :

- subcritical (Fr <1) : the information travels in the upstream and the downstream direction.
- supercritical (Fr >1): the information only travels in one direction.

The equations in (2.13) can be expressed in terms of Q and $\Sigma = \frac{Q^2}{A} + gI_1$. By doing this, we will obtain the dual form of the compatibility equations which is used for several numerical schemes as well. The dual form is thus :

$$\begin{aligned} \frac{\mathrm{d}\Sigma}{\mathrm{d}t} + (-V-c)\frac{\mathrm{d}Q}{\mathrm{d}t} &= (V-c)\left[gA(S_0 - S_f) + gI_2\right] \text{ on } C^- \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V - c \\ \frac{\mathrm{d}\Sigma}{\mathrm{d}t} + (-V+c)\frac{\mathrm{d}Q}{\mathrm{d}t} &= (V+c)\left[gA(S_0 - S_f) + gI_2\right] \text{ on } C^+ \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V + c \end{aligned} \tag{2.14}$$

For future developments in the next chapter, we will consider cases with rectangular prismatic channels. In this case, we will express the compatibility equations and their dual based on the variables h and q. The compatibility equations ((2.13)) become :

$$\begin{aligned} \frac{\mathrm{d}q}{\mathrm{d}t} + (-V-c)\frac{\mathrm{d}h}{\mathrm{d}t} &= gh(S_0 - S_f) \text{ on } C^- \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V - c \\ \frac{\mathrm{d}q}{\mathrm{d}t} + (-V+c)\frac{\mathrm{d}h}{\mathrm{d}t} &= gh(S_0 - S_f) \text{ on } C^+ \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V + c \end{aligned} \tag{2.15}$$

In this case, we have $\Sigma = \frac{q}{h} + \frac{1}{2}gh^2$ and the dual ((2.14)) becomes :

$$\begin{aligned} \frac{\mathrm{d}\Sigma}{\mathrm{d}t} + (-V-c)\frac{\mathrm{d}q}{\mathrm{d}t} &= gh(V-c)(S_0 - S_f) \text{ on } C^- \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V - c \\ \frac{\mathrm{d}\Sigma}{\mathrm{d}t} + (-V+c)\frac{\mathrm{d}q}{\mathrm{d}t} &= gh(V+c)(S_0 - S_f) \text{ on } C^+ \equiv \frac{\mathrm{d}x}{\mathrm{d}t} = V + c \end{aligned} \tag{2.16}$$

Chapter 3

The discontinuous Galerkin method in one dimension

In this chapter, we will see how to integrate the equations we introduce in the previous chapter using the discontinuous Galerkin method with an unstructured mesh. The main difference with the continuous Galerkin method is that we don't impose continuity between elements. The use of this method, which combines the property of finite elements and finite volume methods, is supported by the following.

- An unstructured mesh allows us to fit complex domains as the Scheldt estuary quite easily unlike a structured mesh. For 1D problems, this advantage doesn't hold because our domain is just a line that can be filled easily by elements with the same length.
- It allows to capture solutions with shocks and discontinuous unlike the continuous Galerkin method. This is an important property since the solution of the Saint-Venant's equations are characterized by those. Those shocks are related to hydraulic jumps that can occur when the fluid velocity drops abruptly. For example, it happens when a pipe has a sudden shrinking.
- The method is easier to code for parallel computing than the continuous Galerkin method since the elements are only coupled at their frontier.

The chapter will be structured in the following manner. First, we will derive the weak and the discrete formulation of the Shallow-water equations. Afterwards, we will discuss numerical fluxes that will allow us to link the elements to each other and also about the slope limiter which allows to get a smoother solution. Finally, we will talk about time integration.

3.1 Formulations

Without loss of generality, we consider the one-dimensional problem with a rectangular prismatic channel. Let's recall the equation of Shallow-water in one dimension from the previous chapter :

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{s} \tag{3.1}$$

so that :

$$\mathbf{U} = \begin{pmatrix} h \\ q \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} q \\ \frac{q^2}{h} + g\frac{h^2}{2} \end{pmatrix} \qquad \mathbf{s} = \begin{pmatrix} 0 \\ gh(S_0 - S_f) \end{pmatrix} \tag{3.2}$$

The equations in (3.1) is also called the *strong formulation*. Therefore, our problem is to find the evolution of the nodes located at the extremities of our elements. We will denote the unknowns related to the i^{th} element as $\mathbf{U}_i^- = \begin{pmatrix} h_i^- & q_i^- \end{pmatrix}^T$ and $\mathbf{U}_i^+ = \begin{pmatrix} h_i^+ & q_i^+ \end{pmatrix}^T$ as depicted in 3.1. We will denote the convective terms F_i^- and F_i^+ exactly in the same way.

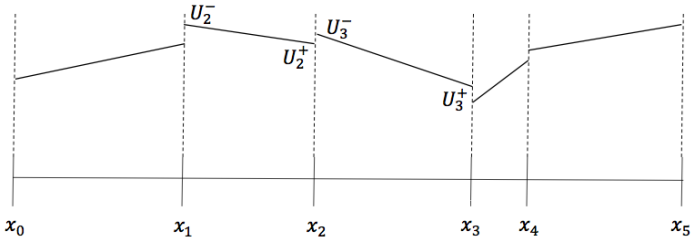


Figure 3.1: Discontinuous Galerkin representation

3.1.1 Interpolation

Before writing the weak formulation, we need to discuss about how we can write an expression related to the i^{th} element along its section, $\mathbf{U}_i(x,t)$, based on the value $\mathbf{U}_i^-(t)$ and $\mathbf{U}_i^+(t)$. First of all, let's consider an element called $\bar{\Omega}$ whose extremities are located at $\xi = -1$ and $\xi = 1$ as depicted in 3.3.

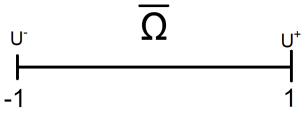


Figure 3.2: Parent element

The interpolation for the element above, $\mathbf{U}(t)$, is obtained as follow :

$$\mathbf{U^h}(\xi,t) = \phi_1(\xi)\mathbf{U}^-(t) + \phi_2(\xi)\mathbf{U}^+(t)$$

The functions $\phi_1(\xi)$ and $\phi_2(\xi)$ are called the *basis functions*. The constraints on those functions are that $\phi_1(-1) = \phi_2(1) = 1$ and $\phi_1(1) = \phi_2(-1) = 0$ which give us the following expressions for $\xi \in [-1,1]$:

$$\phi_1(\xi) = \frac{1-\xi}{2} \qquad \phi_2(\xi) = \frac{1+\xi}{2} \tag{3.3}$$

For $\xi \notin [-1,1]$, we have $\phi_1(\xi) = \phi_2(\xi) = 0$. Now, we wish to do the same interpolation for any element. However, we won't rewrite the basis function for each of them. Indeed, we will use the result we had above for $\bar{\Omega}$ and establish an isomorphism between this element and for any element.

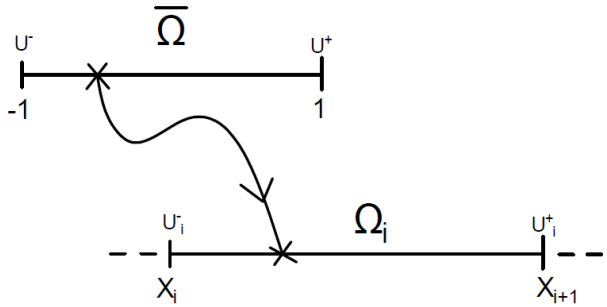


Figure 3.3: isomorphism

Let's have an element Ω_i whose extremities are located at $x = X_i$ and $x = X_{i+1}$. We define the isomorphism in the following way :

$$x(\xi) = \frac{1-\xi}{2} X_i + \frac{1+\xi}{2} X_{i+1} \tag{3.4}$$

$$\xi(x) = \frac{2x - X_i - X_{i+1}}{X_{i+1} - X_i} \tag{3.5}$$

With the above isomorphism, we can obtain the interpolation $\mathbf{U}_i^h(x,t)$ using the basis function we computed above so that :

$$\mathbf{U}_i^h(x,t) = \phi_1(\xi(x))\mathbf{U}_i^-(t) + \phi_2(\xi(x))\mathbf{U}_i^+(t) \tag{3.6}$$

3.1.2 Weak formulation

We wish to somehow express the equations in (3.1) in terms of our unknowns $\mathbf{U}_i^-$ and $\mathbf{U}_i^+$. This can be done by means of the *weak formulation*.

First of all, let's introduce the following notation :

$$\langle f \rangle = \int_\Omega f \, \mathrm{d}\Omega$$

$$\langle\langle f \rangle\rangle = \int_{\partial\Omega} f \, \mathrm{d}s$$

Let's define the space of admissible solutions $\mathcal{U}$ so that a solution of the strong formulation (3.1), $\mathbf{U}$, must verify $\mathbf{U} \in \mathcal{U}$. We deduce the weak formulation by integrating the strong formulation multiplied by a perturbation of the solution, $\hat{\mathbf{U}}$. A perturbation is defined such that $\mathbf{U} + \hat{\mathbf{U}} \in \mathcal{U}$ and that it must verify $\hat{\mathbf{U}} \in \hat{\mathcal{U}}$ where $\hat{\mathcal{U}}$ is the space of variation. With the help of some algebra, we obtain the weak formulation in the following manner:

$$\left\langle \left(\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} - \mathbf{s} \right) \hat{\mathbf{U}} \right\rangle = 0$$

$$\downarrow \text{Integration by part}$$

$$\left\langle \frac{\partial \mathbf{U}}{\partial t} \hat{\mathbf{U}} \right\rangle - \left\langle F \frac{\partial \hat{\mathbf{U}}}{\partial x} \right\rangle + \left\langle \frac{\partial}{\partial x}(F\hat{\mathbf{U}}) \right\rangle - \left\langle s\hat{\mathbf{U}} \right\rangle = 0$$

$$\downarrow \text{Divergence theorem}$$

$$\left\langle \frac{\partial \mathbf{U}}{\partial t} \hat{\mathbf{U}} \right\rangle - \left\langle F \frac{\partial \hat{\mathbf{U}}}{\partial x} \right\rangle + \left\langle\left\langle F\hat{\mathbf{U}} \right\rangle\right\rangle - \left\langle s\hat{\mathbf{U}} \right\rangle = 0$$

The weak formulation can be written as :

Find $\mathbf{U} \in \mathcal{U}$ given that :

$$\left\langle \frac{\partial \mathbf{U}}{\partial t} \hat{\mathbf{U}} \right\rangle - \left\langle F \frac{\partial \hat{\mathbf{U}}}{\partial x} \right\rangle + \left\langle\left\langle F\hat{\mathbf{U}} \right\rangle\right\rangle - \left\langle s\hat{\mathbf{U}} \right\rangle = 0 \qquad \forall \hat{\mathbf{U}} \in \hat{\mathcal{U}} \tag{3.7}$$

3.1.3 Discrete formulation

Now, we are going to see how from the weak formulation we will get a discrete formula of the following form :

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{U}_1^- \\ \mathbf{U}_1^+ \\ \mathbf{U}_2^- \\ \vdots \\ \mathbf{U}_n^- \\ \mathbf{U}_n^+\end{pmatrix} = \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathcal{L}(\mathbf{U}) \tag{3.8}$$

with n being the number of elements in our discretization. Note that the vector $\mathbf{U} = \left(\mathbf{U}_1^-, \mathbf{U}_1^+, \mathbf{U}_2^-, \cdots, \mathbf{U}_n^-, \mathbf{U}_n^+\right)$ is not the same one as in the weak formulation (3.7) which corresponds to a continuous solution through the entire domain. This leads us to the following question : how do we introduce the discrete values in our problem ? This happens by solving the weak formulation for the interpolation of $\mathbf{U}(x,t)$, $\mathbf{U}^h(x,t)$:

$$\boxed{\begin{array}{l}\text{Find } \mathbf{U}^h \in \mathcal{U} \text{ given that :} \\ \left\langle \frac{\partial \mathbf{U}^h}{\partial t}\hat{\mathbf{U}}^h \right\rangle - \left\langle F\frac{\partial \hat{\mathbf{U}}^h}{\partial x} \right\rangle + \left\langle \left\langle F\hat{\mathbf{U}}^h \right\rangle \right\rangle - \left\langle s\hat{\mathbf{U}}^h \right\rangle = 0 \qquad \forall \hat{\mathbf{U}}^h \in \hat{\mathcal{U}}\end{array}} \tag{3.9}$$

