EPFL

Thèse n° 8195

Numerical simulation of sediment dynamics with free surface flows

Présentée le 18 février 2021

Faculté des sciences de base
Groupe Picasso
Programme doctoral en mathématiques

pour l'obtention du grade de Docteur ès Sciences

par

Arwa MRAD

Acceptée sur proposition du jury

Prof. D. Kressner, président du jury
Prof. M. Picasso, Prof. A. Caboussat, directeurs de thèse
Prof. S. Boyaval, rapporteur
Prof. N. Parolini, rapporteur
Prof. S. Deparis, rapporteur

École
polytechnique
fédérale
de Lausanne

2021

In a time of destruction,
create something.
— Maxine Hong Kingston

To my beloved ones: Jalila, Samir, Nama, Khoubayb, Ons, Julia and Fedi.
To Matthieu.

Acknowledgements

A doctoral thesis is significantly long and at times a tough journey, which I would not have been able to complete alone. During the past four years there were surely moments of doubt, but most importantly a lot of hard work and perseverance. I would like to express my gratitude to some of the people who helped me make this happen. Part of this thesis belongs to you.

First and foremost, I want to thank both of my supervisors, Prof. Marco Picasso and Prof. Alexandre Caboussat. I would like to thank Prof. Marco Picasso for allowing me to work within his group for these years on such an interesting subject. I want to thank him for always being open for discussions and for always being so helpful, by providing a good book, good advice on a whiteboard, and constructive criticism that pushes my work for the better. I also want to particularly thank him for caring about the work atmosphere and trying to bring the group together (even in the hard covid times). I also want to thank Prof. Alexandre Caboussat for his continous supervision and the time he dedicated to discuss about my work without hesitation. Having you as a supervisor, with such a positive mindset and energy definitely had a very positive impact on my work and I really learned a lot from you over the years, so thank you. It has been a real pleasure to work with both of you during these years and everything I have learned from you is priceless. I am without a doubt many orders of magnitude better than when I started.
Next, I would like to thank my thesis committee Dr. Sébastien Boyaval, Dr. Nicolas Parolini, and Dr. Simone Deparis. Their insightful comments, questions, and feedback are very appreciated. I would also like to thank Dr. Daniel Kressner who accepted to be my thesis jury president. I would also like to thank the Swiss Confederation for providing me with a "Swiss Government Excellence Scholarship" to financially support my research. I want to particularly thank them for organizing along with their program a lot of events and outings that helped me discover Switzerland and make lifetime friends.

I would also like to thank Dr. Alexandre Masserey and Julien Hess from Ycoor Systems SA for allowing me to work with and contribute to the software cfsFlow++. In particular, I want to thank Julien Hess, with whom I had various discussions about how to best tackle implementation issues, for always being so helpful, patient, and insightful.

Thanks to a great atmosphere in the group throughout these years, I have also enjoyed spending time with my (sometimes crazy) colleagues. I want to thank my office mate

Dimitrios for always being there when I needed him, particularly during the first roller-coaster years of my Ph.D. Thank you for those bike-rides and for your greek jokes that always helped. I want to thank Samuel for bringing such positive energy to the group. Thank you Samuel for always showing up whenever I needed a push and a little piece of advice. I want to thank Paride, for his joyful and caring personality. I want to thank Léo and Emile as well for the various fruitful conversations about different topics we had during coffee and lunch breaks. A big thanks to the rest of the Mathicse group members. I want to thank other friends from EPFL that I met during the past years and with whom I had a lot of laughter such as Tomas R., Alfonso C., Bassel, Lydia A., and Manuel A.. You all made this journey a whole lot better. I also want to particularly thank Javier, one of the first people I met at EPFL, he was always so funny, even with his harsh criticism (ha!) I also want to thank Math (he will recognize himself) for the good times we had together during my first year of Ph.D.

I am profoundly grateful to all my friends outside EPFL in Switzerland with whom I shared a lot of activities outside work. I want to thank Hamed for the endless times he spent trying to help me stress less, thanks to his Iranian jokes and food. I thank his wife Sahar too. I want to thank Annelaure for her social parties and Fabienne for being my "swiss" sister. I want to thank Salma for always being there to listen to my drama. Thank you Jonathan E. for cheering me up with guitar covers. The list is long but I want to thank Alejandra, Ignes, Hannah, Lucia, Thomas, Elise, Anna, Davide, Giulia . . . You each played a role at some point in this journey. I also want to thank my friends all over the globe. Thank you Ameni El A. for your priceless visits and loving heart. Thank you, Maria, for being my soulmate for the last ten years. Thank you, Houssem Ch., Mohamed O. for sharing my favorite sports during my Tunisia holidays. Thank you Hassen M., Abdelhay, Imen Z.

I also want to thank, from the bottom of my heart, my partner Matthieu G. with whom I shared many of my favorite adventures. You made the last two years of my Ph.D. so much better in so many aspects. Thank you for being my second family away from mine. Thank you Philippe and Pascal for always treating me with hospitality and love.

A big thanks for the support I always had from Tunisia. I want to thank Khouloud for being the most supportive cousin. I want to thank my cousins Ons, Yassmine, Safa, and Med Amine for their unconditional love. My aunts Amira, Lamia, and Awatef for the precious time I get to spend with them during my summer breaks and for always believing in me. A big thanks to the whole family and the rest of my cousins, Bochra, Wafa, Marwa . . . for their support. Thank you Jalila for being the best mom I can dream of, for supporting me and checking-in every single day. Thank you, Samir for being a very present and supportive dad even from afar. Thank you dear sister Nama for all the good times I get to spend with you when I go home, and thank you Med Khalil. Thank you my brother Khoubayb and your wife Ons, for all the love (and food) you give me when I am around.

Lausanne, December 10, 2020 A. M.

Abstract

We present a numerical model for the simulation of 3D mono-dispersed sediment dynamics in a Newtonian flow with free surfaces. The physical model is a macroscopic model for the transport of sediment based on a sediment concentration with a single momentum balance equation for the mixture (fluid and sediments). The model proposed here couples the Navier-Stokes equations, with a volume-of-fluid (VOF) approach for the tracking of the free surfaces between the liquid and the air, plus a nonlinear advection equation for the sediments (for the transport, deposition, and resuspension of sediments).
The numerical algorithm relies on a splitting approach to decouple diffusion and advection phenomena such that we are left with a Stokes operator, an advection operator, and deposition/resuspension operators. For the space discretization, a two-grid method couples a finite element discretization for the resolution of the Stokes problem, and a finer structured grid of small cells for the discretization of the advection operator and the sediment deposition/resuspension operator. SLIC, redistribution, and decompression algorithms are used for post-processing to limit numerical diffusion and correct the numerical compression of the volume fraction of liquid.
The numerical model is validated through numerical experiments. We validate and benchmark the model with deposition effects only for some specific experiments, in particular erosion experiments. Then, we validate and benchmark the model in which we introduce resuspension effects. After that, we discuss the limitations of the underlying physical models.

Finally, we consider a one-dimensional diffusion-convection equation and study an error indicator for the design of adaptive algorithms. First, we consider a finite element backward scheme, and then, a splitting scheme that separates the diffusion and the convection parts of the equation.

Keywords: Finite elements, finite volumes, finite differences, particle flow, sediment transport, mixture model, free-surface flow, operator splitting, VOF, advection, Navier-Stokes.

Résumé

Nous présentons un modèle numérique pour la simulation 3D de la dynamique de sédiments mono-dispersés dans un écoulement Newtonien à surfaces libres. Le modèle physique est un modèle macroscopique de transport de sédiments basé sur une concentration de sédiments satisfaisant une equation d'équilibre dynamique pour le mélange (fluide et sédiments). Le modèle proposé ici couple les équations de Navier-Stokes, avec l'approche du volume de fluide (VOF) pour le suivi de la surface libre entre le liquide et l'air, et une équation d'advection non linéaire pour les sédiments (pour le transport, la déposition et la resuspension des sédiments). L'algorithme numérique repose sur un algorithme à pas fractionnaires (splitting) pour découpler les phénomènes de diffusion et d'advection et ainsi obtenir un opérateur de Stokes, un opérateur d'advection et des opérateurs de déposition/resuspension. Pour la discrétisation spatiale, une méthode à deux grilles couple une discrétisation par éléments finis pour la résolution du problème de Stokes, et une grille structurée de petites cellules pour la discrétisation de l'opérateur d'advection et et les opérateurs de déposition/resuspension.
Des algorithmes SLIC, redistribution et décompression sont utilisés pour le post-traitement afin de limiter la diffusion numérique et corriger la compression numérique de la fraction de liquide. Le modèle numérique est validé par des expériences numériques. En premier lieu, nous validons et comparons le modèle avec les effets de déposition uniquement pour certaines expériences spécifiques, en particulier des expériences d'érosion. Ensuite, nous validons et comparons le modèle lorsque nous introduisons les effets de resuspension. Après, nous discutons des limites sous-jacentes du modèle physique.

Enfin, nous considérons une équation de diffusion-convection unidimensionnelle et étudions un indicateur d'erreur utilisé pour la conception d'algorithmes adaptatifs. Tout d'abord, nous considérons un schéma d'éléments finis explicite, puis un schéma de splitting qui sépare les parties diffusion et convection de l'équation.

Mots clés : éléments finis, volumes finis, différences finies, flux de particules, transport de sédiments, modèle mixte, flux à surface libre, approche de splitting, VOF, advection, Navier-Stokes.

Contents

1 Introduction

The modeling of sediment transport in rivers, lakes, or shores is relevant in hydraulic engineering. Sediments influence structural damages, operations efficiency, and management, but also influence the efficiency of energy production in dam retention lakes. Moreover, the accumulation of river sediments modifies the natural environment, which might have significant consequences for hydraulic energy production [1] or environmental regulations.

Our goal is to numerically model and simulate fluid flows with free surfaces and sediment, in three space dimensions. Ultimately, we want our model to cover a wide range of environmental and industrial applications for sediment transport. Since sedimentation is a complex physical phenomenon, in the literature, it has been investigated and simulated using several different approaches yielding different trade-offs between robustness, speed, stability, and accuracy of the proposed models. In the first part of this chapter, we give a brief introduction to sediment dynamics [2, 3]. In the second part, we present a state of the art of sediment models and an overview of our model.

1.1 Introduction to sediment dynamics

Sediment transport is the movement of solid particles (called sediments), typically due to a combination of the gravity force acting on the particles, and/or the movement of the fluid within which the sediment is carried [2–6]. Sediment dynamics occur in various natural systems such as rivers, lakes, seas, and oceans. The sediment particles can be sand, gravel, clay, or boulders [5]. In an aquatic environment, sediments can be suspended (when they float in the water column) or bedded (when they settle at the bottom of the waterbody). Physicists usually refer to sediments in a waterbody by the stream load. A stream load consists of the wash-load (or the dissolved load), the suspended load, and the bed load, as illustrated in Figure 1.1 [7, 8].

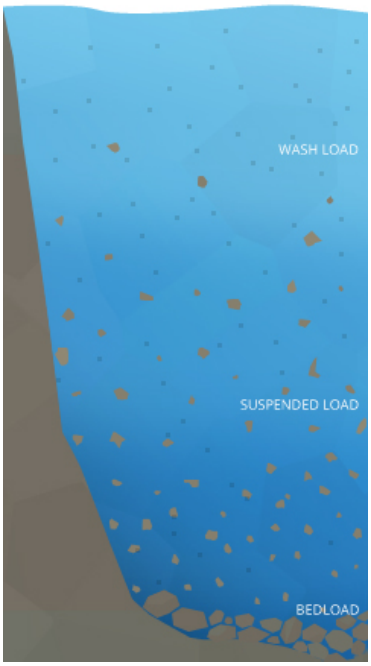


Figure 1.1 – There are three types of stream load in an aquatic environment: the washload load, the suspended load and the bed load. Image downloaded from https://www.fondriest.com/environmental-measurements/parameters/hydrology/sediment-transport-deposition. Last visited on February 26,2020

- **The bedload:** The bedload is the portion of sediments that maintain intermittent contact with the stream bed. These particles can move by rolling, sliding, or bouncing along the bed. However, their movement is not necessarily continuous neither uniform [2, 6]. This type of movement that we refer to by bedload transport happens when the force of the water flow is strong enough to overcome the weight and cohesion of the particles. These particles typically do not move as fast as the water around them, though, since the flow rate is not big enough to fully suspend them [3, 9]. For smaller particles, bedload transport can occur during slow flows. For larger particles, faster flows are required. As a result, in situations where the flow rate is high enough, some of the smaller bedload particles can be pushed up into the water column and become suspended [2, 3, 6, 10].