Since we don't impose continuity between the elements, we can divide our problem into pieces. Thus, we will only write the expression related to the i^{th} element $\mathbf{U_i^h}(\mathbf{x},\mathbf{t})$. The discontinuous Galerkin method is obtained by picking a perturbation equals to the basis functions, $\hat{\mathbf{U}}^h = \phi_j(1\ 1)^T$ with $j = 1, 2$. Recalling that $\mathbf{U}_i^h$ is computed as in (3.6), we can develop the expression (3.9) for our element :

$$\begin{aligned}\left\langle (\phi_1\phi_j)\frac{\mathrm{d}\mathbf{U}_i^-}{\mathrm{d}t} + (\phi_2\phi_j)\frac{\mathrm{d}\mathbf{U}_i^+}{\mathrm{d}t} \right\rangle &= \left\langle \mathbf{F}_i\frac{\partial\phi_j}{\partial x} + \mathbf{s}\phi_j \right\rangle - \langle\langle \mathbf{F}_i\phi_j \rangle\rangle \\ A_i\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} &= \left\langle \mathbf{F}_i\frac{\partial\phi_j}{\partial x} + \mathbf{s}\phi_j \right\rangle - \langle\langle \mathbf{F}_i\phi_j \rangle\rangle\end{aligned}$$

Where A_i is called the rigidity matrix related to the element i. Afterwards, we can compute the terms that depend on the basis functions. We can start by the derivatives $\frac{\partial\phi_j}{\partial x}$ by using the chain rule and the expression of the isomorphism in (3.5):

$$\begin{aligned}\frac{\partial\phi_j}{\partial x} &= \frac{\partial\phi_j}{\partial\xi}\frac{\partial\xi}{\partial x} \\ &\Big\downarrow \quad \frac{\partial\phi_j}{\partial\xi} = \frac{(-1)^j}{2} \qquad \frac{\partial\xi}{\partial x} = \frac{2}{X_{i+1} - X_i} \\ \frac{\partial\phi_j}{\partial x} &= \frac{(-1)^j}{l_i}\end{aligned} \tag{3.10}$$

where l_i is the length of the i^{th} element. Finally we can compute the entries of the rigidity matrix A_i by integrating $\phi_i\phi_j$. Since $\mathbf{U}_i$ contains four unknowns (h_i^-, q_i^-, h_i^+ and q_i^+), A_i is a matrix of size 4×4 where the first and third rows are related to the water height and the other ones to the flow. We start by the computation of the integrals of $\phi_i\phi_j$. This can be done quite easily if we again consider our isomorphism :

$$\begin{aligned}\langle \phi_j \phi_k \rangle &= \int_{X_i}^{X_{i+1}} \phi_j \phi_k \, \mathrm{d}x \qquad \text{with } j,k = 1,2 \\ &= \int_{-1}^{1} \phi_j \phi_k \frac{\partial x}{\partial \xi} \, \mathrm{d}\xi \\ &= \frac{l_i}{2} \left[\int_{-1}^{1} \phi_j \phi_k \, \mathrm{d}\xi \right]\end{aligned}$$

After some computations, we get that :

$$[\ \langle \phi_j \phi_k \rangle\ ]_{j,k} = \frac{l_i}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

We obtain the rigidity matrix by reorganizing the above matrix for the water height and flow equations.

$$A_i = \frac{l_i}{6} \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix} \tag{3.11}$$

Finally, the last terms to write are the remaining volume and surface integrals. We compute the volume integrals by using an integration rule. We can use for example the Gauss-Legendre scheme with 2 points written as :

$$\int_{-1}^{1} f(x) \, \mathrm{d}x \approx \frac{1}{2} \left(f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \right)$$

Since we are in a one-dimensional problem, the surface integrals correspond to the difference between the function evaluated at the element's boundaries.

$$\langle\langle \mathbf{F} \phi_j \rangle\rangle = \mathbf{F}^*(x_{i+1}) \phi_j(\xi(X_{i+1})) - \mathbf{F}^*(x_i) \phi_j(\xi(X_i))$$

The term $\mathbf{F}^*$ corresponds to the value of the convective term at the boundary. These terms will depend on the neighbouring elements and are a crucial key in order to link all the pieces of the puzzle together. In order to compute them, we will use what is called numerical fluxes. Those methods will be explained in the next subsection.

All in all, the equations that the discontinuous Galerkin solves for every element are :

$$\begin{aligned}\frac{l_i}{6} \left(2 \frac{\mathrm{d}\mathbf{U_i^-}}{\mathrm{d}t} + \frac{\mathrm{d}\mathbf{U_i^+}}{\mathrm{d}t} \right) &= \int_{X_i}^{X_{i+1}} \left(\phi_1 \mathbf{s} - \frac{\mathbf{F}_i}{l_i} \right) \mathrm{d}x + \mathbf{F_i^*}(X_i) \\ \frac{l_i}{6} \left(\frac{\mathrm{d}\mathbf{U_i^-}}{\mathrm{d}t} + 2 \frac{\mathrm{d}\mathbf{U_i^+}}{\mathrm{d}t} \right) &= \int_{X_i}^{X_{i+1}} \left(\phi_2 \mathbf{s} + \frac{\mathbf{F}_i}{l_i} \right) \mathrm{d}x - \mathbf{F_i^*}(X_{i+1})\end{aligned}$$

Our system is thus composed of the $4n$ above equations with n being the number of elements. In order to obtain a system as in (3.8), we introduce the rigidity matrix A which is obtained by combining the local one (3.11). All in all, we obtain the following system :

$$A \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = B(\mathbf{U})$$

where $B(\mathbf{U})$ corresponds to the right hand side of the above equations. We obtain the discrete

formulation (3.8) by solving this system.

3.2 Numerical flux

In the previous section, we encountered terms that were shared between several elements. Those were the surface terms, $\mathbf{F}^*$. As we said earlier, it is an important aspect of the discontinuous Galerkin method since they are the only link between the elements. In order to compute those terms, we use what is called a *numerical flux*. A multitude of them exists with their own advantages and defaults. We can quote for example some of the classical ones which are the Roe, HLL or the Lax-Friedrich flux. The two latter ones are the ones we will develop and use thereafter.

We can note that for a discontinuous Galerkin method, the choice of the numerical flux isn't decisive compared to the finite volume method. This is due to the fact that an iteration of a finite volume algorithm is exclusively obtained with the values at the edges of the cells. Whereas, for an iteration of the DG algorithm, we also need to integrate the terms along the elements.

3.2.1 Lax-Friedrich

The first idea that would come to mind in order to compute $\mathbf{F}^*$ would be to simply take the average of $\mathbf{F}$ at the intersection :

$$\mathbf{F}^* = \frac{\mathbf{F}^L + \mathbf{F}^R}{2}$$

where the superscripts L and R correspond to the values on the left and right hand side. However, this idea leads to instable results. In order to be stable, we need to consider the fact the information travels from upstream or downstream. That's why we will model the propagation of information using the compatibility equations and their dual we introduced in the previous chapter. By neglecting the source terms and taking the mean of those equation, we get :

$$\mathbf{F}^* = \frac{\mathbf{F}^L + \mathbf{F}^R}{2} + \lambda_{max}\frac{\mathbf{U}^L - \mathbf{U}^R}{2} \tag{3.12}$$

λ_{max} is the maximum between $|V^L| + c^L$ and $|V^R| + c^R$. Now, our numerical flux becomes stable and is not too difficult to use. However, one disadvantage of this flux is that it tends to dissipate the solution.

3.2.2 HLL

Another classical numerical flux is the HLL flux. This flux is directly based again on the compatibility equations and their duals. Indeed, suppose that the flow is subcritical (the information comes from both direction), F^* is obtained by solving the following system :

$$C^- : \mathbf{F}^* - \mathbf{F}^R = (V + c)(\mathbf{U}^* - \mathbf{U}^R)$$
$$C^+ : \mathbf{F}^* - \mathbf{F}^L = (V - c)(\mathbf{U}^* - \mathbf{U}^L)$$

Let $\lambda^+ = V + c$ and $\lambda^- = V - c$. The solution of this system is given by :

$$\mathbf{F}^* = \frac{\lambda^+}{\lambda^+ - \lambda^-}\mathbf{F}^L - \frac{\lambda^-}{\lambda^+ - \lambda^-}\mathbf{F}^R + \frac{\lambda^-\lambda^+}{\lambda^+ - \lambda^-}(\mathbf{U}^R - \mathbf{U}^L) \tag{3.13}$$

Now, we wonder how to choose λ^+ and λ^- since there is two velocities and celerities that we could use. Those values correspond to the greatest λ^+ and the smallest λ^- between the left or right side of the intersection.

Suppose now that the flow is supercritical. In this case, the information only comes from one direction. For example, if the information comes from downstream, we will obtain $\mathbf{F}^* = \mathbf{F}^L$ and if it comes from upstream, it will be $\mathbf{F}^* = \mathbf{F}^R$. Considering both cases, λ^+ and λ^- are given by :

$$\lambda^+ = \max(0, V^L + c^L, V^R + c^R)$$
$$\lambda^- = \min(0, V^L - c^L, V^R - c^R)$$

If one of those values is 0, it would mean that the flow is supercritical and if not, in a subcritical case.

3.2.3 Boundary conditions

We also need to consider the case where an element shares an extremity with a boundary. There are several ways to impose the boundary conditions but we will focus on the one based on the compatibility equations and their dual form. As for the HLL and Lax-Friedrich fluxes, we neglect the source terms.

The first thing we need to consider is whether we impose the boundary condition upstream or downstream. If it is upstream, we consider the C^- characteristic otherwise we consider the C^+ characteristic. We will consider that that the terms $\mathbf{F}$ and $\mathbf{U}$ are the ones from the cell adjacent to the boundary. We consider λ to be equal to $V - c$ if it a downstream condition otherwise it is equal to $V + c$. Here is a list of boundary conditions we can impose.

- Transmissive :
  A boundary condition is transmissive when it doesn't affect the flow. In other words, it is assumed that the information coming from upstream or downstream is constant. In this case, the surface term $\mathbf{F}^*$ is directly computed as :

  $$\mathbf{F}^* = \mathbf{F}$$

- Wall :
  If there is a wall, the velocity must be equal to 0. In this case we will have :

  $$q^* = 0$$
  $$\Sigma^* = \Sigma - \lambda q$$

  with $\lambda = \pm c$ depending if it is an upstream or downstream condition.

- Imposed flow :
  Suppose that we want to impose a flow at a boundary, $q_b(t)$. In this case, we will apply it in the same manner as the wall condition :

  $$q^* = q_b(t)$$
  $$\Sigma^* = \Sigma - \lambda(q_b(t) - q)$$

  Here $\lambda = V \pm c$.

- Imposed height :

Now suppose that we don't want to impose a flow but a height, $h_b(t)$. Using the compatibility equations, we can express the dual in terms of Σ and h. Indeed, we apply our boundary condition in the following fashion :

$$q^* = q + \lambda(h_b(t) - h)$$
$$\Sigma^* = \Sigma + \lambda^2(h_b(t) - h)$$

3.3 Time integration

Once we have our discrete system in (3.8), all we need is to use a time integration scheme. In the literature, we can find a multitude of those. For example, we can quote the Euler, the Runge-Kutta or the Adams-Moulton schemes. However from all the material we have, two choices are offered to us. The first one is to use an explicit scheme and the other one an implicit scheme. Let's discuss the use of those types of method by comparing the explicit and the implicit Euler scheme.

- *Explicit.* The advantage of using an explicit scheme is that it's easy to use and not computationally expensive. Indeed, if we take one iteration of the Euler scheme, an iteration is computed in the following fashion :

  $$U^{n+1} = U^n + dt\mathcal{L}(U^n)$$

  where dt is the time step and n is a subscript related to time. We see that to find the next iterate we only need to evaluate the right-hand side at the actual iterate. Yet, those schemes suffer from their stability condition which enables them to make simulation over a large period of time. Indeed in the case of the Euler scheme, the time step needs to respect the following :

  $$dt \leq \frac{dx}{\lambda}$$

  dx is the length of the smallest element and λ corresponds to the biggest characteristic speed, $|V| + c$. the right term of the inequality is called the CFL number. Generally, we choose a time step smaller than the CFL number to assure stability. In this case, we clearly see that for precise mesh the explicit schemes suffer a lot which is frequent when we have a complex geometry.

- *Implicit.* Oppositely to the explicit method, the implicit method doesn't suffer from a stability condition since they are unconditionally stable. Those schemes allow to make simulation over a large period of time. However, they are computationally expensive. Indeed, let's compute an iteration of the implicit Euler algorithm :

  $$U^{n+1} = U^n + dt\mathcal{L}(U^{n+1})$$

  We notice that to find the next iterate, we have to solve a non-linear system. In comparison, the explicit method is easier to use. To solve this system, we can use the Newton-Raphson method that demands the computation of the Jacobian, $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}U}$. This makes the use of the implicit methods a lot harder than the explicit schemes.

If it is long simulation, an implicit scheme should be used and if it is short one, it would be better to use an explicit scheme. Regarding the precision of the results, the implicit algorithms

can be less precise if we choose a large time step. Thus in this case, the explicit method can be more accurate due to the small time increment.

3.4 Slope limiter

A slope limiter might be needed in addition to the numerical scheme we use. A slope limiter is used in order to reduce wiggles that comes from shocks, discontinuities or abrupt changes. Those wiggles become more important for high order schemes. As the numerical flux, we have a multitude of them. In this section we will introduce one simple slope limiter.

As its name suggests, we will take care of the element that has an abrupt solution . Let's take the i^{th} element and compare it to its neighbour. Let's consider a quantity ψ_i (in our case it's either the height, h or the flow, q) and let $\bar{\psi}_i$ the mean quantity over the i^{th} element. Let's consider the following case.

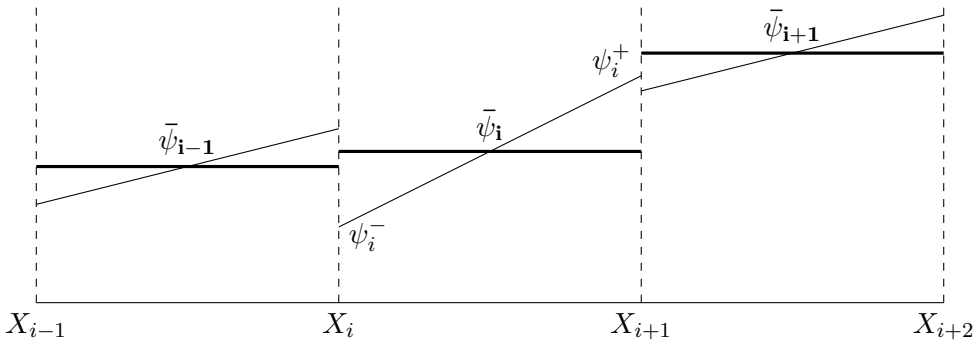


Figure 3.4: Application of our slope limiter : before

We see in the example above that the slope of the element in the middle has to be taken into account since it seems too abrupt in comparison to its neighbour. We will use our slope limiter on the i^{th} element if the values at his edges (ψ_i^- and ψ_i^+) aren't included between the mean values of the adjacent cells ($\bar{\psi}_{i-1}$ and $\bar{\psi}_{i+1}$). In our example, we observe that ψ_i^- is below $\bar{\psi}_{i-1}$ thus we need to change our solution. First of all, we set ψ_i^- to $\bar{\psi}_{i-1}$ so that the solution ψ_i is between the two neighbouring averages. At the end, we change ψ_i^+ so that we don't alter the mean, $\bar{\psi}_i$ and perturb the conservation of mass or momentum. Applying the slope limiter, our example becomes :

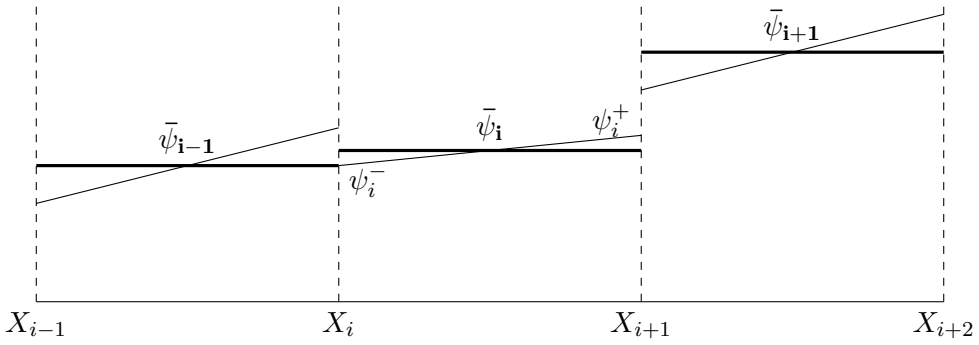


Figure 3.5: Application of our slope limiter : after

The only cases in which we don't apply a slope limiter is when the elements are next to a boundary or when we face a maximum (minimum) that is to say when $\bar{\psi}_i$ is above (below) $\bar{\psi}_{i-1}$ and $\bar{\psi}_{i+1}$. In addition, the slope limiter gives more stable solutions for the height, h if we apply it to the water elevation $\eta = h + z$ (z being the bed elevation) instead of h directly.

Chapter 4

The explicit Wetting-Drying

Now, we get to the heart of the matter. Indeed, all we did previously was to introduce the Saint-Venant equations and the discontinuous Galerkin method but we didn't mention how to take care of the moving boundary. This is a crucial issue if the tides are an important simulation factor. As we've seen in the state of the art, many solutions to this problem exist with their own advantages and defaults. They were 4 categories of methods and the one we are particularly interested in is a thin layer model. As we said earlier, a lot of researches were made for the explicit methods thus a lot of solutions exist

In this chapter, we will introduce an explicit method that keeps a thin layer of fluid everywhere. We will motivate the choice of this layer and explain how to maintain it. At the end of this chapter, we will show some results that shows the efficiency of the method.

4.1 The method

Why should we use a thin layer of fluid ? At first thought, it is contrary to what physics will predict and we can't distinguish the cells that are wet or dry. The answer is quite simple. It assures the continuity between the elements and we can avoid issues related to small water depths. In opposition to the other types of method, a thin layer algorithm is not computationally expensive and is easy to implement. Besides, a thin layer algorithm is more fitted for implicit time integration as we will see in the next chapter.

Thus, all the rest of the section is dedicated to the maintenance of this thin layer. Suppose that the height of our layer is ϵ, we wish to obtain a solution in which the height is above that value everywhere. A thin layer is depicted in the diagram below. z corresponds to the bathymetry. The rest of the section is dedicated to the numerical techniques that allow to keep this layer.

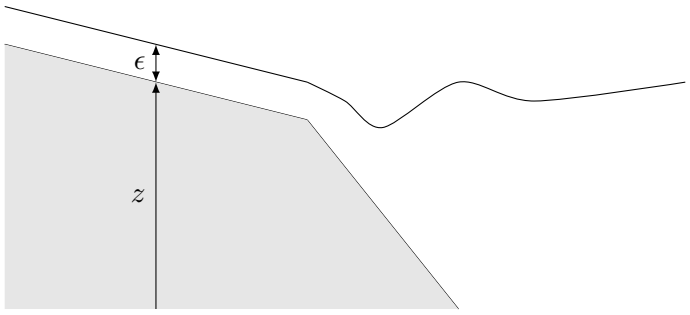


Figure 4.1: Representation of a thin layer

4.1.1 Cancellation of the gravity terms

Physically, we wouldn't be able to keep a small layer everywhere. Indeed, normally the gravity would drag the water to the bottom of the slope. Thus, we have to cancel the forces, that would drain the water, to some point to maintain our layer.

We will distinguish two different cases. For one case, we will cancel the gravity terms and for the other one we will keep those forces. Those cases are illustrated in 4.2 and 4.3. The first case is called flooding because the water will fill in the thin layer. The other type is called draining because the thin layer will dry out due to the gravity. The first case will not cause any problem to our algorithm. However, the second case is the problem we need to take care of.

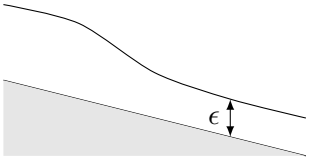
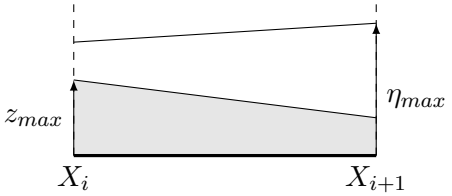


Figure 4.2: case : flooding

Figure 4.3: case : Draining

To differentiate both cases, we introduce a coefficient, f. This coefficient is included between 0 and 1. If it is equal to 0, we are in the case of a draining 4.3 so we cancel the appropriate forces. If it is equal to 1, there is no need to cancel those forces as in the flooding case 4.2. For each element, f is defined by the maximum water elevation $\eta = z + h$ and bathymetry in the cell. Indeed, f is given by the following formula.

$$f = \min\left(1, \max\left(0, \frac{\eta_{max} - z_{max} - \epsilon}{4\epsilon}\right)\right) \qquad (4.1)$$



The function (4.1) guarantees that f will be a value between 0 and 1 and that it is 0 if the element is only composed of the thin layer ($\eta_{max} - z_{max} = \epsilon$). The value at the denominator in (4.1) was found with some optimization. Now that we have the parameter f, we need to include it into the equations we solve. The forces that would drain our water are : the hydrostatic and gravity forces. Let's rewrite the integral related to the hydrostatic force that appear in the weak formulation by using the integration by part twice.

$$\begin{aligned}
\left\langle g\phi_j \frac{\partial}{\partial x}\left(\frac{h^2}{2}\right)\right\rangle &\approx \left\langle\left\langle g\left(\frac{h^2}{2}\right)^* \phi_j\right\rangle\right\rangle - \left\langle g\frac{h^2}{2}\frac{\partial \phi_j}{\partial x}\right\rangle \\
&\approx \left\langle\left\langle g\left(\frac{h^2}{2}\right)^* \phi_j - g\frac{h^2}{2}\phi_j\right\rangle\right\rangle - \left\langle g\frac{h^2}{2}\frac{\partial \phi_j}{\partial x}\right\rangle + \left\langle\left\langle g\frac{h^2}{2}\phi_j\right\rangle\right\rangle \\
&\approx \left\langle\left\langle g\left(\frac{h^2}{2}\right)^* \phi_j - g\frac{h^2}{2}\phi_j\right\rangle\right\rangle + \left\langle g\phi_j \frac{\partial}{\partial x}\left(\frac{h^2}{2}\right)\right\rangle
\end{aligned}$$

Now that we developed the term above, we can place our coefficient f in order to cancel this force. Using the above expression,our expression now becomes :

$$\left\langle g\phi_j \frac{\partial}{\partial x}\left(\frac{h^2}{2}\right)\right\rangle \rightarrow \left\langle\left\langle g\left(\frac{h^2}{2}\right)^* \phi_j - g\frac{h^2}{2}\phi_j\right\rangle\right\rangle + f\left\langle g\phi_j \frac{\partial}{\partial x}\left(\frac{h^2}{2}\right)\right\rangle$$

In addition, we need to multiply by f the source term ghS_0 related to the gravitational force. By doing this, we will be sure to maintain the thin layer. The last term from above is computed by using the fact that we use linear interpolation function ϕ_i. For example, if we consider the i^{th}

element, the expression of the height is given by :

$$h_i(x,t) = \phi_1(x)h_i^-(t) + \phi_2(x)h_i^+(t)$$

Thus, since h_i is a linear function, $\frac{\partial h}{\partial x}$ is simply its slope, $(h_i^+ - h_i^-)/2$. Then, the last term is given simply by :

$$\left\langle g\phi_j \frac{\partial}{\partial x}\left(\frac{h^2}{2}\right)\right\rangle = \left\langle g\phi_j h\frac{\partial h}{\partial x}\right\rangle$$

All in all, this technique allows to maintain the thin layer just by introducing a coefficient f into the equations. The cancellation of the gravity terms should be enough to work in explicit. However, it won't be the case in the implicit integration for high order schemes. This is why in the next part we will make some modification to the slope limiter, we discussed in the last chapter, in order to keep the thin layer of fluid.

4.1.2 Slope limiter

Last chapter, we discussed the slope limiters and explained how they are useful to reduce the wiggles in our solution. We can add in its list of advantages the fact that it can help us maintain our thin layer. Indeed, we will complete our method to take care of heights that are less than ϵ.

Without loss of generality, we consider the i^{th} element and suppose that h_i^+ is less than ϵ and h_i^- bigger than ϵ. In this case, the slope limiter will set h_i^- to ϵ and change h_i^- so that we keep the same water mean, $\bar{h}_i$ and conserve the mass. This is depicted as below.

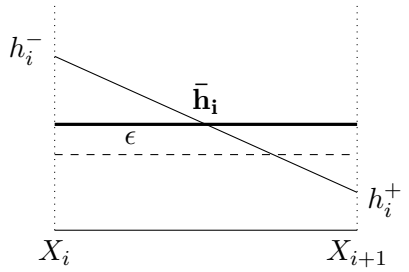


Figure 4.4: Before : Slope limiter

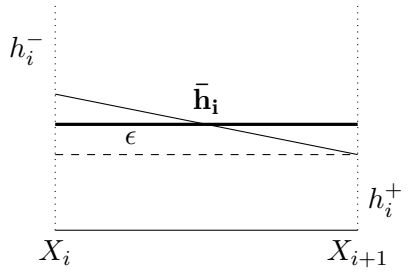


Figure 4.5: After : Slope limiter

Another case we need to take care of is when the water mean is less than ϵ. In this case, we reduce the solution to a polynomial of degree 0. In other words, we set the slope of the height to 0 and thus h_i^- and h_i^+ to $\bar{h}_i$.