- **The suspended load:** The suspended load is the amount of sediment particles that are carried within the water column by the water stream [2, 3, 6, 11]. Suspended loads don't exist if there is no flow. The water stream creates small currents. Those currents help uplift the particles above the bed. The size of the particles that can be suspended usually depends on the flow rate. However, in most streams, the suspended load consists of smaller-sized particles. On the other hand, unless the flow rate rises, bigger particles are likely to fall through the currents down to the bottom. Moreover, suspended sediments themselves can become part of the bedload when the flow rate slows down [2, 3, 6, 9, 11].

- **The washload:** The washload is a subset of the suspended load. It consists of the smallest particles among those suspended. Typically their diameter is smaller than 10^{-3} [mm]. As opposed to suspended load, the washload particles remain permanently suspended even when there is a slow flow or even more, when no flow exists any longer. These particles are small enough to bounce off the water molecules and float. Nevertheless, the dissolved load and the suspended load are hard to tell apart during flow periods. The washload is then carried along the stream without deposition [2, 3, 10, 11].

Many parameters affect the movement of sediments. For a start, whether they are part of the suspended load or the bedload, the sediments' movement is particularly linked to the fluid flow. When the near-bed fluid velocity rises to a certain level, some sediment particles begin to move non-continuously. The higher the fluid velocity, the more sediment particles are involved in this movement. For instance, a particle in the bedload would move according to complex dynamics such as [3, 10, 12]:

- Suspension: the particles float in the liquid mass and remain in suspension, when the lift force balances that of gravity.
- Sliding: the particles move smoothly (slide) on the stream bed.
- Rolling: the coarser particles can roll instead of sliding. Their size and shape allow them to rotate.
- Saltation or resuspension: the particles can move on the stream bed through jumps. The interaction between the lift and drag forces help remove these particles from the bed and be carried by the fluid, before being transported back to the surface [13].

When a particle is finally suspended, it can continue to move when there is enough flow. However, when the fluid velocity drops below a certain level, sediments settle down on the sea or the river bed as an example [3, 6, 8]. Other parameters that affect the sediment movement are the force of gravity and the size and characteristics of the sediments as well as the fluid. The gravity acts to move the particles down along the surface on which they are resting. The grain size is an important factor in determining the settling velocity of granular sediment particles. Heavier grains may settle quickly, while small grains like silts and clays may stay in the water column for a longer time. Stokes law is normally used to calculate the settling velocity of fine particles [3, 7, 12, 14].

1.2 Physical modeling of sediment dynamics

In the past decades, numerical models have been playing an essential role in assessing sediment transport problems in coastal and river engineering. Over time, much research has been directed to the development and improvement of existing models. In the next subsections,

we discuss some of the popular physical representations of the recent models, as well as the numerical methods used for the approximation of such models.

Among the simplest and earliest mathematical models that can be applied to problems of suspended sediment motion in river-engineering is the so-called classical diffusion model [15, 16]. The basic assumptions for such a model are fine sediment particles and very low concentrations. Navier-stokes equations are used to describe the water motion while an advection-diffusion equation is used to describe the sediment motion. Applications of this model have been reported in [17–20]. These studies, however, indicated that phenomena like local scour, flow-induced erosion, or landslide-induced water waves are difficult to model with such classical techniques. To represent these phenomena, proper treatment of the physical behavior of the sediment, as well as the sediment-water interaction, is necessary.

In recent decades, the modeling of sediment transport in a flow began to rely on a multiphase model. A very widespread model is to treat the liquid and the sediment as two distinct phases and is referred to as the two-phase model. Each phase can be classified as continuous or disperse. Based on this classification, a two-phase model can be classified into Lagrangian-Lagrangian, Eulerian-Eulerian, and Eulerian-Lagrangian models.

- **Eulerian-Lagrangian models:** An Eulerian-Lagrangian model treats the water phase as a continuum while tracking the movement of each sediment particle at the microscopic scale [21–24]. For macroscopic two-phase flow applications, where macroscopic particles are dispersed in a continuous fluid, interface-resolved simulation methods are commonly used. This class of two-phase applications is also referred to as particle-laden flows. The interface-resolved methods directly compute the hydrodynamic force on the dispersed particles and therefore are also called DNS (Direct Numerical Simulation methods). Various DNS methods exist, including the arbitrary Lagrangian-Eulerian method [25], the lattice Boltzmann method [26], and the fictitious domain method [27]. In the fictitious domain approach, the complex geometry is embedded in a larger, simpler domain. The discretization is performed in this larger domain and Lagrange multipliers are used to enforce the correct boundary conditions at the (fluid/solid) interfaces. Moreover, in non-macroscopic but dilute situations, methods where the disperse phase is microscopically modeled by an additional concentration are used and are well-validated at low concentration levels [28, 29]. However, when applied to high concentration problems, these models require a very large number of particles to track the dispersed phase and are therefore very time and resources consuming.

- **Lagrangian-Lagrangian models:** A Lagrangian-Lagrangian model (also called discrete particle model) depicts both phases as particles (disperse). The motion of the liquid phase is often described with the SPH (Smoothed Particle Hydrodynamics) [30–32] or MPS (Moving Particle Semi-implicit) [33, 34]. In contrast, the sediment motion may be determined by using a particle-tracking scheme or the DEM (Discrete Element Method) [35–37]. The Lagrangian-Lagrangian models are the least used because they are the

most computationally expensive.

- ◇ **Eulerian-Eulerian models:** They are very frequently adopted in the literature [38–43]. They consider both phases (liquid and sediment) as a continuous phase. The Eulerian-Eulerian approach can be further classified into mixed or separated-fluid approaches [43]. The separated-fluid approach assumes that both the fluid (the carrier) and the particles (the dispersed phase) comprise two separate continua (immiscible phases). Therefore, in the case of a two-phase flow, two sets of momentum equations are required: one for the continuous phase and the other for the dispersed phase. The separated fluid method is also often called the two-fluid method, since two sets of PDEs and two sets of velocity fields are required (one for each phase) [43]. On the other hand, the mixed-fluid approach assumes that the relative velocities, and temperatures between the two phases are small compared to the overall flow variaton. It distinguishes only the mass fractions of the particle and fluid phases in a mixed volume (miscible phases). This results in having a single set of momentum conservation equations for the flow mixture [43]. The mixed-fluid approach allows more simplicity and can generally handle both dispersed and dense conditions.

An overview comprising most of the above-mentioned approaches and methods for hydrodynamics and sediment transport is given in [44]. For the sake of completeness, we want to mention that stochastic approaches in sedimentation engineering are becoming increasingly popular [45–48]. In these models, stochastic partial differential equations, where variables in the deterministic equations are made random with prescribed probability distributions are being used and investigated [45–48] . Stochastic models are particularly relevant when turbulence is a prominent feature of the bedload flow. To account for the random nature of turbulence and sediment movement, researchers developed approaches which describe and quantify an observable state of motion. The bed shear stress for example, can be modeled according to a probability distribution to obtain realistic bedload transport rates at incipient motion, instead of the conventional threshhold criterion such as Shields model [45–48] . Furthermore, turbulence models based on the Reynolds-Averaged Navier-Stokes equations (RANS), are being applied in hydraulic engineering practices and sedimentation dynamics. Other existing turbulence models such as the Detached Eddy Simulation (DES) and the Large Eddy Simulation (LES) are now being applied to sedimentation problems[49–51].

In our model, we rely on an Eulerian-Eulerian mixed-fluid (miscible) approach that we describe further on in the sequel. Thus, we quickly present a hierarchy of models within the Eulerian-Eulerian frame. In a "full" two-fluid model, the liquid and the sediment are treated as two immiscible continuous phases. The models consider the fluid and the sediment as two fluids with distinct properties [52–55]. They use a second liquid field for the dilute sediment phase, with a different momentum equation in addition to that of the liquid field. Models of this kind include two-way coupling, i.e., the momentum exchange between the fluid and the sediment, and vice versa. Another more recent intermediate model that uses variables for the mixture of sediment and water with an algebraic equation for the slip velocity between

the phases is the Partial Two-Fluid Model PTFM [56–58]. This model has extended in [59]. However, it was found that it cannot provide good results, particularly under non-dilute conditions. A complex Two-Fluid Model (CTFM) [56] has been deployed to improve the accuracy in non-dilute conditions but was shown to have large computational time as opposed to PTFM. More recently, the mixture model has been introduced [56]. This model directly deals with the sediment-water mixture. The basic equations of the model include the mass and momentum conservation equations for the sediment-water mixture, and the mass conservation equation for sediment. The velocity of the mixture is solved for a single momentum equation. For each phase, an individual continuity equation is solved to obtain its volume fraction. Various methods have been studied, e.g., in [56–58, 60], to model the relative motion between the fluid and the sediment, but most of the formula proposed are either valid only for the gravity-induced settling or only in limited ranges of concentrations or grain sizes.

Overview of our physical model: The model we propose here is a three-dimensional two-phase (mixture) model with an Eulerian-Eulerian approach. We investigate a miscible model based on a sediment concentration with a single momentum balance for the mixture (sediments and liquid). The present work is, therefore, aimed to establish a general (and not just for non-diluted situations) valid mixture model with the expectation of promoting computation efficiency. We aim to have a broader range of applicability than just gravity-settling situations or fluid-flow induced motion of the sediments, by introducing resuspension effects. Our model is built to have a continuous concentration taking into account dilute and non-dilute situations and that can be extended to various grain sizes (poly-dispersed).

We focus here on sediments in suspension or accumulated in a Newtonian fluid. Moreover, the Newtonian fluid is at contact with air, and we refer to the liquid/air surface with the notion of free surface. From the brief introduction to sediments and sediment dynamics in the subsection above, we retain a few key facts that helped us sample our model:

- Suspended sediment concentration in a stream varies from the water surface to the bottom and laterally across the stream.
- The concentration generally increases from a minimum at the water surface to a maximum at or near the bed (at the bottom).
- The motion of sediments is a result of the flow velocity, the lift, and the settling velocities.
- The grains' characteristics affect their motion.

The concentration of sediments takes values between zero and a maximal concentration, which allows us to describe the different states in which the sediments can be. For instance, the bedload zone corresponds to the region where the concentration is very close to the maximum, i.e., packed sediments. On the other hand, low values of the concentration describe the regions where the mixture sediments/fluid is very dilute. As explained above, many parameters affect the motion of the sediments. These parameters include the fluid flow, the bed slope, as well as

the fluid and the sediment characteristics themselves. In our model, as illustrated in Figure 1.2, we consider that the sediment is subject to three main forces:

- The flow velocity
- The gravity force (deposition)
- The bed shear-induced force (resuspension)

This means that a particle velocity is nothing but a composition of the fluid, the deposition and the resuspension velocities [61].

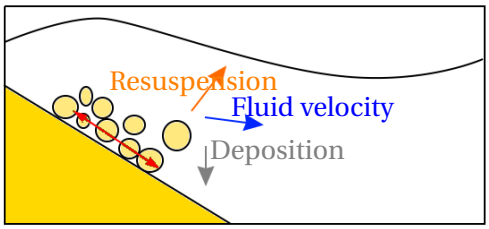


Figure 1.2 – Sediment dynamics in our model: sediments move according to three velocities, the fluid velocity, the deposition velocity and the resuspension velocity.

1.3 Numerical methods for sediment dynamics

The governing equations are transformed into a discretized form corresponding to an appropriate numerical method. Numerical methods can be classified into mesh-based methods (for the Eulerian-Eulerian approaches) and particle methods (for the Lagrangian-Lagrangian and the Eulerian-Lagrangian approaches).

- **Particle methods:**
  For particle methods, a set of particles is used. An overview of some standard particle methods is in [62–64]. Smoothed particle hydrodynamics (SPH) is probably the most popular mesh-free method [65–68]. Since its invention, the SPH method has been extended to a vast range of fluid mechanics applications, where the Navier-Stokes equations need to be solved. The Lagrangian form of the Navier-Stokes equations for a weakly compressible viscous fluid corresponds to a system of ordinary differential equations. In SPH, field variables can be obtained by approximating the governing equations on a set of particles. The particles interact with each other, controlled by a smoothing function, and carry physical properties, such as mass, momentum, density, pressure, and velocity.

More details are provided in [30–32]. SPH methods are very flexible in handling complex fluid phenomena such as splashes, breaking waves and topological changes [69–73]. SPH is also famous for the investigation of wave propagation and wave breaking, e.g., [31, 74]. Because of the Lagrangian nature of SPH, advection is treated exactly, and tracking of the phase interface is simple. Although SPH can be a powerful method for many flow and sediment simulations, improvements are still needed to use SPH, mainly due to the high computational demand. SPH is, in comparison to mesh-based methods, more computationally expensive, especially for large scale simulations. When applied to high concentration problems, a vast number of particles must be tracked. In such a case, parallel programming is required, and supercomputers are needed.

- **Mesh-based methods:**
  In the literature, three main mesh-based methods are used to solve the fluid mechanics PDEs: the finite differences method, the finite volumes method, and the finite element method. These three methods have been applied to solve sedimentation models' governing equations.

  The Finite Differences Method (FDM) [75] has been widely applied to conservation equations. In [59], first and second-order schemes have been used to solve the governing equations for a mixture model. For the discretization of the computational domain, structured grids are required. However, the application of FDM can be difficult when complex geometries are involved.

  Another widely used method for computational sediment dynamics is the Finite Volumes Method (FVM) [76, 77]. The computational domain is subdivided into control volumes (tetrahedrons, cubes, etc), and the governing equations are applied to each control volume. Due to the possibility of having an unstructured grid, the FVM is more suitable than the FDM for representing complex geometries (see [53] as an example of two-phase models solved with FVM techniques).

  Furthermore, the Finite Elements Method (FEM) is commonly used in fluid dynamics, where also unstructured grids and discrete volumes in terms of finite elements are used. The main difference is that the equations are multiplied by a weight function before they are integrated over the domain. However, FEM must be carefully formulated to be conservative [52, 55, 78].

  One of the main challenges of mesh-based methods arises when movable or free boundaries have to be simulated. We remind that a free boundary is a segment of the domain which is moving/changing (unknown) over time. As an example of a free-boundary problem we cite bubble and droplet deformation within a fluid. Both aspects are very common in hydraulic and river engineering. The hydraulic structures have a rather complex surface geometry, and the flow in rivers is characterized by its free surface. Besides, when sediment transport is considered, the interface fluid/sediment becomes a movable/diffuse boundary. Thus, movable boundaries have to be handled appropriately.

Two main approaches for the simulation of movable boundaries are distinguished: surface-tracking and volume-tracking (also called interface capturing).

- For **surface-tracking** methods, the deformation of the interface is explicitly described. Thus, the interface consists of discrete points. This approach is sometimes used for the simulation of Fluid-Structure Interaction (FSI) problems in combination with an adaptive finite-element method [79]. Another approach where the surface points are not connected to the computational grids is the immersed boundary method [80, 81].
- In the **volume-tracking** approaches, the interface is captured based on a scalar marker quantity that changes according to the motion or deformation of the material, e.g., the fluid. A very popular interface capturing method for fluid interfaces of two-phase flows is the Volume-Of-Fluid (VOF) method [82, 83], where the marker quantity describes the volume-fraction of one fluid phase relating to a computational cell. Thus, the interface lies in cells where the volume-fraction is between 0 and 1. An alternative to volume tracking method is the level-set method intorduced by Osher and Sethian [84, 85] and being applied for various physical applications ever since [86–88]. In level-set methods, the marker quantity corresponds to a function describing the distance from the interface. It relies on an implicit formulation of the interface, represented through a time-dependent initial-value partial-differential equation. The zero-level set of this associated level-set function, gives the location of the propagating interface. Another method is the Marker-And-Cell (MAC) method as proposed in [89] and applied in [90], where marker particles are used to identify different phases. Because the advection of the marker particles is computationally expensive, this method is not commonly used. An overview of volume-fraction techniques is given in [91].

These methods are particularly performant in large scale applications. Covering low and high concentration levels is done without exponentially affecting the computational time, which is the case for mesh-free methods.

Current implementations of these methods support steady vs unsteady flows, specific physical properties and can be single or multi-dimensional. A review of one-dimensional, two-dimensional, and three-dimensional models is given in [92]. Before we proceed with describing our numerical method, for the sake of completeness, we give a quick overview of some of the most known software (academic and commercial).

- **Delft3D:**

  Delft3D is an open-source modeling system used for hydrodynamics, sediment transport, and morphology. It has various sub-models for different processes. The (FLOW) module is the core of Delft3D and is a multi-dimensional (2D or 3D) hydrodynamic

simulation program for non-steady flow and transport phenomena [93]. It solves the Navier-Stokes equations for an incompressible fluid, under the shallow water and the Boussinesq assumptions. The sediment transport and morphology (MOR) module supports both bedload and suspended load, transport of non-cohesive sediments, and suspended load of cohesive sediments.
For the "bedload" fraction, the suspended load advection-diffusion equation is not solved. Delft3D is meant to model flow phenomena of which the horizontal length and time scales are significantly larger than the vertical scales [93]. We refer to [94–96] for examples of applications and limitations.

- **Flow3D:** (based on an Eulerian/Eulerian approach with VOF)

  FLOW-3D is a commercial simulation software developed by FlowScience, Inc. for hydrodynamic applications [97]. FLOW-3D's hydrodynamic solver is fully coupled with a sediment transport module that simulates bedload and suspended sediment transport, entrainment, and erosion for non-cohesive soils [97, 98]. It uses the volume of fluid (VOF) method for tracking the free surface. The solid geometry is represented using a cell porosity technique called the FAVOR method. All empirical relationships used in bedload, entrainment, and settling processes are fully customizable, and up to 10 different sediment species (with different properties such as grain size, mass density, and critical shear stress) can be defined. FLOW-3D is ideal for simulating local scour over short episodic time scales. We refer to [99–101] for examples of applications and limitations.

- **OpenFoam:** (based on an Eulerian/Eulerian approach with VOF)

  OpenFOAM is a freely available open-source platform containing several libraries and applications that numerically solve continuum mechanics problems. In OpenFOAM, a three-dimensional two-phase flow solver, SedFoam, is presented for sediment transport applications. The solver is based on the "twoPhaseEulerFoam" module available in the official release of the open-source CFD software OpenFOAM. In this approach, the sediment phase is modeled as a continuum, and constitutive laws have to be prescribed for the sediment stresses [102, 103]. We refer to [104–106] for examples of applications.

- **Fluent:** (based on an Eulerian/Eulerian approach with VOF but offers an Eulerian/Lagrangian approach)

  FLUENT is a Computational Fluid Dynamics (CFD) package from the American company Ansys. In FLUENT, sediment transport is modeled using an Euler-Lagrange approach. The flow of the fluid phase is calculated by solving the discretized Reynolds equations. The discrete phase (the sediment particles) is solved by tracking the particles through the calculated flow field: a Lagrangian approach. It also deals with a free surface for the fluid flow with the volume-of-fluid VOF method [107]. In [108], a comparison was run between Delft3D and FLUENT for several applications. In [109], the same is done for OpenFoam and FLUENT.

More softwares are described and compared in [92]. The choice of a certain model for solving a specific problem depends on the nature and complexity of the problem itself. As of today, most of the existing models have specific scopes of applications and no model has proven to be superior in all applications, yet.

Overview of our numerical method:
Our numerical method is mesh-based. The mixture model that we propose here couples the Navier-Stokes equations, with a volume-of-fluid (VOF) approach for the tracking of the free surfaces between the water and air, plus a nonlinear advection equation for the sediments dynamics. A mathematical model for the simulation of Newtonian fluids with free surfaces and without sediments has been presented and validated in [110–113]. The addition of sediments has a direct effect on the density and viscosity of the flow, resulting in a modified set of Navier-Stokes equations of the following form

$$\rho_m(f_s)\Big(\frac{\partial \mathbf{v}}{\partial t} + \big(\mathbf{v}\cdot\nabla\big)\mathbf{v}\Big) - 2\nabla\cdot\Big(\mu_m(f_s)\mathbf{D}(\mathbf{v})\Big) + \alpha_m(f_s)\mathbf{v} + \nabla p = \rho_m(f_s)\mathbf{g},$$
$$\nabla\cdot\mathbf{v} = 0,$$

where f_s is the solid fraction used to track the sediments in a continuous (diffusive) manner and ρ_m and μ_m are respectively the density and the viscosity of the liquid and the sediments mixture. This equation is penalized with a Darcy-like reaction coefficient α_m, modeling the porous solid matrix [114, 115]. These terms will be explicitely detailed through the dissertation.

The volume fraction of liquid φ is introduced to track the fluid volume. This is done by defining a characteristic function φ that takes zero values outside the liquid, and one inside, as for the VOF method. The volume conservation leads to the advection of this characteristic function, which is given by the equation below

$$\frac{\partial \varphi}{\partial t} + \mathbf{v}\cdot\nabla\varphi = 0,$$

where $\mathbf{v}$ is the velocity of the fluid. Apart from the modifications that apply to Navier-Stokes equations, the additional sedimentation equation reads as follows:

$$\frac{\partial f_s}{\partial t} + \mathbf{v}\cdot\nabla f_s + \nabla\cdot\big(F_d\left(f_s\right) + F_r\left(f_s\right)\big) = 0, \tag{1.1}$$

where $F_d\left(f_s\right)$ and $F_r\left(f_s\right)$ are respectively deposition and resuspension fluxes, describing sediment dynamics. The deposition part consists of a convex flux allowing the deposition of sediments along the gravity vector. The resuspension part is a degenerate-flux allowing the resuspension to occur on the direction of the gradient of f_s . The details will follow in the dedicated chapters.

The resolution of the whole set of equations relies on a splitting algorithm and a two-grid method. The time-splitting algorithm is used to decouple different physical phenomena, precisely the diffusion, and the advection. The two-grid method consists of two meshes: a

coarser non-structured mesh (finite elements) and a finer structured mesh. By introducing the additional finer structured grid, we aim at improving the accuracy of the approximation of the free-surface (between the air and the liquid). The finite element grid remains a good option to solve the Navier-Stokes equations while the Cartesian structured grid is used for fluid tracking. We do not reconstruct any interface between the sediment and the water since we allow a continuous distribution of the sediment. Since Eulerian methods are numerically diffusive, to precisely track and reconstruct the interface (the free surface) we rely on a Simple Line Interface Calculation (SLIC) algorithm first introduced in [116]. It consists in reconstructing the interface with axis-aligned segments (in 2D) or rectangles (in 3D). The VOF solver was introduced in [117] for fixed structured grids and has successfully been used to simulate bubbles with surface tension computation [111] and viscoelastic fluid flows [110] when coupled with the mass and momentum equations. In [118], a dynamic adaptive meshing was used to introduce an octree-scheme with the same VOF approach. To summarize, finite elements are used to solve diffusion problems (that is, the diffusion part of Navier-Stokes equation), whereas characteristics methods adapted to advection problems are used to solve transport parts on the Cartesian grid. Interpolation operators between the two grids are put into place.

We wanted to build a computationally and physically efficient sedimentation model that merges with the existing set of equations. As a result, the novelty of our approach does not only come from the uniqueness of the chosen physical model, as much as from the choice of a dedicated numerical method proposed to solve this multiphysics model coupled with free surfaces.

So far, the advocated splitting algorithm efficiently decouples the various physical phenomena. Following the same approach, we split the sedimentation equation into a transport part and a deposition/resuspension part. This allows us to address each equation with dedicated techniques: finite elements are used for the Stokes equation and the characteristics method for transport equations. In addition, for the sedimentation part, we propose various approaches. The first approach is to solve the remaining part $\frac{\partial f_s}{\partial t} + \nabla \cdot \left(F_d\left(f_s\right) + F_r\left(f_s\right)\right) = 0$ with a finite volumes method. The second approach is to split the remaining part into resuspension and deposition operators. This allows us to use a Riemann solver with Godunov scheme [76] for the deposition part, and to use a higher order finite difference scheme to solve the resuspension part. The choice of the adequate approach is always a trade-off between accuracy and efficiency. More details will be given in the following chapters. This model has proven to be good enough for a range of applications. Specifically, it works very well for experiments with deposition, flushing, and scouring effects. However, it lacks precision when it comes to experiments with a lot of turbulence.