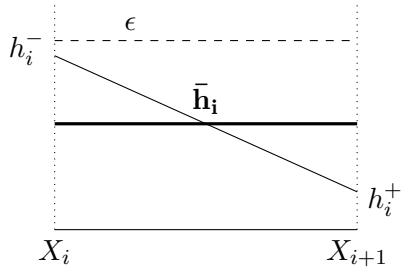


Figure 4.6: Before : Slope limiter

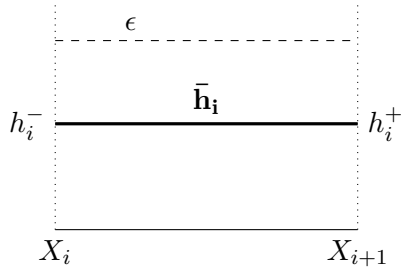


Figure 4.7: After : Slope limiter

Normally, there would be no need to add this to our slope limiter since in explicit time integration we would be sure to keep the thin layer. Indeed, as in explicit, we only choose time step that give stable results, the integration will maintain the thin layer. However, we use it by precaution even though we could ignore it. This modification is rather useful in implicit time integration because there is no guarantee to keep the thin layer.

4.2 Results

In this section, we will show the efficiency of our method by showing two test cases. The first one corresponds to water falling into a parabolic bathymetry and the second one corresponds to a column of water that spreads into a complex relief. The first example will show that the thin layer is maintained along the domain in the scenarios we discussed earlier in 4.2 and 4.3. The second one will show that the method works even for complex problems.

For the simulations, we chose a Runge-Kutta scheme of order 2. To obtain our results, we applied the slope limiter after each step of the integration scheme. The numerical flux we selected was the HLL flux. Regarding the boundary conditions, we chose to put walls at both ends. The time step was set to a fifth of the maximal CFL number at each iteration. The width of the channel was considered to be to 1 meter for both cases. The other parameters will be specified for each case.

4.2.1 Parabolic case

For the first case, the domain is set between -3 and 3 $[m]$. The bathymetry and the initial height are both equal to quadratic functions. The initial flow is set to 0 everywhere and the parameter ϵ to $0.2[m]$. Regarding the mesh, a structured grid is chosen and is composed of 100 elements. Thus, each cell lengths $0.06[m]$. We cancel friction forces by setting the manning coefficient to 0. With no friction force, the solution won't be damped by those and will tend to oscillate a lot.

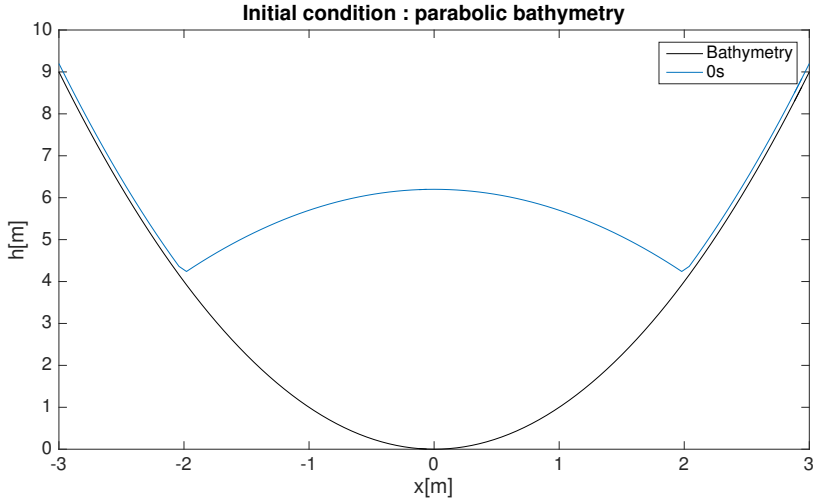


Figure 4.8: Initial condition

A simulation can be found in 4.9 and based on it we can interpret the behaviour of the fluid. At first, the column of water will collapse and the sides will be covered by the flow at the same moment as depicted by the blue curve. After a period of time, the fluid will stop collapsing and the water at the sides will fall back and refill the column as depicted by the red curve. Thus, the solution will oscillate and alter between the two scenarios we just mentioned . This is thus a perfect example to test if the height of the thin layer is maintained in the cases of a flood or a draining.

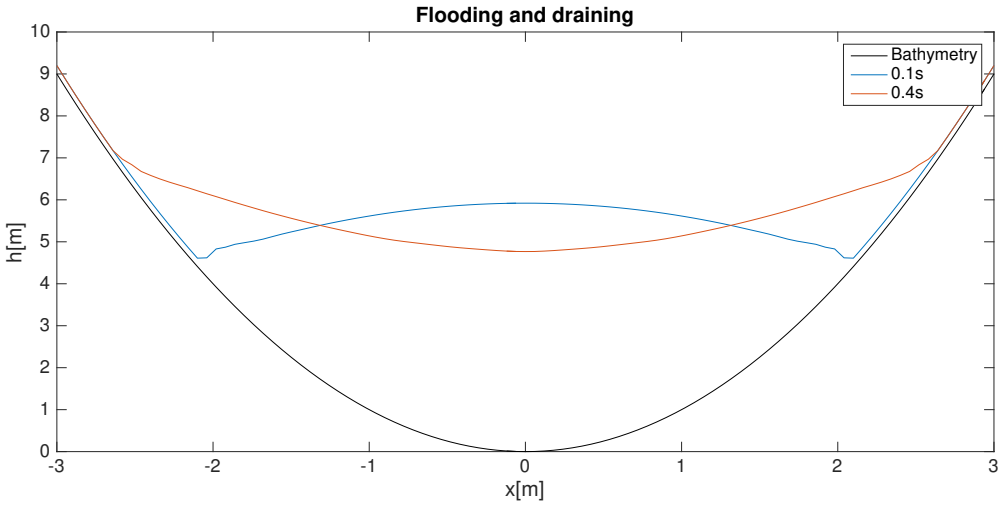


Figure 4.9: Solution : 2 cases

We showed that the method we explained earlier works for two different cases : one when the flow wets the sides and the other when it dries the sides. Now, we will zoom in our solution in order to observe whether or not the thin layer will remain.

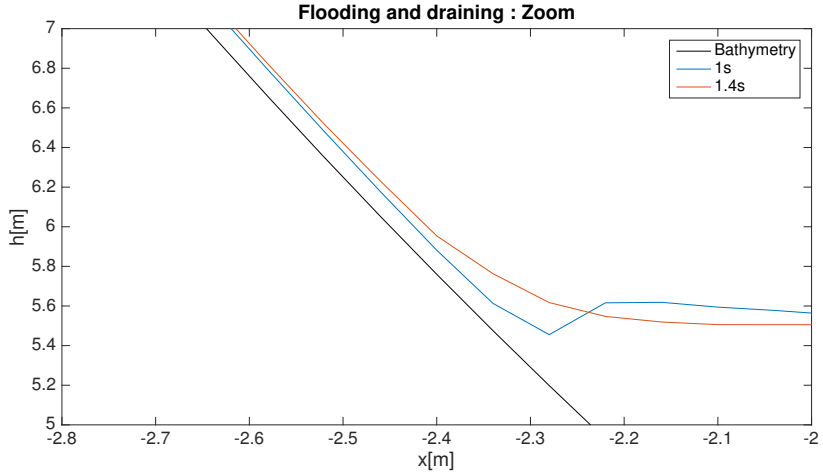


Figure 4.10: Solution : zoom

In 4.10, the blue curve corresponds to a flood and the red curve to a draining. As expected, the thin layer is unchanged in both cases. This is a good news because it means that our method works in both scenarios and especially the hard case where the cells are drying up.

4.2.2 Complex problem

Now that we have seen that our method was able to keep the thin layer of fluid, we can test it on a complex test case. It will allow us to check if the method works with a difficult relief. We have 200 elements of the same length across a boundary set between 0 and 20$[m]$. The manning coefficient is set to 0.015 so that the solution will be at rest quicker. The thin layer is set to 0.01$[m]$. The initial condition is a water column of 0.5$[m]$ high placed at the center of the domain. The width of the column is 5$[m]$. All in all, the initial condition is depicted in 4.11. This problem is a variant of the dam-break problem with a weird bathymetry. This problem is a difficult one because of the abrupt transition between the dry and the wet areas.

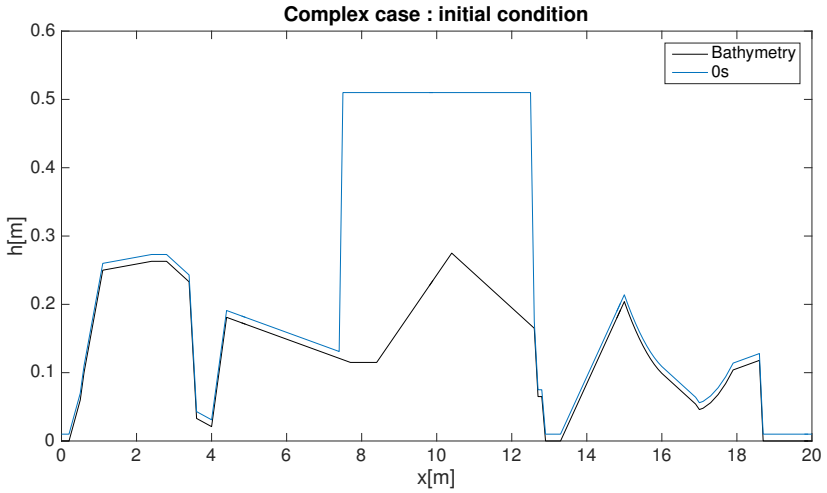


Figure 4.11: initial condition

Let's run our algorithm for our problem and observe the behaviour of the solution. As we see, there will be a lot of shocks and reflexions against the walls. We will see if there are no excessive wiggles or irregularities. First of all, we will observe what we obtain after 5 seconds in 4.12.

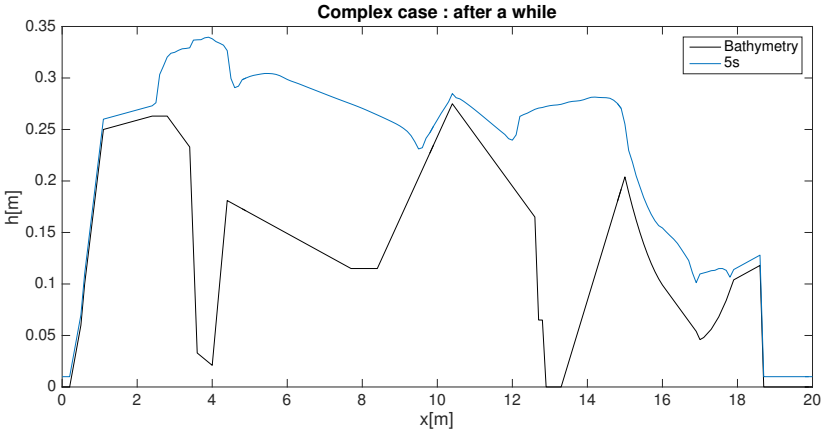


Figure 4.12: Solution after 5 seconds

We notice that the solution maintains the thin layer as expected. We observe that the solution is also smooth because it seems that there is no wiggle. The fluid behaves as we expected: the water column first fills in the gaps and the water is propelled due to the shock against the walls. Let's see what happened if we run our solution after a long time (100 seconds). Normally, the water should be at rest.

The solution appears in 4.13. We can see that at time the flow isn't moving. We clearly see, after this long period of time, that our thin layer is still present. Another aspect we didn't mention in our simulations is the conservation of mass. This is an important feature for the wetting drying methods because it should respect the physics laws. Since the result is a linear piecewise function, we can compute the water volume quite easily. Indeed, if we use the trapezoidal rule, we will get the exact volume without any error. In the initial condition, we had a volume $1.7744[m^3]$ and after 100 seconds, we still obtain the same volume. This proves that our method also conserves the mass.

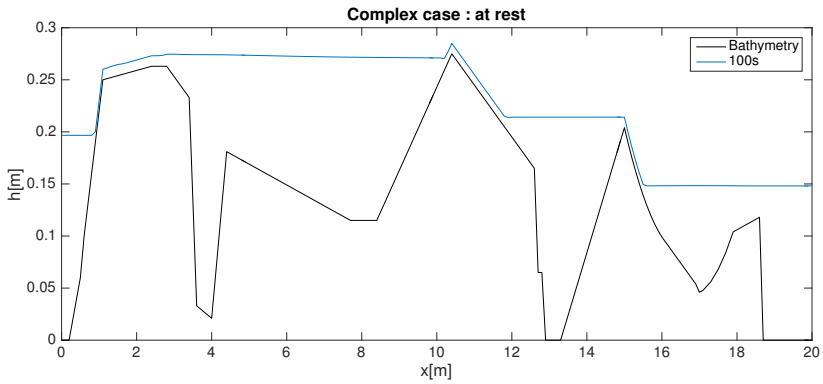


Figure 4.13: Water at rest

4.2.3 Summary

Through both examples, we saw the efficiency of our method and the advantages to have a thin layer of fluid. We saw that we were able to keep this layer in both cases that occur with small heights depicted in 4.2 and 4.3. This layer allows a continuity between the cells and avoid the problem related to heights close to 0. Therefore, we didn't need extra treatment related to the water front. We also notice that our algorithm doesn't produce any wiggles and that it conserves the mass.