1.4 Thesis outline

The thesis is structured as follows: In **Chapter 2**, we describe the physical and mathematical model that will be used for sedimentation dynamics. In essence, we present a macroscopic

model for the transport of sediment based on a sediment concentration with a single momentum balance for the mixture (fluid and sediments). We introduce the full set of equations that we use to describe such a physical problem.
In **Chapter 3**, we explain the numerical methods used to solve the equations introduced in the first chapter. We rely on a splitting approach [117, 119] to decouple diffusion and advection phenomena such that we are left with a Stokes operator, an advection operator, and deposition/resuspension operators. For the space discretization, a two-grid method couples a finite element discretization for the resolution of the Stokes problem, and a finer structured grid of small cells for the discretization of the advection operator and the sediment deposition/resuspension operator.
In **Chapter 4**, we present the numerical results that we used to validate the chosen physical model as well as the numerical approach. The validation is done in two steps: In the first part, we validate and benchmark the model with deposition effects only for some specific experiments. In the second part, we validate and benchmark the model when we introduce resuspension effects. At the end of Chapter 3, we discuss the limitations of the underlying physical model.
Finally, in **Chapter 5**, we consider a one-dimensional diffusion-convection equation and study a posteriori error estimate for the design of adaptive algorithms. First, we consider a finite element backward scheme, and then, a splitting scheme that separates the diffusion and the convection parts of the equation.

2 A mathematical model for sediment dynamics with free surface flows

In this chapter, we introduce a macroscopic model for the transport of sediment based on a sediment fraction with a single momentum balance for the mixture (fluid and sediments). A mathematical model for the simulation of Newtonian fluids with free surfaces, without sediment transport, has been presented and validated in [110–113], and has been applied to hydraulic engineering situations in [120]. It is extended here to include sediment transport along with resuspension effects, thus extending the method described in [121].

The model proposed here couples the Navier-Stokes equations, with a volume-of-fluid approach for the tracking of the free surfaces between liquid and air, plus a nonlinear advection equation for the sediment migration within the fluid. Since both dilute and undilute sediment concentrations in the liquid need to be described, a model able to describe not only the two phases but also the migration of the sediments and the resulting density variations is chosen. This requires a *miscible* model as opposed to an immiscible multiphase flow model [122]. We consider a mono-dispersed model (a model with a single particle population) for the miscible sediment in the liquid.

2.1 Mathematical model

Let $\Lambda \in \mathbb{R}^3$ be the bounded computational domain containing the fluid (the mixture of the liquid and sediments) and the ambient air and let $T > 0$ be the final time of the simulation. For any given time $t \in (0, T)$, let $\Omega_t \subset \Lambda$ be the domain occupied by the fluid so that the remaining part of the domain Λ is occupied by the ambient air (which is assumed to not influence the fluid).

Let Q_T denote the space-time domain containing the liquid, that is $Q_T = \{(\mathbf{x}, t) : \mathbf{x} \in \Omega_t, 0 < t < T\}$. Let Γ_T be the free surface between the liquid and the air and defined by $\Gamma_T = \{(x, t) \in \partial Q_T \setminus \big(\partial\Lambda \times (0, T)\big)\}$.

First let us introduce the set of **unknowns**:

- We use a volume-of-fluid method (VOF) [91, 117] to track the fluid volume. This is achieved by introducing a characteristic function of the liquid $\varphi : \Lambda \times (0,T) \to \{0,1\}$, which implies that $Q_T = \{(\mathbf{x},t) \in \Lambda \times (0,T) : \varphi(\mathbf{x},t) = 1\}$ and $\Omega_t = \{\mathbf{x} \in \Lambda : \varphi(\mathbf{x},t) = 1\}$.
- The velocity field $\mathbf{v} : Q_T \to \mathbb{R}^3$ and the pressure field $p : Q_T \to \mathbb{R}$.
- The sediment concentration is defined in the liquid domain as $f_s : Q_T \to [0, f_{s_{CR}}]$ where $f_{s_{CR}}$ is the maximal sediment solid fraction. In fact, the presence rate of the sediment in the liquid domain is denoted by the solid fractions f_s. The sediment fraction is a percentage of solid sediment in a given volume. The total amount of sediment is actually limited by a critical maximum value $f_{s_{CR}} < 1$ (i.e the solid fraction of the packed sediments). This critical value is essentially related to grains' size and shape. For example, if we consider a single population of non-moist spherical fine sand particles and without consolidation, this value is approximately equal to 0.61. Moreover, in our model, the sediment particles are spherical, and only exist in the liquid (and not in the air).

A sample sketch of the domain and the unknowns, in two space dimensions, is illustrated in Figure 2.1. The set of corresponding equations is detailed here below.

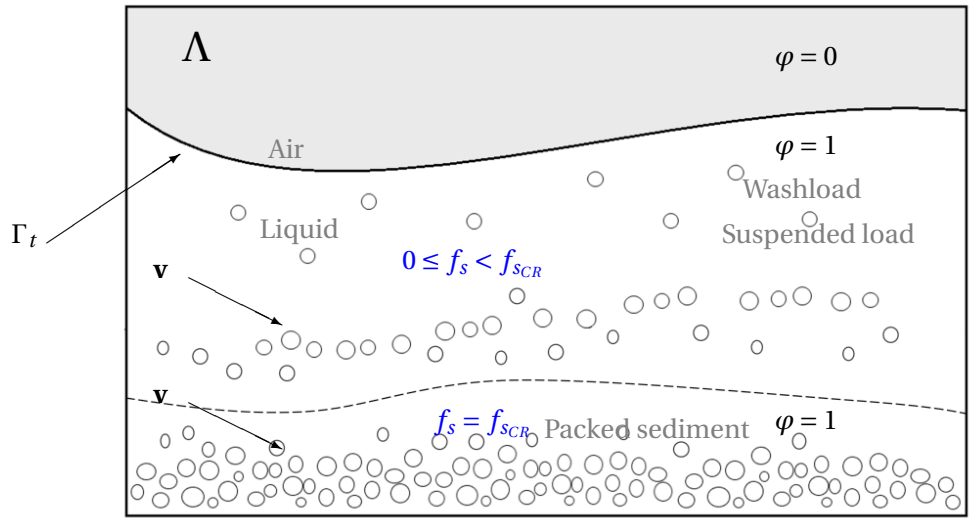


Figure 2.1 – 2D sketch of the computational domain for sedimentation. The cavity Λ contains the air and the liquid. The liquid domain is separated from the ambient air by the liquid-air interface Γ_t, the liquid domain is described by its characteristic function φ, while the solid fraction of the sediment f_s ranges between zero and a critical value $f_{s_{CR}}$.

2.1.1 The volume-of-fluid equation

In order to describe the kinematics of the free surface, $\varphi : \Lambda \times (0,T) \to \mathbb{R}$ must satisfy (in a weak sense):

$$\frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi = 0 \qquad \text{in } \Lambda \times (0,T). \tag{2.1}$$

This equation corresponds to the fact that the fluid particles move at velocity $\mathbf{v}$, more precisely, $\varphi(\mathbf{X}(t),t) = \varphi(\mathbf{X}(0),0)$, where $\mathbf{X}(t)$ is the trajectory of a fluid particle starting from $\mathbf{X}(0)$ at time $t = 0$, thus $\mathbf{X}'(t) = \mathbf{v}(\mathbf{X}(t),t)$.

The characteristic function of the liquid domain φ is given at initial time, which is equivalent to defining the initial liquid region $\Omega_0 = \{\mathbf{x} \in \Lambda : \varphi(\mathbf{x},0) = 1\}$. Boundary conditions are given for φ on the inlet part of $\partial\Omega_t$ (the part of the boundaries with an inflow, if any: $\{\mathbf{x} \in \partial\Omega_t \cap \partial\Lambda; \mathbf{v}\cdot\mathbf{n} < 0\}$).

2.1.2 Navier-Stokes equations with sediment dynamics

We assume that the liquid mixture velocity and pressure $\mathbf{v} : Q_T \to \mathbb{R}^3$ and $p : Q_T \to \mathbb{R}$ satisfy:

$$\rho_m(f_s)\Big(\frac{\partial \mathbf{v}}{\partial t} + \big(\mathbf{v}\cdot\nabla\big)\mathbf{v}\Big) - 2\nabla\cdot\Big(\mu_m(f_s)\mathbf{D}(\mathbf{v})\Big) + \alpha_m(f_s)\mathbf{v} + \nabla p = \rho_m(f_s)\mathbf{g} \quad \text{in } Q_T, \tag{2.2}$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } Q_T. \tag{2.3}$$

Here $\mathbf{D}(\mathbf{v}) = 1/2(\nabla\mathbf{v} + \nabla\mathbf{v}^T)$ is the symmetric deformation tensor, $\mathbf{g}$ denotes the gravity field. The density ρ_m is given by a linear weighted combination of the individual densities ρ_l and ρ_s

$$\rho_m(f_s) = \rho_l(1 - f_s) + \rho_s f_s, \tag{2.4}$$

where ρ_l (resp. ρ_s) is the fluid density (resp. the density of sediment). The viscosity μ_m , represents the apparent viscosity of the fluid with the suspended particles. It is modeled with the so-called Ishii and Zuber law for particle flows [123]:

$$\mu_m(f_s) = \begin{cases} \mu_l\Big(1 - \dfrac{f_s}{f_{s_{CR}}}\Big)^{-2.5 f_{s_{CR}}}, & \text{if } f_s < f_{s_{CO}}, \\ \mu_l\Big(1 - \dfrac{f_{s_{CO}}}{f_{s_{CR}}}\Big)^{-2.5 f_{s_{CR}}}, & \text{otherwise}, \end{cases} \tag{2.5}$$

where μ_l is the Newtonian dynamic viscosity of the fluid and $f_{s_{CO}}$ is a cohesion threshold parameter to be calibrated. In fact, when f_s is large ($f_s \to f_{s_{CR}}$), μ_m is large ($\mu_m \to \infty$). The choice of (2.5) is validated in the literature for small values of the sediment fraction f_s [123]. Moreover, the velocity in the Navier-Stokes equations is penalized with a Brinkman term using Carman-Kozeny empirical law, which represents the coupling with Darcy flow in porous media

[114, 115]. The reaction coefficient $\alpha_m = \alpha_m(f_s)$ in (2.2) is given by:

$$\alpha_m(f_s) = K\frac{\mu_l f_s^2}{d_*^2(f_{s_{CR}} - f_s)^3 + \varepsilon}, \tag{2.6}$$

where $K = 1$ in practice, and ε is a constant to be calibrated and d_* is the mean sediment particle diameter.

Note that, practically, the parameter ε is small and avoids a division by zero when $f_s = f_{s_{CR}}$ in (2.6). The Navier-Stokes equations (2.2)-(2.3) are completed with initial and boundary conditions. The initial conditions for the velocity are

$$\mathbf{v}(0) = \mathbf{v}_0 \qquad \text{in} \quad \Omega_0,$$

For the boundary of the liquid domain that is in contact with the boundary of the cavity $\partial\Lambda$, various options for the boundary conditions are possible. We can impose slip or no-slip boundary conditions on the cavity walls. The no-slip condition is the homogeneous Dirichlet condition that imposes all three velocity components to be zero. The slip condition imposes zero normal velocity and tangential stress on the boundary. Moreover, we can impose inflow and outflow boundary conditions. The inflow condition translates into a non-homogeneous Dirichlet condition while the inflow can either be a non-homogeneous Dirichlet, or a free outflow condition, which translates into a homogeneous Neumann condition (zero normal derivatives of the velocity on the boundary).

Surface tension effects on the liquid-gas interface are not taken into account, and the ambient air is supposed to not influence the liquid and is treated as a vacuum. The boundary conditions on the liquid-gas interface are thus given by the no-force boundary condition:

$$-p\mathbf{n}_\Gamma + 2\mu_m(f_s)\mathbf{D}(\mathbf{v})\mathbf{n}_\Gamma = 0 \qquad \text{on } \Gamma_t, \tag{2.7}$$

with $\mathbf{n}_\Gamma$ the external normal vector to Γ_t.

2.1.3 The sedimentation equation

The sediment fraction $f_s : Q_T \to \mathbb{R}$ satisfies:

$$\frac{\partial f_s}{\partial t} + \mathbf{v}\cdot\nabla f_s + \nabla\cdot\left(F_d\left(f_s\right) + F_r\left(f_s\right)\right) = 0, \tag{2.8}$$

where $F_d\left(f_s\right)$ is a deposition flux and $F_r\left(f_s\right)$ is a resuspension flux.

The deposition flux

Various multiphase models for the deposition of sediments flux exist in the literature. Here, we consider a parabolic model given by:

$$F_d\left(f_s\right) = f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)\Big(k v_{Stokes}\frac{\mathbf{g}}{\|\mathbf{g}\|}\Big), \tag{2.9}$$

where k is a positive parameter independent from f_s and v_s is the maximal sediment velocity given by the Stokes' law:

$$v_{Stokes} = \frac{{d_*}^2\|\mathbf{g}\|(\rho_s - \rho_l)}{18\mu_l},$$

We remind that d_* is the mean sediment particle diameter. The flux (2.9) vanishes when $f_s = 0$ (no sediments) and when $f_s = f_{s_{CR}}$, (packed sediments), see, e.g., [124, 125]. With such a flux, the maximum principle (i.e $f_s \in [0, f_{s_{CR}}]$) holds [126]. Moreover, we illustrate later the benefit of this (convex) parabolic flux for our numerical method [76]. Other models for settling fluxes can be found in, e.g., [127, 128].

The resuspension flux

The resuspension of particles is taken into account through the resuspension flux given by

$$F_r\left(f_s\right) = -f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)A(f_s,\mathbf{v})\nabla f_s, \tag{2.10}$$

where based on [129], $A(f_s,\mathbf{v})$ is given by:

$$A(f_s,\mathbf{v}) = K_r\Big(\frac{\tau(f_s,\mathbf{v}) - \tau_{CR}}{\tau_{CR}}\Big)_+,$$

where K_r is the resuspension constant $[m^2/s]$, $(v)_+ = \max(v,0)$ and $\tau(f_s,\mathbf{v})$ $[\mathrm{N/m^2}]$ is given by the tangential part of the stress tensor:

$$\tau(f_s,\mathbf{v}) = 2\mu_m(f_s)\|\mathbf{D}(\mathbf{v})\mathbf{n}(f_s) - (\mathbf{D}(\mathbf{v})\mathbf{n}(f_s)\cdot\mathbf{n}(f_s))\mathbf{n}(f_s)\|, \tag{2.11}$$

where $\mathbf{n}(f_s) = \dfrac{\nabla f_s}{\|\nabla f_s\|}$ for $\nabla f_s \neq 0$, $\tau(f_s,\mathbf{v}) = 0$ otherwise. Given (2.10), note that equation (2.8) is degenerate parabolic. In the expression of the resuspension flux, three key assumptions are made:

- The resuspension does not occur if the sediment is packed ($f_s = f_{s_{CR}}$), or if there is no sediment ($f_s = 0$). This is assured by virtue of $f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)$
- The shear induced motion of sediments happens in the opposite direction of ∇f_s.
- The resuspension occurs only when there is enough shear on the sediment bedload i.e.

when $\tau(f_s, \mathbf{v})$ is above some critical shear value τ_{CR}. This is assured in the expression of $A(f_s, \mathbf{v})$.

The sediment particles are then allowed to move as a viscous fluid [61]. Practically, typical critical shear values vary in a range 0.02 to 0.5 [130]. Our model supports two configurations of the critical shear:

- **Constant critical shear**: in this case we consider a constant critical shear value $\tau_{CR} \in [0.02, 0.5]$ (see [130]).
- **Shields shear**: in this case, based on [131, 132], we define the critical shear as:

$$\tau_{CR}(f_s, \mathbf{v}) = \theta(f_s, \mathbf{v}) \left(\rho_s - \rho_l\right) \|\mathbf{g}\| d_*,$$

where $\theta(f_s, \mathbf{v})$ is given by:

$$\theta(f_s, \mathbf{v}) = \begin{cases} 0.010595 \ln(Re_*(f_s, \mathbf{v})) + \frac{0.110476}{Re_*} + 0.0027197 & \text{for} \quad Re_* \leq 500, \\ 0.068 & \text{for} \quad Re_* > 500. \end{cases}$$

The shear Reynolds number $Re_*(f_s, \mathbf{v})$ [133, 134] is given by:

$$Re_*(f_s, \mathbf{v}) = \frac{u_*(f_s, \mathbf{v}) d_*}{\nu_l},$$

where ν_l is the kinematic viscosity of the fluid and

$$u_*(f_s, \mathbf{v}) = \sqrt{\frac{\tau(f_s, \mathbf{v})}{\rho_l}},$$

is the shear velocity.

The use of a constant shear proved to be sufficient in most applications. It is important to note that, when $K_r = 0$, the sedimentation model given by equation (2.8) reduces to the deposition model described in [121].

3 A numerical method for sediment dynamics with free surface flows

In the previous chapter, a mathematical model for the simulation of a free surface flow with sediment dynamics has been presented. In this chapter, we present the numerical methods that we used to solve the equations (2.1)–(2.3) and (2.8).

In [117], an operator splitting approach [119] was proposed to solve (2.1)–(2.3), we use the same approach here to addionally incorporate (2.8). The splitting scheme decouples the equations as follows:

- Step 1: modified Stokes operator (2.2) without the advection terms and (2.3),
- Step 2: deposition/resuspension operator in (2.8),
- Step 3: advection operator, transport terms in (2.1), (2.2) and (2.8).

For the space discretization, two grids are used: a finite element unstructured mesh for the resolution of the modified Stokes problem, and a finer structured grid of small cells for the discretization of the advection operator and of the sediment deposition/resuspension operator. Thus, we solve the modified Stokes problem using finite element techniques, a characteristics method is then used to approximate the transport problems on the structured grid. Various approaches are presented to solve the deposition/resuspension problem namely finite volumes, finite differences and finite elements. Moreover, the interpolations between the two grids and other post-processing algorithms will also be detailed in this chapter.

3.1 Time discretization : operator splitting

In order to solve equations (2.1), (2.2), (2.3) and (2.8), a splitting scheme is used. This scheme has been successfully used in [110–112, 117, 120, 135, 136]. without sediment dynamics but in other multiphysics problems; it allows us to decouple the advection from diffusion, deposition

and resuspension.

Let $0 = t^0 < t^1 < t^2 < \ldots < t^N = T$ be a subdivision of the time interval $[0,T]$ and $\Delta t^n = t^{n+1} - t^n$, $n = 0,1,2,\ldots,N-1$ be the time step. Let $\varphi^n : \Lambda \to \mathbb{R}$ and $\Omega^n = \{\mathbf{x} \in \Lambda : \varphi^n(\mathbf{x}) = 1\}$ be the approximate liquid region at time t^n. Let $\mathbf{v^n} : \Omega^n \to \mathbb{R}^3$, $p^n : \Omega^n \to \mathbb{R}$, $f_s^n : \Omega^n \to \mathbb{R}$ be approximations of φ, $\mathbf{v}$, p, f_s respectively available at time t^n.

The approximations φ^{n+1}, $\mathbf{v^{n+1}}$, p^{n+1}, f_s^{n+1} at time t^{n+1} are computed by means of the following splitting algorithm.

1. **Modified Stokes step:** A discretized time-dependent Stokes problem is solved to obtain a prediction of the velocity $\mathbf{v}^{n+1/2}$ and the pressure p^{n+1} in the liquid domain Ω^n.
2. **Deposition and resuspension step:** The equation (2.8), that describes the vertical deposition (along the gravity field) and the resuspension of the sediments, is solved to predict the solid fraction $f_s^{n+1/2}$ in Ω^n.
3. **Advection step:** The predicted velocity $\mathbf{v}^{n+1/2}$ is used to transport the volume fraction of liquid φ^n, the solid fraction $f_s^{n+1/2}$, and the velocity $\mathbf{v}^{n+1/2}$. From the new volume fraction of liquid φ^{n+1}, the new liquid domain Ω^{n+1} is then obtained so as corrections of the velocity $\mathbf{v}^{n+1}$ and solid fraction f_s^{n+1} in Ω^{n+1}.

These steps are illustrated in Figure 3.1,

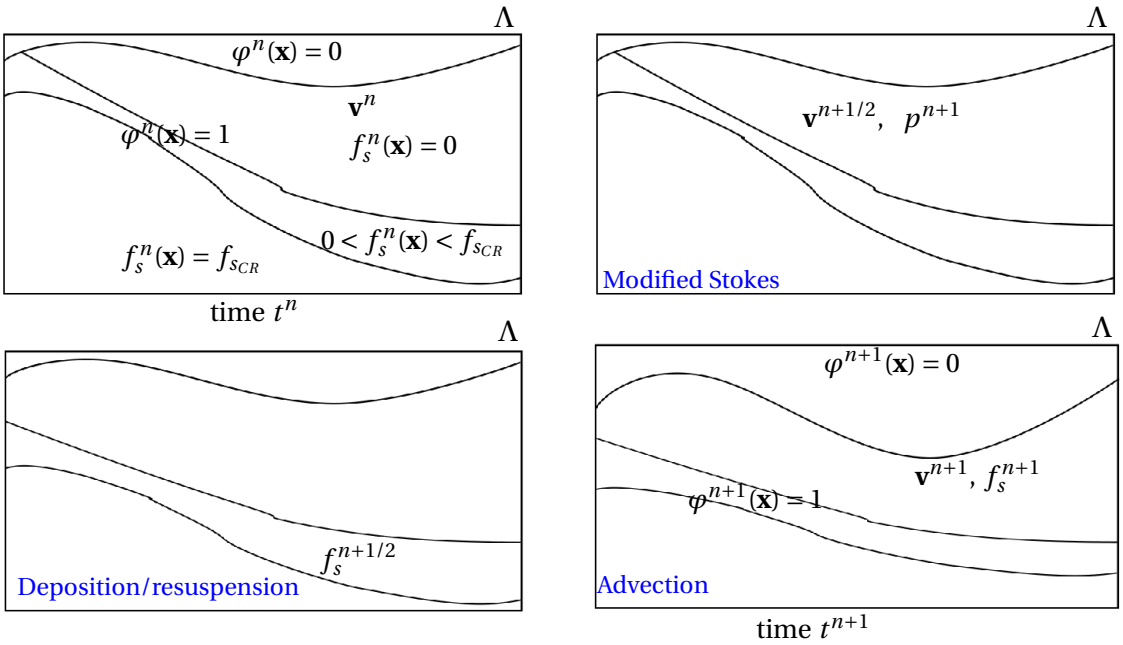


Figure 3.1 – Operator splitting algorithm (from left to right, top to bottom from t^n to t^{n+1}). A modified Stokes problem is solved to obtain the predicted velocity $\mathbf{v}^{n+1/2}$ and the pressure p^{n+1} in the liquid domain $\Omega^n = \{\mathbf{x} \in \Lambda : \varphi^n(\mathbf{x}) = 1\}$. Second, the sediment deposition and resuspension problems are computed to obtain a predicted sediment fraction $f_s^{n+1/2}$ in Ω^n. Finally, advection problems are solved to determine the new approximation of the characteristic function φ^{n+1} (and thus the new liquid domain $\Omega^{n+1} = \{\mathbf{x} \in \Lambda : \varphi^{n+1}(\mathbf{x}) = 1\}$), the corrected velocity $\mathbf{v}^{n+1}$ and solid fraction f_s^{n+1} in Ω^{n+1}.