Chapter 5

The implicit Wetting-Drying

In the previous chapter, we explained an explicit wetting drying method. We have seen that the method was really efficient for various reasons. Since explicit integration isn't fit to make simulation over a long period of time, we would like our method to integrate our equations implicitly. In addition to the thin layer technique, we add a method called the Patankar trick produced By Burchard *et al* in [5] used in the work of S. Ortleb and A. Meister in [13]. This technique was proven to be conservative and positive which are important features for a wetting-drying method. Another aspect of the method is that it allows to use an implicit scheme of high order. The aim of the thesis is to test this method through several test cases proposed by Balzano in [1] and Ritter (dam-break problem).

The chapter is organized in the following way. First of all, we will introduce some precautions related to the implicit time integration. Afterwards, we will explain the Patankar trick on the production-destruction equations and adapt it to the shallow water equations. Finally, we will use our new methods on different situations and make a review of the method based on the results we obtained.

5.1 Precautions for implicit schemes

As we said earlier, it is more difficult to use implicit schemes than the explicit schemes since we need to solve a system at each iteration. Since we need to solve a system ($f(\mathbf{U}) = 0$), it means that we require a non-linear solver that generally demands the Jacobian, $\frac{\mathrm{d}f}{\mathrm{d}\mathbf{U}}$. For example, we use the Newton-Raphson method that has this requirement. Since we have to work with a Jacobian in implicit, we need to add modifications to some functions in our equations.

Indeed, we need to take care of the functions that can bring instabilities in our algorithm. In explicit time integration, we were guaranteed to have heights above ϵ. However, in implicit it is possible to have a height which is below the thin layer. A possibility is to set the functions, that cause problems, to 0 when the height is below a certain threshold. Yet, we can't do that since those functions would be discontinuous and thus the Jacobian wouldn't exist at those discontinuity.

The solution to our problem is to "smooth" our functions. we will rewrite all the division by h to a function which isn't close to 0. Indeed, we will use the following :

$$\frac{\cdot}{h} \Rightarrow \frac{2 \ \cdot}{h + \max(\kappa, h)}$$

This way, the small height shouldn't cause any issue. They might be better ways to smooth our function since the Jacobian doesn't exist if $h = \kappa$ but it is still an improvement.

5.2 Patankar trick

Here, we introduce the Patankar trick from [13]. We will first deduce it from the production-destruction equations. Afterwards, we will translate this theory to our equations.

5.2.1 Production-destruction equations

The production-destruction equations are common in geobiochemical models. They describe the evolutions of different components c_i that interact with each other. Those components are positive and the equations are said to be conservative meaning that no mass is produced or destroyed. All in all, the equations are depicted below.

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} = P_i(\mathbf{c}) - D_i(\mathbf{c}) \tag{5.1}$$

$\mathbf{c}$ is the vector containing the concentrations of the different components, $\mathbf{c} = \begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix}^T$. P_i and D_i are called the production and destruction rate of the i^{th} component respectively. Those rates are defined by the behaviour of the other constituents with c_i. We can indeed define those rates as :

$$P_i(\mathbf{c}) = \sum_j p_{ij}(\mathbf{c}) \qquad D_i(\mathbf{c}) = \sum_j d_{ij}(\mathbf{c}) \tag{5.2}$$

We can define the terms, which are positive, in (5.2) as the following :

- $p_{ij}(\mathbf{c})$: the rate at which the j^{th} component transforms into the i^{th} component.
- $d_{ij}(\mathbf{c})$: the rate at which the i^{th} component transforms into the j^{th} component.

By definition, we have the following relationship $p_{ij} = d_{ji}$. In regards to the numerical integration, we need a scheme which allows to compute the concentrations at next time step, c_i^{n+1} based on the current one, c_i^n. It is thus essential for the scheme to have the following properties :

- It must be conservative : $\sum_i (c_i^{n+1} - c_i^n) = 0$
- The components must remain positive : $\mathbf{c} \geq 0$

An implicit method has been developed by Burchard *et al.* in [5] that has been proven to meet such requirements. Using the Euler explicit scheme, the next iterate would be :

$$c_i^{n+1} = c_i^n + \mathrm{d}t \left(\sum_j p_{ij}(c_i^n) - \sum_j d_{ij}(c_i^n) \right)$$

with $\mathrm{d}t$, the time step. The Patankar trick adds to the explicit formulation above some weights to the production and destruction rates that guarantee the above features but that make our scheme implicit. The scheme becomes then :

$$c_i^{n+1} = c_i^n + \mathrm{d}t \left(\sum_j p_{ij}(c_i^n) \frac{c_j^{n+1}}{c_j^n} - \sum_j d_{ij}(c_i^n) \frac{c_i^{n+1}}{c_i^n} \right) \tag{5.3}$$

Now that we have a positive and conservative scheme, let's apply it with numerical methods of a higher order. We can select any integration schemes of our choice. Thus, without loss of generality, we pick up the SDIRK3 scheme which is recapped in the following table.

$$\begin{array}{c|ccc} \gamma & \gamma & & \\ \gamma+\delta & \delta & \gamma & \\ 1 & \alpha & \beta & \gamma \\ \hline & \alpha & \beta & \gamma \end{array} \quad \text{with} \quad \begin{aligned} \alpha &= 1.2084966491760101 \\ \beta &= -0.6443631706844691 \\ \gamma &= 0.4358665215084580 \\ \delta &= 0.2820667392457705 \end{aligned}$$

We select this scheme because it is widely used and also because it has a negative coefficient that needs a special treatment. In this case, we solve an iteration in the following fashion. First of all, we apply the first step that doesn't need any special treatment because the Euler implicit method is already conservative and positive. From this step, we obtain $\mathbf{c}^{(1)}$.

$$c_i^{(1)} = c_i^n + \gamma\,\mathrm{d}t\left(P_i(\mathbf{c}^{(1)}) - D_i(\mathbf{c}^{(1)})\right)$$

The second step doesn't need any other special treatment as well for the same reason as before. Indeed, we are guaranteed that $\mathbf{c}^{(1)}$ is positive everywhere. After this step, we obtain $\mathbf{c}^{(2)}$.

$$c_i^{(2)} = c_i^n + \delta\,\mathrm{d}t\left(P_i(\mathbf{c}^{(1)}) - D_i(\mathbf{c}^{(1)})\right) + \gamma\,\mathrm{d}t\left(P_i(\mathbf{c}^{(2)}) - D_i(\mathbf{c}^{(2)})\right)$$

Actually, only the last case needs the use of the Patankar trick because now the components can reach negative values. We introduce intermediate variables z_i^n such that :

$$\begin{aligned} c_i^{(3)} &= z_i^n + \gamma\,\mathrm{d}t\left(P_i(\mathbf{c}^{(3)}) - D_i(\mathbf{c}^{(3)})\right) \\ c_i^{n+1} &= c_i^{(3)} \end{aligned}$$

z_i^n are the possible modifications of the possibly negative states :

$$\tilde{z}_i^n = c_i^n + \alpha\,\mathrm{d}t\left(P_i(\mathbf{c}^{(1)}) - D_i(\mathbf{c}^{(1)})\right) + \beta\,\mathrm{d}t\left(P_i(\mathbf{c}^{(2)}) - D_i(\mathbf{c}^{(2)})\right) \tag{5.4}$$

This is where the Patankar trick comes into play. We use the scheme in (5.3) in order to change $\tilde{z}_i^n$. This will guarantee the positiveness and conservation of our variables z_i^n. From the equation in (5.3), we have now :

$$\begin{aligned} z_i^n = c_i^n &+ \alpha\,\mathrm{d}t\left(\sum_j p_{ij}(c_i^{(1)})\frac{z_j^n}{\tilde{c}_j^{(1)}} - \sum_j d_{ij}(c_i^{(1)})\frac{z_i^n}{\tilde{c}_i^{(1)}}\right) \\ &+ \beta\,\mathrm{d}t\left(\sum_j p_{ij}(c_i^{(2)})\frac{z_i^n}{\tilde{c}_i^{(2)}} - \sum_j d_{ij}(c_i^{(2)})\frac{z_j^n}{\tilde{c}_j^{(2)}}\right) \end{aligned}$$

where :

$$\tilde{c}_i^{(k)} = \begin{cases} \tilde{z}_i^n & \text{if } \tilde{z}_i^n > \epsilon \\ c_j^{(k)} & \text{otherwise} \end{cases}$$

ϵ is a parameter that we are free to choose. We can draw our intention on the weights multiplying the rates that are swapped when the coefficient is β. This is due to the fact that when the coefficient is negative, the production rate becomes the destruction rate and vice-versa. The modifications z_i^n are obtained by solving a linear system $A\mathbf{z}^n = \mathbf{c}^n$. Indeed, the entries of the matrix, $a_{i,j}$ are :

$$a_{i,i} = 1 + \alpha \,\mathrm{d}t \sum_j \frac{d_{ij}(\mathbf{c}^{(1)})}{\tilde{c}_i^{(1)}} - \beta \,\mathrm{d}t \sum_j \frac{p_{ij}(\mathbf{c}^{(2)})}{\tilde{c}_i^{(2)}}$$
$$a_{i,j} = -\alpha \,\mathrm{d}t \sum_j \frac{p_{ij}(\mathbf{c}^{(1)})}{\tilde{c}_j^{(1)}} + \beta \,\mathrm{d}t \sum_j \frac{d_{ij}(\mathbf{c}^{(2)})}{\tilde{c}_j^{(2)}} \qquad i \neq j$$

We notice that only the diagonal entries are positive. It is proven in [13] that this scheme is conservative, unconditionally positive and first-order accurate in terms of the local truncation error. The last feature shows that we have the following relationship.

$$c_i^{n+1} = c_i^n + \mathcal{O}(\mathrm{d}t)$$

5.2.2 Shallow-water equations

Now that we wrote our scheme for the production-destruction equations, let's apply them to the Saint-Venant equations. In order to achieve that, we have to emphasize equations that resemble the equations in (5.1). This is the case of the ODE that describe the evolution of the mean water height in the cells, $\bar{h}_i$. We obtain this equation by adding the ODE's related to the water height from the discrete formulation.

$$l_i \frac{\mathrm{d}\bar{h}_i}{\mathrm{d}t} = q^*(X_i) - q^*(X_{i+1})$$

with q^* being the flow at the boundary computed using a numerical flux. The above equation can be rewritten in terms of a production and destruction rate. Indeed, we set the following :

$$p_{i,i-1} = \max(0, q^*(X_i)) \qquad d_{i,i-1} = \max(0, -q^*(X_i))$$
$$p_{i,i+1} = \max(0, -q^*(X_{i+1})) \qquad d_{i,i+1} = \max(0, q^*(X_{i+1}))$$

The terms above represent the exchanges of water between the elements and allow to make the distinction between the positive and negative flux contributions. We note that we have $p_{ij} = d_{ji}$. Since we are in a one-dimensional problem, the elements only share water with their neighbours. The equations can be rewritten in terms of the rates.

$$l_i \frac{\mathrm{d}\bar{h}_i}{\mathrm{d}t} = (p_{i,i-1} + p_{i,i+1}) - (d_{i,i-1} + d_{i,i+1}) \tag{5.5}$$

Now that we have our equations, we can apply the schemes we introduced earlier. First of all, we apply the first two steps of the SDIRK3 schemes since it guarantees the positivity of the height. For the third state, we will use the Patankar trick. The intermediate variable is called $\bar{h}_i^z$ and is equal to :

$$\begin{aligned}\bar{h}_i^z = \bar{h}_i^n &+ \frac{\mathrm{d}t}{l_i}\left[\sum_j \alpha\left(p_{ij}^{(1)}\frac{\bar{h}_j^z}{\tilde{h}_j^{(1)}} - d_{ij}^{(1)}\frac{\bar{h}_i^z}{\tilde{h}_i^{(1)}}\right) + \beta\left(p_{ij}^{(2)}\frac{\bar{h}_i^z}{\tilde{h}_i^{(2)}} - d_{ij}^{(2)}\frac{\bar{h}_j^z}{\tilde{h}_j^{(2)}}\right)\right] \\ &+ \frac{\mathrm{d}t}{l_i}\left[\alpha\left(p_i^B(U^{(1)}, t^n + \gamma\,\mathrm{d}t) - d_i^B(U^{(1)}, t^n + \gamma\,\mathrm{d}t)\frac{\bar{h}_i^z}{\tilde{h}_i^{(1)}}\right)\right. \\ &\left. + \beta\left(p_i^B(U^{(2)}, t^n + (\gamma+\delta)\,\mathrm{d}t)\frac{\bar{h}_i^z}{\tilde{h}_i^{(1)}} - d_i^B(U^{(2)}, t^n + (\gamma+\delta)\,\mathrm{d}t)\right)\right]\end{aligned} \tag{5.6}$$

The above expression introduces new notations as t^n which is simply the time at the n^{th} iteration. The superscripts of the rates p_{ij} and d_{ij} indicate at which value they are evaluated. For instance, if it is (1) it means that those rates are evaluated using the results after the first

step of our scheme, $\mathbf{U}^{(1)}$. However, the superscript B is used for the flux contributions at the boundaries. $\tilde{h}_i^{(k)}$ is defined as the following :

$$\tilde{h}_i^{(k)} = \begin{cases} \bar{h}_i^{\tilde{z}} & \text{if } \tilde{z}_i^n > \epsilon \\ \bar{h}_i^{(k)} & \text{otherwise} \end{cases}$$

ϵ is set to the the height of the thin layer of fluid by default. $\bar{h}_i^{\tilde{z}}$ makes reference to the state $\tilde{z}_i$ we obtained previously in (5.4). Here we show to compute the different terms in (5.6). Since we smooth the divisions, the fractions become:

$$\frac{\cdot}{\tilde{h}_i^{(k)}} \Rightarrow \frac{2 \cdot}{\tilde{h}_i^{(k)} + \max(\tilde{h}_i^{(k)}, \kappa)}$$

Regarding the boundary condition, we compute q^* using the expressions in the chapter dedicated to the discontinuous Galerkin method. We thus have :

$$\begin{aligned} p_1^B &= \max(0, q^*(X_0)) & d_1^B &= \max(0, -q^*(X_0)) \\ p_n^B &= \max(0, -q^*(X_{n+1})) & d_n^B &= \max(0, q^*(X_{n+1})) \end{aligned}$$

Now that we have all the elements to compute the expression in (5.6). We find $\bar{h}_i^z$ by solving a linear system $A\bar{h}_i^z = B$. Note that B is equal to $\bar{h}_i^n$ except for the first and the last entry due to the contribution of the boundary fluxes, $p_i^B(U^{(1)}, t^n + \gamma\,\mathrm{d}t)$ and $-d_i^B(U^{(2)}, t^n + (\gamma + \delta)$.

As soon as we have the values $\bar{h}_i^z$, we have to recover the corresponding state $\mathbf{U}^z$. That's why we will change the vector $\mathbf{U}^{\tilde{z}}$.

$$\mathbf{U}^{\tilde{z}} = \mathbf{U}^n + \alpha\,\mathrm{d}t\,\mathcal{L}(U^{(1)}) + \beta\,\mathrm{d}t\,\mathcal{L}(U^{(2)})$$

We will only change the variables related to the water height. We swap the water height's mean by those we computed, $\bar{h}_i^z$ while conserving the slope of our solution. However, we make an exception to the cells where the height's mean is below the height of the thin layer, ϵ. Indeed, for those elements, we decrease the order of our solution so that the heights and the flow become constant functions.

With those modifications, we obtain the vector $\mathbf{U}^z$. Finally, the last step of our scheme is found by solving the following non-linear system.

$$\mathbf{U}^{n+1} = \mathbf{U}^z + \mathcal{L}(\mathbf{U}^{n+1})$$

Doing this, we are guaranteed to have positive water heights and to respect the mass conservation. Here is a pseudo-code that sums up all the method.

input : $\mathbf{U}^n$, The current iterate
output : $\mathbf{U}^{n+1}$, The next iterate

Solve the first step of the SDIRK3 scheme.
$\mathbf{U}^{(1)} = \mathbf{U}^n + \gamma\mathcal{L}(\mathbf{U}^{(1)})$
Apply our slope limiter.
Solve the second step of the SDIRK3 scheme.
$\mathbf{U}^{(1)} = \mathbf{U}^n + \delta\mathcal{L}(\mathbf{U}^{(1)}) + \gamma\mathcal{L}(\mathbf{U}^{(2)})$
Apply our slope limiter.
Compute the intermediate value :
$\mathbf{U}^{\tilde{z}} = \mathbf{U}^n + \alpha\,\mathrm{d}t\,\mathcal{L}(U^{(1)}) + \beta\,\mathrm{d}t\,\mathcal{L}(U^{(2)})$
Compute the water means corresponding to $\mathbf{U}^{\tilde{z}}$, $h_i^{\tilde{z}}$.
if $\exists i \ s.t \ h_i^{\tilde{z}} \leq \epsilon$ **then**
  Compute the production and destruction rates, p_{ij} and d_{ij}.
  Solve the linear system to find $\bar{h}_i^z$.
  Obtain $\mathbf{U}^z$ by rearranging the vector, $\mathbf{U}^{\tilde{z}}$ to take into account $\bar{h}_i^z$.
else
  $\mathbf{U}^z = \mathbf{U}^{\tilde{z}}$
end
Solve the last step of the SDIRK3 scheme.
$\mathbf{U}^{n+1} = \mathbf{U}^z + \mathcal{L}(\mathbf{U}^{n+1})$
Use our slope limiter.

Algorithm 1: Implicit method of S.Ortleb and A.Meister

5.3 Results

In this section, we show some results we could obtain with the method we have just depicted. First of all, we will test it on the one-dimensional problems proposed by Balzano in [1]. The two first problems correspond to tides that cover and uncover different types of bathymetries. The first one is a simple slope and the second one has a convex bathymetry with a plateau. The third test case corresponds to a relief that has a basin. At first all the domain is covered by water and thereafter, the water is flowing through an open boundary. Thus at the end, only the water in the basin should remain.

5.3.1 Balzano

For the simulations, we pick up a time step which is the quadruple of the maximal CFL number. Normally, for realistic test cases, we pick up a time step which is over 50 times the CFL number so that implicit time integration remains attractive for simulations over a long period of time. However, for those cases, the wetting-drying problem is a problem only related to a tiny part of the domain and thus the time step is appropriate. Hence, for our idealized test cases, we have to pick up a time step close to the CFL number since the wetting-drying problem represents the major part of the simulation.

Regarding the other parameters, we will do as follows. We set the thin layer height to $0.05[m]$ and κ as well. For the manning coefficient, we pick up the same one as Balzano which is $n = 0.02$. We use a structured mesh with elements of length, $0.1[m]$. Regarding the boundary conditions, for all of the test cases we have a wall upstream. The length of our domain is equal to $6[m]$. The numerical flux, we choose, is the HLL flux.

Linear relief

In this test case, we will simulate tides that cover and uncover a slope. At the downstream boundary, we impose the height of the water to be equal to a sinusoidal function. The period of this function is $12[s]$ with an amplitude of $0.35[m]$ as we see below.

$$h_b(t) = 1 + 0.35 \sin\left(\frac{\pi}{6}t\right)$$

At first, the water elevation is set to $1[m]$ everywhere. This is depicted in the following figure where we can visualize the bathymetry.

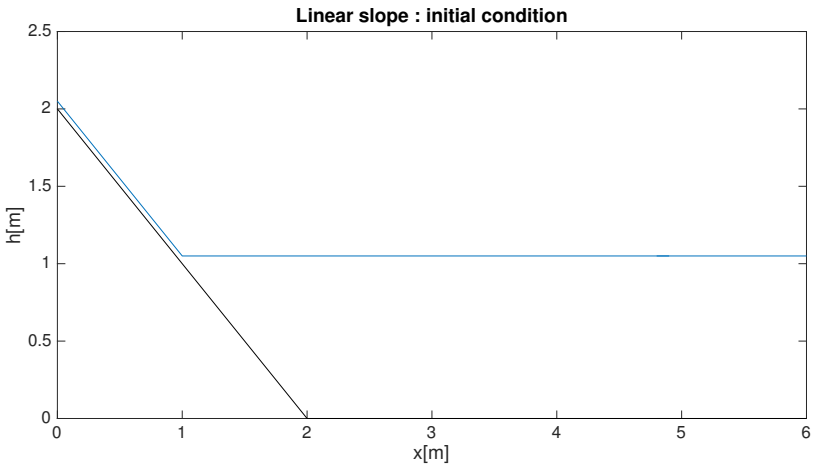


Figure 5.1: Initial condition : 1^{st} Balzano test case

We will consider two cases. The first one is related to the flooding tides and the other one to the ebbing tides. The following results are obtained after a period of time ($9[s]$). For the flooding tides, we pick up different results that lie between $10 - 12[s]$ and the ebbing tides between $17 - 21$ $[s]$.

Here are the results for the flooding tides.

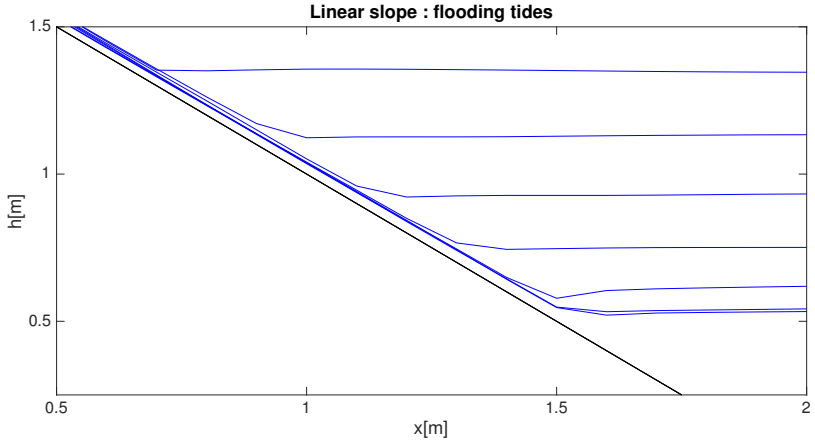


Figure 5.2: Flooding tides : 1^{st} Balzano test case

Here we note nothing particular. Indeed, it seems for this case we didn't need the use of the Patankar trick. For this test case, the theory we mentioned in the previous chapter is already sufficient. We note that the thin layer is conserved as in the explicit time integration. Aside from the fact that the thin layer has a harmful effect on the physical aspect of the solution, we can argue that our method works well.

Now that we've seen the flooding tides, we will analyze what we obtain for the ebbing tides.

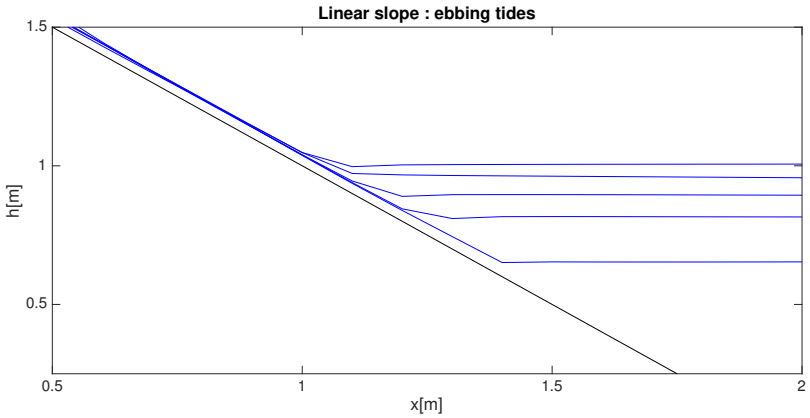


Figure 5.3: Ebbing tides : 1^{st} Balzano test case

As in the flooding tides, we didn't need the Patankar tricks. Actually, the conclusion is the same as before. We see that the thin layer is conserved and that the method works properly also in this case.

Convex relief

We consider the same periodic elevation at the downstream boundary as the previous case. The difference with the first case is that we have a more complex bathymetry. This surface is composed of two slopes and a plateau. This is depicted in the following graph with the initial condition which is the same as before.