3.1.1 Modified Stokes step

The approximation f_s^n in Ω^n allows us to define the approximations $\rho_m^n = \rho_m(f_s^n)$, $\mu_m^n = \mu_m(f_s^n)$ and $\alpha_m^n = \alpha_m(f_s^n)$ of the density $\rho_m(f_s)$, viscosity $\mu_m(f_s)$ and reaction coefficient $\alpha_m(f_s^n)$, respectively following (2.4), (2.5) and (2.6). Given $\mathbf{v}(t^n) = \mathbf{v}^n$, we solve in $\Omega^n \times [t^n, t^{n+1}]$

$$\rho_m^n \frac{\partial \mathbf{v}}{\partial t} - 2\nabla \cdot \left(\mu_m^n \mathbf{D}(\mathbf{v})\right) + \alpha_m^n \mathbf{v} + \nabla p = \rho_m^n \mathbf{g}, \tag{3.1}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{3.2}$$

We use zero force condition on the liquid-air interface and no-slip, slip, inflow or outflow conditions on the boundary of the cavity Λ. An implicit Euler scheme can be used

$$\begin{aligned} &\rho_m^n \frac{\mathbf{v}^{n+1/2} - \mathbf{v}^n}{\Delta t^n} - 2\nabla \cdot \left(\mu_m^n \mathbf{D}(\mathbf{v}^{n+1/2})\right) + \nabla p^{n+1} + \alpha_m^n \mathbf{v}^{n+1/2} = \rho_m^n \mathbf{g}, \\ &\nabla \cdot \mathbf{v}^{n+1/2} = 0. \end{aligned} \tag{3.3}$$

3.1.2 Deposition and resuspension step

We solve in $\Omega^n \times [t^n, t^{n+1}]$

$$\frac{\partial f_s}{\partial t} + \nabla \cdot \left(f_s \Big(1 - \frac{f_s}{f_{s_{CR}}}\Big) \Big(k v_{stokes} \frac{\mathbf{g}}{\|\mathbf{g}\|} - A(f_s, \mathbf{v}^{n+1/2}) \nabla f_s \Big) \right) = 0, \tag{3.4}$$

with initial condition $f_s(t^n) = f_s^n$. We denote $f_s^{n+1/2}$ the solution obtained at t^{n+1}.

3.1.3 Advection step

We solve in $\Omega^n \times [t^n, t^{n+1}]$

$$\frac{\partial \varphi}{\partial t} + \mathbf{v}^{n+1/2} \cdot \nabla \varphi = 0, \tag{3.5}$$

$$\frac{\partial f_s}{\partial t} + \mathbf{v}^{n+1/2} \cdot \nabla f_s = 0, \tag{3.6}$$

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}^{n+1/2} \cdot \nabla) \mathbf{v} = 0, \tag{3.7}$$

with initial conditions $\varphi(t^n) = \varphi^n$, $f_s(t^n) = f_s^{n+1/2}$, $\mathbf{v}(t^n) = \mathbf{v}^{n+1/2}$. We set $\varphi^{n+1} = \varphi(t^{n+1})$, $f_s^{n+1} = f_s(t^{n+1})$, $\mathbf{v}^{n+1} = \mathbf{v}(t^{n+1})$

These equations are solved with the characteristics method [137–139]. For all $\mathbf{x} \in \Omega^n$, we have

$$\varphi^{n+1}(\mathbf{x} + \Delta t^n \mathbf{v}^{n+1/2}(\mathbf{x})) = \varphi^n(\mathbf{x}), \tag{3.8}$$

$$f_s^{n+1}(\mathbf{x} + \Delta t^n \mathbf{v}^{n+1/2}(\mathbf{x})) = f_s^{n+1/2}(\mathbf{x}), \tag{3.9}$$

$$\mathbf{v}^{n+1}(\mathbf{x} + \Delta t^n \mathbf{v}^{n+1/2}(\mathbf{x})) = \mathbf{v}^{n+1/2}(\mathbf{x}). \tag{3.10}$$

The new liquid domain Ω^{n+1} is then defined as $\Omega^{n+1} = \{\mathbf{x} \in \Lambda; \varphi^{n+1}(\mathbf{x}) = 1\}$.

3.2 Space discretization

The splitting algorithm allowed diffusion, advection, and deposition/resuspension phenomena to be decoupled. In order to take advantage of this situation, two different prescribed grids are used for space discretization following [111–113, 117]. Since finite element techniques are well suited for solving (3.3) in domains with complex shapes, an unstructured finite element mesh is used. On the other hand, structured grids are well suited for solving the advection problem with the characteristic method. The two grids are illustrated in Figure 3.2 (in two space dimensions): a regular grid of small structured cells (left) is used to solve the

deposition/resuspension problem (3.4) and the advection problems (3.8) (3.9) (3.10) while an unstructured tetrahedral finite element mesh (right), is used to solve the diffusion problem (3.3). The structured grid is always chosen to be finer than the finite element grid, aiming to increase the accuracy of the approximation of the free surfaces (by reducing the numerical diffusion of the approximation φ^{n+1} in (3.8)), while keeping reasonable the computational cost of solving the modified Stokes problem. Adaptive techniques for both grids have been investigated in [112, 140] but are not discussed here.

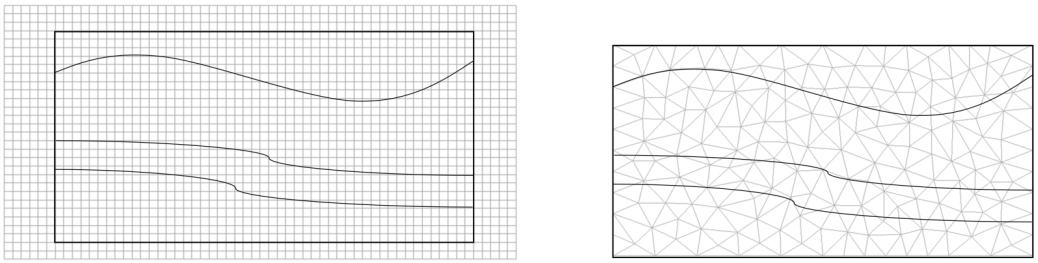


Figure 3.2 – The two grids used (2D sketch): the advection problems and the deposition/resuspension problems are solved on a structured grid of small rectangular cells (left), and the modified Stokes problem is solved on a coarser unstructured finite element mesh (right). The size h of the structured grid and the size H of the mesh satisfy $H \simeq 3h - 5h$.

Let $\mathcal{T}_H$ be a finite element tetrahedral discretization of $\overline{\Lambda}$, with typical size H. The cavity Λ is embedded into a parallelipipedic box discretized into a structured Cartesian grid $\mathcal{C}_h$, which is made out of small cells whose dimensions are denoted by $(\Delta x, \Delta y, \Delta z)$, with a typical size $h := \max\{\Delta x, \Delta y, \Delta z\}$. We label each cell by the indices $(i\,j\,k)$, and denote by $C_{i\,j\,k}$ a generic cell of $\mathcal{C}_h$. Following [117], we typically advocate $H \simeq 3h - 5h$ in the numerical experiments presented in the next chapter. Since the characteristics method is used to solve the advection, CFL numbers larger than one can be used.

3.2.1 Modified Stokes step

Let φ_H^n be the constant piecewise linear approximation of the volume fraction of liquid at time t^n. The liquid region Ω_H^n is defined by the union of all tetrahedra of the finite element mesh $\mathcal{T}_H$ having (at least) one of its vertices P with a value $\varphi_P^n > 0.5$.

Let us define the finite element spaces:

$$V_H^n = \left\{ v_H \in C^0(\overline{\Omega_H^n}) : \ v_H|_K \in \mathbb{P}_1^B, \ \forall K \in \mathcal{T}_H, K \subset \overline{\Omega_H^n} \right\}, \tag{3.11}$$

$$Q_H^n = \left\{ q_H \in C^0(\overline{\Omega_H^n}) : \ q_H\big|_K \in \mathbb{P}_1, \ \forall K \in \mathcal{T}_H, K \subset \overline{\Omega_H^n} \right\}, \tag{3.12}$$

where $\mathbb{P}_1^B$ is the classical space of polynomials of the first degree on K enriched with a bubble function [141].

Let us define ρ_H^n, μ_H^n and α_H^n the piecewise constant approximations of ρ^n, μ^n and α^n on each tetrahedron, respectively. The finite element approximation of (3.3) consists in finding the velocity $\mathbf{v}_H^{n+1/2} \in (V_H^n)^3$, satisfying the essential boundary conditions on $\partial\Omega_H^n$, and the pressure $p_H^{n+1} \in Q_H^n$ such that:

$$\int_{\Omega_H^n} \left(\rho_H^n \frac{\mathbf{v}_H^{n+1/2} - \mathbf{v}_H^n}{\tau^n} \cdot \mathbf{w} + 2\mu_H^n \mathbf{D}(\mathbf{v}_H^{n+1/2}) : \mathbf{D}(\mathbf{w}) + \alpha_H^n \mathbf{v}_H^{n+1/2} \cdot \mathbf{w} \right) d\mathbf{x}$$
$$- \int_{\Omega_H^n} p_H^n \nabla \cdot \mathbf{w} d\mathbf{x} - \int_{\Omega_H^n} q \nabla \cdot \mathbf{v}_H^{n+1/2} d\mathbf{x} = \int_{\Omega_H^n} \rho_H^n \mathbf{g} \cdot \mathbf{w} d\mathbf{x}, \tag{3.13}$$

for all $\mathbf{w} \in (V_H^n)^3$ and $q \in Q_H^n$ the velocity and pressure test functions, compatible with the essential boundary conditions on $\partial\Omega_H^n$. The corresponding linear system is solved with a sequential preconditioned GMRES method of the IML library.

3.2.2 Interpolation on the structured grid

The velocity $\mathbf{v}_H^{n+1/2}$ on Ω_H^n is interpolated at the center of each cell C_{ijk} to obtain values $\mathbf{v}_{ijk}^{n+1/2}$ on the structured grid $\mathcal{C}_h$ (i.e. a piecewise constant approximation). For each liquid cell C_{ijk} (i.e. a cell such that $\varphi_{ijk}^n > 0$), whose center belongs to the element K, the field is interpolated by a *geometric distance-based interpolation* [142]. Denoting by $\mathcal{P}_J$, $J = 1, \ldots, 4$ the vertices of K, by $\mathcal{P}_0$ the barycenter of K, and by $\mathbf{v}_{H,J}^{n+1/2}$ the values of the velocity at point $\mathcal{P}_J$ for $J = 0, \ldots, 4$, the interpolation operator consists in computing a weighted average

$$\mathbf{v}_{ijk}^{n+1/2} = \frac{\sum_{J=0}^{4} \mathbf{v}_{H,J}^{n+1/2} \operatorname{dist}(C_{ijk}, \mathcal{P}_J)^{-2}}{\sum_{J=0}^{4} \operatorname{dist}(C_{ijk}, \mathcal{P}_J)^{-2}}, \tag{3.14}$$

where $\operatorname{dist}(\cdot,\cdot)$ denotes the Euclidean distance. When the cell center coincides exactly with a node P of a tetrahedron K, we set $\mathbf{v}_{ijk}^{n+1/2} = \mathbf{v}_H^{n+1/2}(P)$. The same formula is used to interpolate the volume fraction of fluid and the solid fraction on the structured grid.

3.2.3 Deposition and resuspension step

In order to solve the deposition-resuspension equation (3.4), three methods are proposed:

- **A finite volume method** which consists in solving (3.4) on the grid $\mathcal{C}_h$, using a time-explicit finite volume method, namely the Godunov scheme [76] or the Engquist-Osher

scheme [143].

- **A splitting method with finite volumes and finite differences** which consists in splitting the equation (3.4) into two parts: the deposition part and the resuspension part:

$$\frac{\partial f_s}{\partial t} + \nabla \cdot \left(k f_s \left(1 - \frac{f_s}{f_{s_{CR}}} \right) v_{Stokes} \frac{\mathbf{g}}{\|\mathbf{g}\|} \right) = 0, \tag{3.15}$$

$$\frac{\partial f_s}{\partial t} - \nabla \cdot \left(f_s \left(1 - \frac{f_s}{f_{s_{CR}}} \right) A(f_s, \mathbf{v}^{n+1/2}) \nabla f_s \right) = 0. \tag{3.16}$$

  The model already described in [121] only consisted of the deposition part (3.15). We solve it on the rectangular cells using a finite volumes method. To solve (3.16), we use an explicit finite difference scheme on $\mathcal{C}_h$.

- **A splitting method with finite volumes and finite elements** which corresponds to solving (3.16) using a semi-implicit finite elements scheme on $\mathcal{T}_H$ instead of the finite difference scheme on $\mathcal{C}_h$.