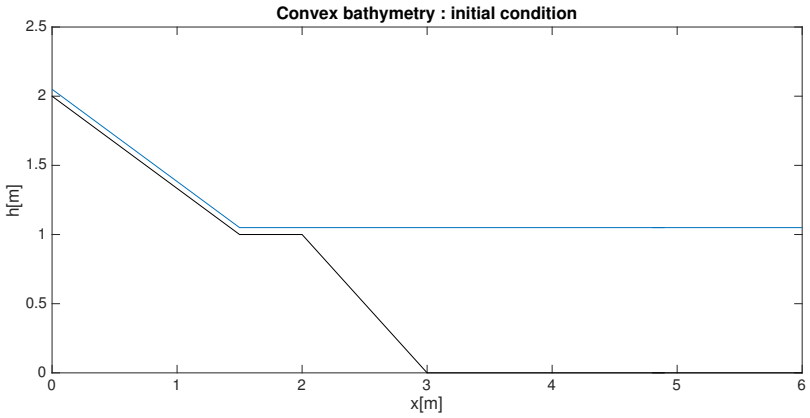


Figure 5.4: initial condition : 2^{nd} Balzano test case

As previously, we will analyze the ebbing and flooding tides and particularly the effect of the plateau on our solution. The solutions for the flooding tides were taken between $10 - 14[s]$ and the one for the ebbing tides between $16 - 20[s]$.

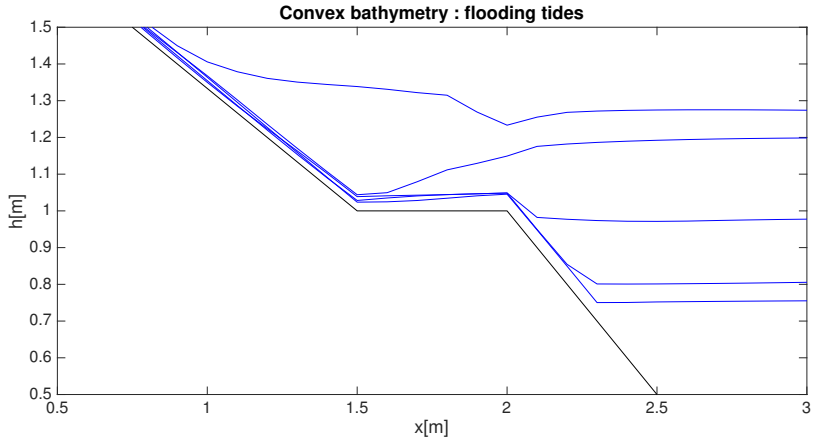


Figure 5.5: flooding tides: 2^{nd} Balzano test case

In this case, We observe that we conserve the thin layer as usual. We clearly see the effect of the plateau on our solution. Indeed, the solution seems to be discontinuous at the intersection of the slope and the plateau. Thus, we observe that in a case of flooding our method works.

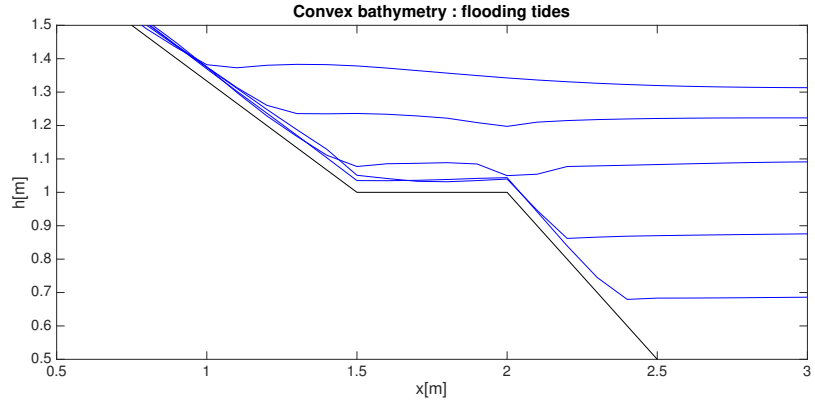


Figure 5.6: ebbing tides: 2^{nd} Balzano test case

Regarding the ebbing tides, we see again that the thin layer is conserved and the effect of the plateau is again visible. We observe that the water is drained more slowly on the flat area than on the steep one. We can thus see that our method is really efficient for the second test of Balzano.

One important aspect, we can observe, is the time to compute one iteration. Indeed, it is essential since we always want a simulation to end quickly. We computed the mean time of one iteration after we run our algorithm for a period of $10[s]$. We will visualize the effect of two parameters on the computation time : the number of elements and the time step. We obtain those results with ϵ and κ equal to $0.1[m]$.

First of all, we tested our algorithm with different structured meshes. We keep the time step equal to four times the maximum CFL number at each iteration. Here is the result we obtain in 5.7. We clearly see on this graph that the computational time with respect to the number of elements corresponds to a linear function. If we fit our result with a straight line, we obtain a slope that is more or less equal to $0.004[s/\#elements]$. It means that if we add one element to our mesh, the computational time of one iteration will be increased by 0.004 seconds.

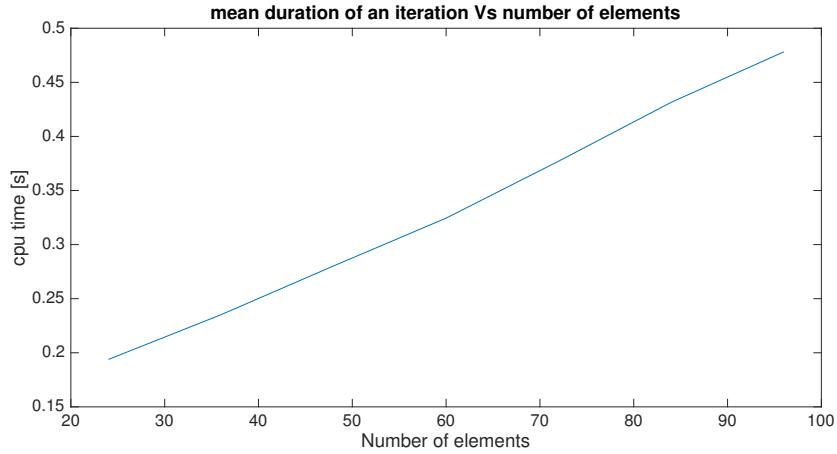


Figure 5.7: Computational time Vs Number of elements

Now let's look at the computational time with respect to the time step. The number of elements is fixed to 60. For the previous simulations, we picked up a time increment which was the quadruple of the maximum CFL number. For this problem, it was approximately $0.1[s]$ at each iteration. We will thus observe the evolution of the computational time when we choose a time increment between 0.025 and 2 seconds. The result is shown in 5.8. As previously, we observe a linear pattern. By performing a linear regression, we fitted our curve by a straight line with a slope of more less 0.5. Theoretically, it would mean that if we increase the time step by $1[s]$, we would increase the computational time of one iteration to $0.5[s]$. However, it is not really true because after a certain point, the time increment can lead to unstable results.

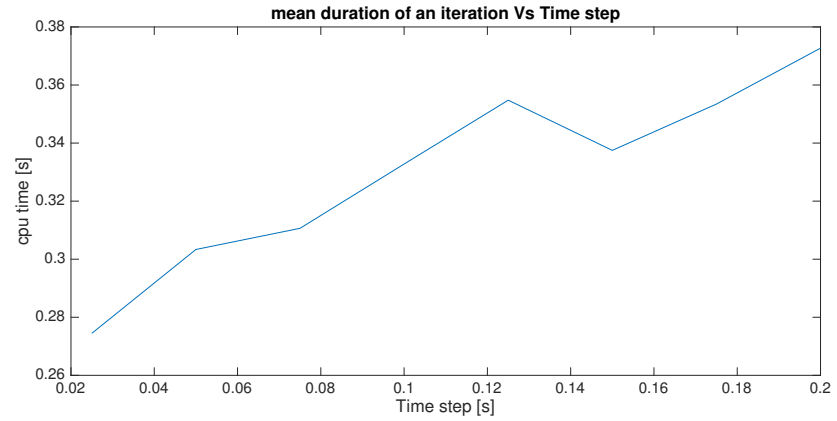


Figure 5.8: Computational time Vs Time step

All in all, in the above figures, we see that the computational time increases if we have a more precise mesh. Regarding the time step, we see that one iteration takes longer to compute but however, we would still consider taking a time step to be equal to $0.2[s]$ since we have less iterations to compute.

Basin

For the last test of Balzano, we have a bathymetry with a basin. This test has an outflow downstream that dries all the domain except the basin that should be full. To drain the water, we impose such boundary condition.

$$h_b(t) = 1.5 - \frac{t}{24}$$

As soon as the height is below the height of the thin layer ϵ, we switch for a transmissive boundary condition (at $t = 34.8[s]$). At the beginning, the water elevation is equal to $1.5[m]$ everywhere. All in all, the bathymetry and the initial condition is shown below.

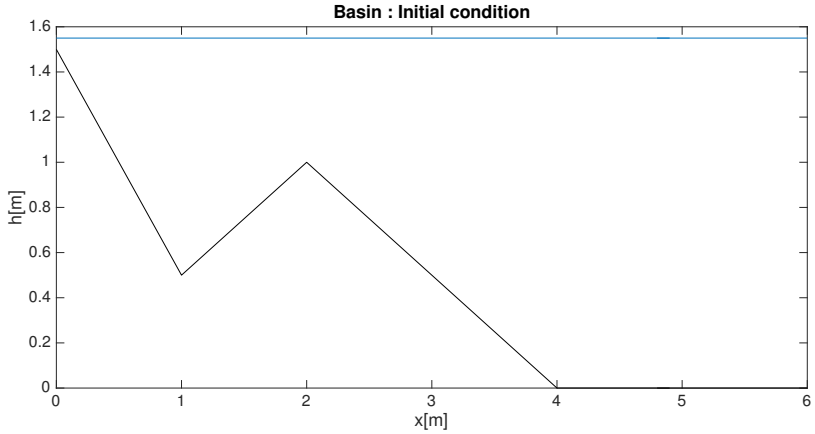


Figure 5.9: Initial condition: 2^{rd} Balzano test case

We run our algorithm for $100[s]$ in order to have a solution at rest. This is the result we obtain.

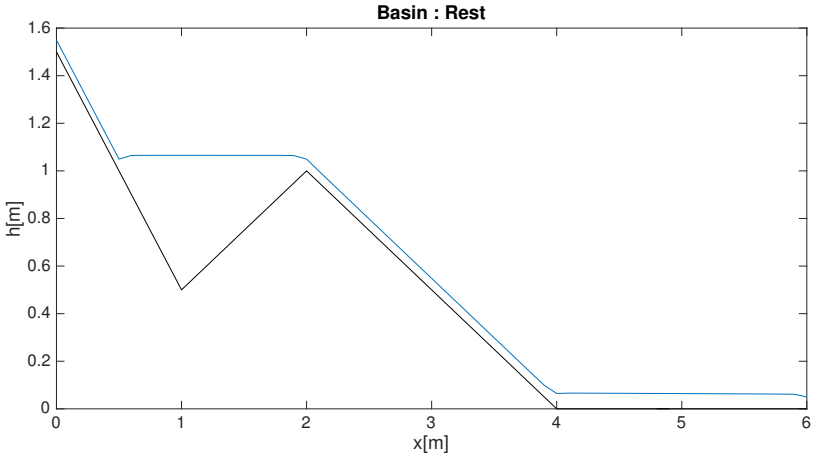


Figure 5.10: Rest : 2^{rd} Balzano test case

Here, we see that we get the behaviour we predicted. The thin layer is also maintained everywhere. However, we can see that the water level in the basin should normally be lower. This is due to the fact that we also need to keep a height of ϵ at the peak. Hence, the water level is ϵ too high. All in all, the algorithm works for that test case as well showing the efficiency of the method.

5.3.2 Dam-break problem

The last test case we propose is the Ritter problem. The problem corresponds to a water column that falls abruptly in a dry area. The initial condition is depicted below. For this problem, the water will propagate in the right directions where we put a transmissive condition at the end.

For this problem, we choose a structured mesh composed of 60 elements between 0 and 8 meters. The height of the thin layer and κ are set to $0.1[m]$. The manning coefficient is equal to 0 since we want to compare our solution with the analitycal solution.

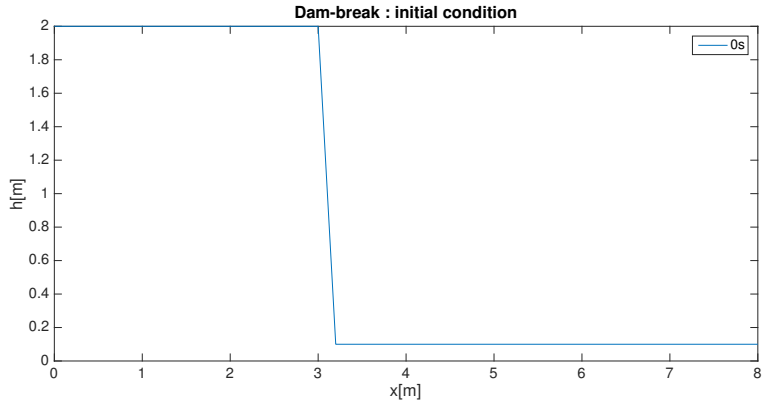


Figure 5.11: Initial condition : dam-break problem

First of all, we tried to run the simulation with a time step which is four times the maximum CFL number. However, this didn't work because our Newton-Raphson algorithm was unable to converge due to the effect of the small height. Besides, this problem is really difficult to solve in implicit due to the abrupt steepness of the solution between the wet and dry area. In order to get a solution, we resigned ourselves to take a time step smaller than the maximum CFL number and set it to $0.005[s]$.

The advantage of this problem is that it has an analytical solution. Suppose that the initial height of the water column is h_0 and that the frontier between the dry and wet areas is initially set at $x = 0$. The corresponding analytical solution is divided into three parts and is depicted below.

$$h(x,t) = \begin{cases} h_0 & \text{if} \quad x \leq -c_0 t \\ \frac{(2c_0 - x/t).^2}{9*g} & \text{if} \quad -c_0 t \leq x \leq 2c_0 t \\ 0 & \text{if} \quad 2c_0 t \leq x \end{cases}$$

with $c_0 = \sqrt{g h_0}$. Now, we can compare the analytical solution and our simulation. We obtain the following results.

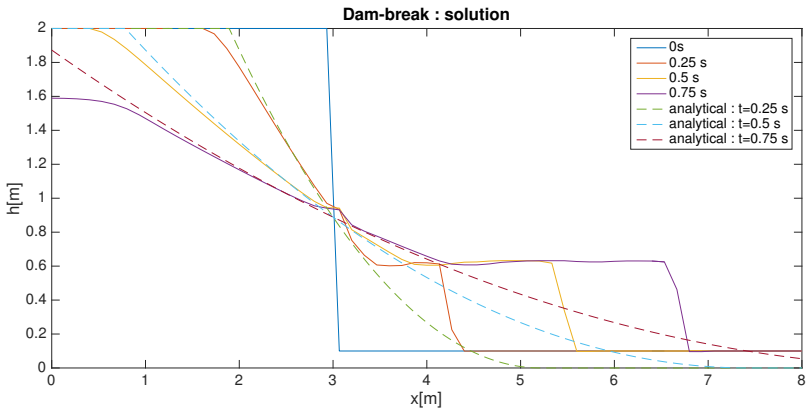


Figure 5.12: Solution : dam-break problem

If we look at our results from the left to the right, our simulation approaches the analytical solution quite well after a certain moment. Indeed, our algorithm clearly suffers from the wetting-drying problem. The fact that we add a thin layer of fluid completely changes the profile of the solution. We could diminish the height of the thin layer but it would demand to reduce the time step that was really too small for implicit time integration.

5.3.3 Summary

In our tests, we applied our algorithm on well-known tests. For the Balzano test cases, the method works fine but however we saw that when we have a thin layer the solution can become imprecise as in our solution for the the Ritter solution in 5.12. We saw in all the cases that we were able to maintain a thin layer.

Regarding the Patankar trick, we rarely have to use it. Indeed, the test cases we tested weren't too complex. Besides, we had to choose a small time step in order to be stable and especially for the Ritter problem because our program was extremely sensitive to the small heights. This shows once again that the wetting-drying problem is not easy at all.

However, we can note that the Patankar trick does its job when it faces correct circumstances. When it faces an unstable situation, the method doesn't create a miracle and does something unstable as in the dam-break problem.

We can suppose that the Patankar trick adds a considerable amount of computational time since we need to solve a linear system. Let's test it. We run the same simulation of the first Balzano case twice with two different cases : one where we don't use any Patankar trick and another one where we use it at every iteration. The parameters are the same as previously. The mean computational times for one iteration in both cases are :

- Without Patankar trick : $0.314[s]$
- With Patankar trick : $0.351[s]$

We thus see an increase by more or less 12% if we use the Patankar trick method. This is not an important increase since solving the Newton-Raphson algorithm demands to solve a linear system several times. Besides, this increase is irrelevant if we use the Patankar trick rarely as in our case.

Chapter 6

Conclusion

Through this thesis, we derived and analysed the Saint-Venant equations. From them, we derived the compatibility equations along with their dual form. This leads us to the second chapter. Indeed, we saw that those equations were useful for computing the boundary conditions and the numerical fluxes. We saw throughout this chapter how to implement the discontinuous Galerkin algorithm by discussing the transition between the weak and discrete formulation. Afterwards, we talked about wetting-drying methods in explicit and implicit time integration. We introduced the thin layer method as well as a modification to the slope limiter from the previous chapter. Then, we explained the work of Meister A. and Ortleb S. ([13]).

In the last chapters, we performed different test cases. The goals of the first ones were to test the thin layer method with an explicit time integration. The purpose of the first problem was to observe the behaviour of the thin layer of fluid in two different cases :

- Flooding : the water fills in the layer
- Draining : The gravity and hydrostatic forces tend to dry our layer

We saw in both cases that our method was keeping the layer efficiently. Afterwards, we simulated a difficult problem with some complex bathymetry. This allowed to show that our method should work in any cases. We saw that our algorithm was able to run without any difficulty showing that the method is efficient. Besides, this test allowed us to check that the mass was conserved throughout the simulation.

However, we knew that with explicit time integration, there exist many wetting-drying methods in the literature that work. The thing we are interested in is the implicit time integration. As a reminder, our objective was to test the work done in [13] in combination with with a thin layer of fluid in order to use implicit schemes of higher order.

We decided to test our method on classical test cases (Balzano and Ritter). For Balzano, the algorithm gives correct solutions. Despite having a thin layer of fluid, It reproduces quite well the physics for those problems. However, the effect of the thin layer has negative effects on the results for the dam-break problem. Besides of being unable to reproduce the analytical solution, we had to pick a time step below the CFL number which doesn't make sense if we use an implicit scheme since an explicit scheme does the job quite easily.

We also made some analysis on the computational time. We saw that the computational time with respect to the number of elements in the mesh and the time step has a linear profile. We also compare the time of computing an iteration with or without the Patankar trick method in [5]. We saw that with the Patankar trick the computational time increases by 12%. However, for simple problems, we just occasionally need this trick, thus this increase is not relevant. Even though we don't use it regularly, this technique does its job as long as we are not in an unstable situation.

If we had additional time, we would have considered extending the thesis in the following manner. We could have coded the discontinuous Galerkin problem for two dimensional problems. We would have considered the Thacker test in [15] which has an analytical solution. We could have considered also realistic cases as the Mont Saint-Michel bay, the Scheldt estuary (6.1) or the Mekong delta (6.2).

Figure 6.1: Scheldt estuary

Figure 6.2: Mekong delta

Another thing we would have considered is to improve our code. Indeed, We noticed that our algorithm was extremely sensitive to the small height as we saw in the dam-break problem. Indeed, when the situation was a bit too hard to solve, generally the Newton-Raphson would crash. Hence, there is most likely a way to improve our code.

In this thesis, we considered only one wetting-drying method but it would be interesting to analyze another algorithm different from the thin layer approach. We could have compared both methods and highlight the defaults and qualities of both methods. For example, we could have tested the asymptotic approach of Eric Deleersnijder in his recent working note in [8] and assessed its relevance. Another recent article, we could have tested, is the one propose by Adam Candy in [6] which is also a method in the family of thin layer algorithms. Indeed, it could have been interesting to compare two methods of the same category.

List of symbols

Here is a list of the symbols that were introduced in this thesis :

- P : wetted perimeter $[m]$
- z : bed elevation $[m]$
- ρ : density $[kg/m^3]$
- A : area contained at a channel section $[m^2]$
- Q : flow $[m^3/s]$
- V : flow velocity $[m/s]$
- $\mathrm{d}x$: length of the control volume $[m]$
- S_0 : bed slope
- τ : shear stress along the bed $[Pa]$
- R : hydraulic radius $[m]$
- S_f : friction slope
- n : manning coefficient
- I_1 : cross-section momentum $[m^3]$
- I_2 : change in section's width $[m^2]$
- u : horizontal speed $[m/s]$
- β : Boussinesq approximation
- $\mathbf{U}$: vector containing the unknowns
- $\mathbf{F}$: vector containing the convective terms
- $\mathbf{s}$: vector containing the source terms
- h : water height $[m]$
- q : flow divided by the channel width $[m^2/s]$
- c : relative celerity $[m/s]$
- Fr : Froude number
- X_i : position of the i^{th} node $[m]$
- $\cdot_i^-$: related to the left side of the i^{th} element
- $\cdot_i^+$: related to the right side of the i^{th} element
- ξ : position in the space of the parent element $[m]$
- ϕ_i : basis function
- $\langle\cdot\rangle$: volume integral
- $\langle\langle\cdot\rangle\rangle$: surface integral
- $\cdot^*$: obtained from a numerical flux
- η : water elevation $[m]$
- ϵ : height of the thin layer $[m]$
- κ : parameter for smoothing unstable functions
- c_i : concentration of the i^{th} component $[mol/m^3]$
- p_{ij} : rate at which the j^{th} component transforms into the i^{th} component $[1/s]$
- d_{ij} : rate at which the i^{th} component transforms into the j^{th} component $[1/s]$

Bibliography

[1] A. Balzano. Evaluation of methods for numerical simulation of wetting and drying in shallow water flow models. *Coastal Engineering*, 34:83–107, 1998.

[2] P. D. Bates and J-M. Hervouet. A new method for moving-boundary hydrodynamic problems in shallow water. *Proceedings: Mathematical, Physical and Engineering Sciences*, 455:3107–3128, 1999.

[3] O. Bokhove. Flooding and drying in discontinuous Galerkin finite-flement discretizations of shallow-water equations. Part 1: one dimension. *Journal of Scientific Computing*, 22:47–82, 2005.

[4] S. Bunya, E. J. Kubatko, J. J. Westerink, and C. Dawson. A wetting and drying treatment for the Runge–Kutta discontinuous Galerkin solution to the shallow water equations. *Computer Methods in applied Mechanics and Engineering*, 198:1548–1562, 2009.

[5] H. Burchard, E. Deleersnijder, and A. Meister. A higher-order conservative Patankar-type discretization for stiff systems of production–destruction equations. *Applied Numerical Mathematics*, 47:1–30, 2003.

[6] A. S. Candy. An implicit wetting and drying approach for non-hydrostatic baroclinic flows in high aspect ratio domains. *Advances in Water Resources*, 102:188–205, 2017.

[7] B. De Brye, A. De Brauwere, O. Gourgue, T. Karna, J. Lambrechts, R. Comblen, and E. Deleersnijder. A finite-element, multi-scale model of the Scheldt tributaries, river, estuary and ROFI. *Coastal Engineering*, 57:850–863, 2010.

[8] E. Deleersnijder. Establishing the consistency of the asymptotic formula of Deleersnijder (2002) with the exact one- dimensional dam break flow solution of Ritter (1892). *http://hdl.handle.net/2078.1/196717*, 2018.

[9] O. Gourgue, R. Comblen, J. Lambrechts, T. Karna, V. Legat, and E. Deleersnijder. A flux-limiting wetting–drying method for finite-element shallow-water models, with application to the Scheldt estuary. *Advances in Water Resources*, 32:1726–1739, 2009.

[10] T. Karna, B. De Brye, O. Gourgue, J. Lambrechts, R. Comblen, V. Legat, and E. Deleersnijder. A fully implicity wetting-drying method for DG-FEM shallow water models, with an application to the Scheldt estuary. *Computer Methods in Applied Mechanics and Engineering*, 200:509–524, 2011.

[11] Y. Le Bars, V. Vallaeys, E. Deleersnijder, E. Hanert, L. Carrere, and C. Channeliere. Unstructured-mesh modeling of the Congo river-to-sea continuum. *Ocean Dynamics*, 66:589–603, 2016.

[12] J. J. Leenderste. Numerical simulation of water quality in coastal waters and estuaries. *Rand Corporation Report RM-6230-rc*, 1, 1970.

[13] S. Ortleb and A. Meister. On unconditionally positive implicit time integration for the DG scheme applied to shallow water flows. *International journal for numerical methods in fluids*, 76:69–94, 2014.

[14] S. Stelling, A. K. Wiersma, and J. B. T. M. Willemse. Practical aspect of accurate tidal computations. *Journal of Hydraulic Engineering*, 112:802–816, 1986.

[15] W.C. Thacker. Some exact solutions of the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508, 1981.

[16] D. Vincent, O. Karatekin, V. Vallaeys, , A.G. Hayes, M. Mastrogiuseppe, C. Notarnicola, V. Dehant, and E. Deleersnijder. Numerical study of tides in Ontario Lacus, a hydrocarbon lake on the surface of the Saturnian moon titan. *Ocean Dynamics*, 66:461–482, 2016.

[17] L. Zhao, J. Mao, X. Bai, X. Liu, T. Li, and J.J.R. Williams. Finite element implementation of an improved conservative level set method for two-phase flow. *Computers & Fluids*, 100:138–154, 2014.

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