In the numerical results chapter, we show results using all three methods. But we rely on the second method for the benchmarking phase. Before we explain these methods, it is important to specify the approximation of the boundaries of the computational domain on the structured grid $\mathcal{C}_h$, especially if the computational domain is not a box.

In a three-dimensional cartesian coordinates (Ox, Oy, Oz), let N_x, N_y and N_z the number of cells in the directions of Ox, Oy and Oz respectively. Moreover, from now on and for all numerical experiments presented in Chapter 3, the gravity $\mathbf{g}$ is assumed to be aligned with Oz. We have already mentioned that the structured grid englobes the computational domain, however, practically, we have to detect the boundaries.

- A cell C_{ijk}, $1 \leq i \leq N_x$, $\quad 1 \leq j \leq N_y$, $\quad 1 \leq k \leq N_z$ is active if its center is inside the computational domain (defined through the finite elements mesh). Only the active cells are involved in the computation. We denote by Ω_h^n the set of active cells at time t^n.

- A cell C_{ijk}, $1 \leq i \leq N_x \quad 1 \leq j \leq N_y \quad 1 \leq k \leq N_z$ is a boundary cell if it is located next to a non-active cell (i.e. if a cell sharing a face is non-active).

The boundary on the structured grid is defined by all the boundary cells. An example is illustrated in Figure 3.3 in two dimensions. The computational domain in this case is cylindrical. The purple cells illustrates the boundaries of the structured grid in this case.

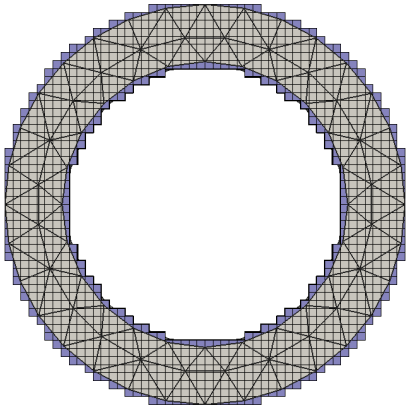


Figure 3.3 – An illustration of boundary definition on the structured grid in two dimensions: The cells intersecting with the finite elements boundary, represented in purple, define the boundary cells of the structured grid.

Assume that the values of $(f_s)^n_{ijk}$ are known for all active cells. We need to compute the approximations $(f_s)^{n+1/2}_{ijk}$ from (3.4). In order to compute approximations of $A(f_s,\mathbf{v})$ on the structured grid cells, we first compute τ^n_{ijk} the approximation of the tangential stress defined by (2.11) at cell ijk at time t^n:

$$\tau^n_{ijk} = 2\mu_m((f_s)^n_{ijk})\|\mathbf{D}^{n+1/2}_{ijk}\mathbf{n}^n_{ijk} - (\mathbf{D}^{n+1/2}_{ijk}\mathbf{n}^n_{ijk}.\mathbf{n}^n_{ijk})\mathbf{n}^n_{ijk}\|, \tag{3.17}$$

where $\mathbf{D}^{n+1/2}_{ijk}$ and $\mathbf{n}^n_{ijk}$ are approximations of $\mathbf{D}(\mathbf{v})$ and $\mathbf{n}(f_s)$ computed using centered finite differences. The second order tensor $\mathbf{D}^{n+1/2}_{ijk}$ at cell ijk in the matricial form is denoted by:

$$\mathbf{D}^{n+1/2}_{ijk} = \begin{pmatrix} (\mathbf{D}_{xx})^{n+1/2}_{ijk} & (\mathbf{D}_{xy})^{n+1/2}_{ijk} & (\mathbf{D}_{xz})^{n+1/2}_{ijk} \\ (\mathbf{D}_{yx})^{n+1/2}_{ijk} & (\mathbf{D}_{yy})^{n+1/2}_{ijk} & (\mathbf{D}_{yz})^{n+1/2}_{ijk} \\ (\mathbf{D}_{zx})^{n+1/2}_{ijk} & (\mathbf{D}_{zy})^{n+1/2}_{ijk} & (\mathbf{D}_{zz})^{n+1/2}_{ijk} \end{pmatrix},$$

The velocity vector (resp. the normal vector) at cell ijk is defined by its three components $(\mathbf{v}_x)^{n+1/2}_{i+1jk}$, $(\mathbf{v}_y)^{n+1/2}_{i+1jk}$ and $(\mathbf{v}_z)^{n+1/2}_{i+1jk}$ at time t^n (resp. $(\mathbf{n}_x)^n_{ijk}$, $(\mathbf{n}_y)^n_{ijk}$ and $(\mathbf{n}_z)^n_{ijk}$ at time t^n) in directions Ox, Oy and Oz, respectively.

For example, the component $(\mathbf{D}_{xx})^{n+1/2}_{ijk}$ is given by

$$(\mathbf{D}_{xx})^{n+1/2}_{ijk} = \frac{(\mathbf{v}_x)^{n+1/2}_{i+1jk} - (\mathbf{v}_x)^{n+1/2}_{i-1jk}}{2\Delta x},$$

and the component $(\mathbf{n}_x)^n_{ijk}$ is given by

$$(\mathbf{n}_x)^n_{ijk} = \frac{\dfrac{(f_s)^n_{i+1jk} - (f_s)^n_{i-1jk}}{2\Delta x}}{\sqrt{\left(\dfrac{(f_s)^n_{i+1jk} - (f_s)^n_{i-1jk}}{2\Delta x}\right)^2 + \left(\dfrac{(f_s)^n_{ij+1k} - (f_s)^n_{ij-1k}}{2\Delta y}\right)^2 + \left(\dfrac{(f_s)^n_{ijk+1} - (f_s)^n_{ijk-1}}{2\Delta z}\right)^2}}.$$

Similar definitions are used to compute the other components. On the boundaries, instead of using the centered differences, we use left-sided or right-sided second order finite difference schemes [144], depending on the boundary side. For example, for $i = 0$, we use

$$\left(\mathbf{D}^{n+1/2}_{0jk}\right)_{xx} = \frac{-3\left(\mathbf{v}_x\right)^{n+1/2}_{0jk} + 4\left(\mathbf{v}_x\right)^{n+1/2}_{1jk} - \left(\mathbf{v}_x\right)^{n+1/2}_{2jk}}{2\Delta x},$$

and

$$\left(\mathbf{n}^n_{0jk}\right)_x = \frac{\dfrac{-3(f_s)^n_{0jk} + 4(f_s)^n_{1jk} - (f_s)^n_{2jk}}{2\Delta x}}{\sqrt{\left(\dfrac{-3(f_s)^n_{0jk} + 4(f_s)^n_{1jk} - (f_s)^n_{2jk}}{2\Delta x}\right)^2 + \left(\dfrac{(f_s)^n_{0j+1k} - (f_s)^n_{0j-1k}}{2\Delta y}\right)^2 + \left(\dfrac{(f_s)^n_{0jk+1} - (f_s)^n_{0jk-1}}{2\Delta z}\right)^2}}.$$

The finite volume method

We first split (3.4) in directions Ox, Oy and Oz

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial x}\left(-f_s\left(1 - \frac{f_s}{f_{s_{CR}}}\right)\left(A(f_s, \mathbf{v}^{n+1/2})\frac{\partial f_s}{\partial x}\right)\right) = 0,$$

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial y}\left(-f_s\left(1 - \frac{f_s}{f_{s_{CR}}}\right)\left(A(f_s, \mathbf{v}^{n+1/2})\frac{\partial f_s}{\partial y}\right)\right) = 0,$$

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(-f_s\left(1 - \frac{f_s}{f_{s_{CR}}}\right)\left(k v_{Stokes} + A(f_s, \mathbf{v}^{n+1/2})\frac{\partial f_s}{\partial z}\right)\right) = 0.$$

Each of these equations can be written as an instance of a generic one-dimensional nonlinear conservation law. We use Godunov type schemes [76] to solve each equation:

$$(f_s)^{n+\frac{1}{6}}_{ijk} = (f_s)^n_{ijk} - \frac{\Delta t^n}{\Delta x}\left((FV_X)^n_{i+\frac{1}{2}jk} - (FV_X)^n_{i-\frac{1}{2}jk}\right), \tag{3.18}$$

$$(f_s)^{n+\frac{1}{3}}_{ijk} = (f_s)^{n+\frac{1}{6}}_{ijk} - \frac{\Delta t^n}{\Delta y}\left((FV_Y)^{n+\frac{1}{6}}_{ij+\frac{1}{2}k} - (FV_Y)^{n+\frac{1}{6}}_{ij-\frac{1}{2}k}\right), \tag{3.19}$$

$$(f_s)_{ijk}^{n+\frac{1}{2}} = (f_s)_{ijk}^{n+\frac{1}{3}} - \frac{\Delta t^n}{\Delta z}\Big((FV_Z)_{ijk+\frac{1}{2}}^{n+\frac{1}{3}} - (FV_Z)_{ijk-\frac{1}{2}}^{n+\frac{1}{3}}\Big). \tag{3.20}$$

The numerical fluxes are defined by:

$$(FV_X)_{i+\frac{1}{2}jk}^{n} = \begin{cases} \max\limits_{(f_s)_{ijk}^n \le f_s \le (f_s)_{i+1jk}^n} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(A_{i+\frac{1}{2}jk}^n {G_X}_{i+\frac{1}{2}jk}^n\Big)\right] & \text{if } (f_s)_{ijk}^n \le (f_s)_{i+1jk}^n \\ \min\limits_{(f_s)_{i+1jk}^n \le f_s \le (f_s)_{ijk}^n} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(A_{i+\frac{1}{2}jk}^n {G_X}_{i+\frac{1}{2}jk}^n\Big)\right] & \text{if } (f_s)_{i+1jk}^n < (f_s)_{ijk}^n \end{cases} \tag{3.21}$$

$$(FV_Y)_{ij+\frac{1}{2}k}^{n+\frac{1}{6}} = \begin{cases} \max\limits_{(f_s)_{ijk}^{n+\frac{1}{6}} \le f_s \le (f_s)_{ij+1k}^{n+\frac{1}{6}}} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(A_{ij+\frac{1}{2}k}^n {G_Y}_{ij+\frac{1}{2}k}^{n+\frac{1}{6}}\Big)\right] & \text{if } (f_s)_{ijk}^{n+\frac{1}{6}} \le (f_s)_{ij+1k}^{n+\frac{1}{6}} \\ \min\limits_{(f_s)_{ij+1k}^{n+\frac{1}{6}} \le f_s \le (f_s)_{ijk}^{n+\frac{1}{6}}} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(A_{ij+\frac{1}{2}k}^n {G_Y}_{ij+\frac{1}{2}k}^{n+\frac{1}{6}}\Big)\right] & \text{if } (f_s)_{ij+1k}^{n+\frac{1}{6}} < (f_s)_{ijk}^{n+\frac{1}{6}} \end{cases} \tag{3.22}$$

$$(FV_Z)_{ijk+\frac{1}{2}}^{n+\frac{1}{3}} = \begin{cases} \max\limits_{(f_s)_{ijk}^{n+\frac{1}{3}} \le f_s \le (f_s)_{ijk+1}^{n+\frac{1}{3}}} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(k\nu_{Stokes} + A_{ijk+\frac{1}{2}}^n {G_Z}_{ijk+\frac{1}{2}}^{n+\frac{1}{3}}\Big)\right] & \text{if } (f_s)_{ijk}^{n+\frac{1}{3}} \le (f_s)_{ijk+1}^{n+\frac{1}{3}} \\ \min\limits_{(f_s)_{ijk+1}^{n+\frac{1}{3}} \le f_s \le (f_s)_{ijk}^{n+\frac{1}{3}}} \left[-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(k\nu_{Stokes} + A_{ijk+\frac{1}{2}}^n {G_Z}_{ijk+\frac{1}{2}}^{n+\frac{1}{3}}\Big)\right] & \text{if } (f_s)_{ijk+1}^{n+\frac{1}{3}} < (f_s)_{ijk}^{n+\frac{1}{3}} \end{cases} \tag{3.23}$$

where,

$${G_X}_{i+\frac{1}{2}jk}^{n} = \frac{(f_s)_{i+1jk}^n - (f_s)_{ijk}^n}{\Delta x},$$

$${G_Y}_{ij+\frac{1}{2}k}^{n+\frac{1}{6}} = \frac{(f_s)_{ij+1k}^{n+\frac{1}{6}} - (f_s)_{ijk}^{n+\frac{1}{6}}}{\Delta y},$$

$${G_Z}_{ijk+\frac{1}{2}}^{n+\frac{1}{3}} = \frac{(f_s)_{ijk+1}^{n+\frac{1}{3}} - (f_s)_{ijk}^{n+\frac{1}{3}}}{\Delta z},$$

and $A_{ijk}^n = K_r\left(\dfrac{\tau_{ijk}^n - \tau_{CR}}{\tau_{CR}}\right)_+$.

Moreover, zero-flux conditions are imposed on the boundaries. For example, if we consider the vertical direction, the fluxes at the top and at the bottom of each column are set to zero. This means $(FV_Z)_{ij\frac{1}{2}} = (FV_Z)_{ijN_z+\frac{1}{2}} = 0$. We apply the same approach to compute the boundaries in directions Ox and Oy.

Remark: Even though we relied on the Godunov scheme [76] to solve the deposition-resuspension conservation law in most experiments, we implemented another conservative scheme, namely the Engquist-Osher scheme [143], capable of solving the same equations. For

instance, the numerical schemes in equations (3.18)–(4.11) are in this case replaced by:

$$(FV_X)^n_{i+\frac{1}{2}jk} = \frac{1}{2}\left[F_X((f_s)^n_{i+1jk}) + F_X((f_s)^n_{ijk}) - \int_{(f_s)^n_{ijk}}^{(f_s)^n_{i+1jk}} \left|\frac{d}{df_s}\left(-f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(A^n_{i+\frac{1}{2}jk}G_X{}^n_{i+\frac{1}{2}jk}\right)\right)\right| df_s\right], \tag{3.24}$$

$$(FV_Y)^n_{ij+\frac{1}{2}k} = \frac{1}{2}\left[F_Y((f_s)^n_{ij+1k}) + F_Y((f_s)^n_{ijk}) - \int_{(f_s)^n_{ijk}}^{(f_s)^n_{ij+1k}} \left|\frac{d}{df_s}\left(-f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(A^n_{ij+\frac{1}{2}k}G_Y{}^n_{ij+\frac{1}{2}k}\right)\right)\right| df_s\right], \tag{3.25}$$

$$(FV_Z)^n_{ijk+\frac{1}{2}} = \frac{1}{2}\left[F_Z((f_s)^n_{ijk+1}) + F_Z((f_s)^n_{ijk}) - \int_{(f_s)^n_{ijk}}^{(f_s)^n_{ijk+1}} \left|\frac{d}{df_s}\left(-f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(kv_{Stokes} + A^n_{ijk+\frac{1}{2}}G_Z{}^n_{ijk+\frac{1}{2}}\right)\right)\right| df_s\right], \tag{3.26}$$

instead of (3.21), (3.22) and (3.23) respectively. In fact, in (3.24) (resp. in (3.25) and (3.26)), in order to evaluate the integrals, we do not need to explicitly compute the derivatives but we rather need to evaluate their signs. For example, if we look at equation (3.24), we can evaluate the sign of

$$\frac{d}{df_s}\left(-f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(A^n_{i+\frac{1}{2}jk}G_X{}^n_{i+\frac{1}{2}jk}\right)\right),$$

by looking at the convexity of

$$-f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(A^n_{i+\frac{1}{2}jk}G_X{}^n_{i+\frac{1}{2}jk}\right),$$

which is mainly determined by the sign of $A^n_{i+\frac{1}{2}jk}G_X{}^n_{i+\frac{1}{2}jk}$. Details are given in [143].

A comparison is done in one-dimensional cases between both schemes and results are shown in the numerical results section.

The splitting method with finite differences for the resuspension

First, we solve (3.15) as in [121] to obtain $f_s^{n+1/4}$, using finite volumes scheme:

$$(f_s)^{n+\frac{1}{4}}_{ijk} = (f_s)^n_{ijk} - \frac{\Delta t^n}{\Delta z}\left((FV_Z)^n_{ijk+\frac{1}{2}} - (FV_Z)^n_{ijk-\frac{1}{2}}\right), \tag{3.27}$$

where the Godunov flux applied to (3.15) is given by

$$(FV_Z)^n_{ijk+\frac{1}{2}} = \begin{cases} \max\limits_{(f_s)^n_{ijk} \le f_s \le (f_s)^n_{ijk+1}} \left[-f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right) k v_{Stokes} \right] & \text{if } (f_s)^n_{ijk} \le (f_s)^n_{i+1jk}, \\ \min\limits_{(f_s)^n_{ijk+1} \le f \le (f_s)^n_{ijk}} \left[-f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right) k v_{Stokes} \right] & \text{if } (f_s)^n_{ijk+1} < (f_s)^n_{ijk}. \end{cases} \tag{3.28}$$

Second, we solve the equation (3.16) to obtain $f_s^{n+1/2}$ on the structured grid cells using the following explicit finite differences scheme:

$$\frac{(f_s)^{n+\frac{1}{2}}_{ijk} - (f_s)^{n+\frac{1}{4}}_{ijk}}{\Delta t^n} = (FD_X)^{n+\frac{1}{4}}_{ijk} + (FD_Y)^{n+\frac{1}{4}}_{ijk} + (FD_Z)^{n+\frac{1}{4}}_{ijk}. \tag{3.29}$$

For simplification purposes, we denote $F_l(f_s) := f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right)$. The numerical fluxes are given by:

$$(FD_X)^{n+\frac{1}{4}}_{ijk} = \frac{A^{n+\frac{1}{4}}_{i+\frac{1}{2}jk} F^{n+\frac{1}{4}}_{i+\frac{1}{2}jk} - A^{n+\frac{1}{4}}_{i-\frac{1}{2}jk} F^{n+\frac{1}{4}}_{i-\frac{1}{2}jk}}{\Delta x^2},$$

$$(FD_Y)^{n+\frac{1}{4}}_{ijk} = \frac{A^{n+\frac{1}{4}}_{ij+\frac{1}{2}k} F^{n+\frac{1}{4}}_{ij+\frac{1}{2}k} - F^{n+\frac{1}{4}}_{ij-\frac{1}{2}k} A^{n+\frac{1}{4}}_{ij-\frac{1}{2}k}}{\Delta y^2},$$

$$(FD_Z)^{n+\frac{1}{4}}_{ijk} = \frac{A^{n+\frac{1}{4}}_{ijk+\frac{1}{2}} F^{n+\frac{1}{4}}_{ijk+\frac{1}{2}} - A^{n+\frac{1}{4}}_{ijk-\frac{1}{2}} F^{n+\frac{1}{4}}_{ijk-\frac{1}{2}}}{\Delta z^2},$$

where,

$$F^{n+\frac{1}{4}}_{i+\frac{1}{2}jk} = F_l\left(\frac{(f_s)^{n+\frac{1}{4}}_{i+1jk} + (f_s)^{n+\frac{1}{4}}_{ijk}}{2}\right)\left((f_s)^{n+\frac{1}{4}}_{i+1jk} - (f_s)^{n+\frac{1}{4}}_{ijk}\right),$$

$$F^{n+\frac{1}{4}}_{ij+\frac{1}{2}k} = F_l\left(\frac{(f_s)^{n+\frac{1}{4}}_{ij+1k} + (f_s)^{n+\frac{1}{4}}_{ijk}}{2}\right)\left((f_s)^{n+\frac{1}{4}}_{ij+1k} - (f_s)^{n+\frac{1}{4}}_{ijk}\right),$$

$$F^{n+\frac{1}{4}}_{ijk+\frac{1}{2}} = F_l\left(\frac{(f_s)^{n+\frac{1}{4}}_{ijk+1} + (f_s)^{n+\frac{1}{4}}_{ijk}}{2}\right)\left((f_s)^{n+\frac{1}{4}}_{ijk+1} - (f_s)^{n+\frac{1}{4}}_{ijk}\right).$$

Similar definitions are used to compute $F^{n+\frac{1}{4}}_{i+\frac{1}{2}jk}$, $F^{n+\frac{1}{4}}_{ij-\frac{1}{2}k}$ and $F^{n+\frac{1}{4}}_{ijk-\frac{1}{2}}$. Note that $A^{n+\frac{1}{4}}_{ijk} = K_r\left(\dfrac{\tau^{n+\frac{1}{4}}_{ijk} - \tau_{CR}}{\tau_{CR}}\right)_+$ with $\tau^{n+\frac{1}{4}}_{ijk}$ being the approximation of $\tau\left((f_s)^{n+1/4}_{ijk}, \mathbf{v}^{n+1/2}_{ijk}\right)$ computed by

(3.17). We want to implement Neuman conditions on the grid boundaries :

$$f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)A(f_s,\mathbf{v})\nabla f_s\cdot\mathbf{n}=0. \tag{3.30}$$

Let us suppose we want to compute the scheme in a cell at the boundary. Looking back at the scheme (3.29), FD_X, FD_Y and FD_Z represent fluxes in the directions Ox,Oy and Oz respectively. Thus, we can compute the fluxes on the boundaries by considering the normal vector in each direction separately : depending on the flux we want to compute, $\mathbf{n}$ is either in the direction of Ox,Oy or Oz To better understand this, we consider a simplified example, illustrated in Figure 3.4. We denote by i_b the index of the rightmost horizontal position of the cells on the boundary where $\mathbf{n}=\mathbf{e_x}$. On the blue surface, illustrated in Figure 3.4, implementing (3.30) yields

$$(f_s)_{i_b+1jk}=(f_s)_{i_bjk}.$$

for any $0\le j\le N_y$ and $0\le k\le N_z$. We apply the same method in order to compute FD_Y and FD_Z at the boundaries.

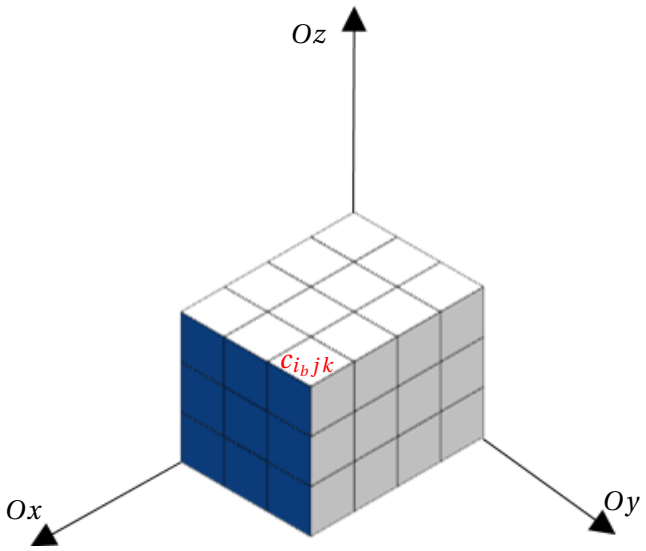


Figure 3.4 – The blue interface represents the boundary where $\mathbf{n}=\mathbf{e_x}$. The cells with a blue interface are positioned at $i=i_b$ and the flux FD_X through this interface is set to zero.

Stability condition: The stability of this explicit scheme is assured under the following condition [145]:

$$\left|\frac{\Delta t^n\max\limits_{i,j,k}\Big(A\big((f_s)_{ijk}^{n+1/4},\mathbf{v}_{ijk}^{n+1/2}\big)\Big)}{h^2}\right|\le\frac{1}{6}, \tag{3.31}$$

where $h=\min(\Delta x,\Delta y,\Delta z)$. This has been verified through various tests.

Time step for the resuspension using finite differences: If $K_r \neq 0$, the resuspension step comes after the deposition step. For every cell (ijk), $i = 1,\dots,N_x, j = 1,\dots,N_y$, $k = 1,\dots,N_z$, we solve the resuspension equation. The stability condition (3.31) for the resuspension equation can be very restrictive. Thus, we don't change the overall time step Δt^n but we implement a sub-stepping approach:

- We compute Δt_r such that it fits the stability criterion. If $\Delta t^n < \Delta t_r$, $\Delta t_r = \Delta t^n$, otherwise, we compute
$$N_r = \lfloor (\Delta t/\Delta t_r \rfloor + 1,$$
$$\Delta t_r = \Delta t^n / N_r,$$
where $\lfloor (\Delta t/\Delta t_r \rfloor$ is the integer portion of $(\Delta t^n/\Delta t_r)$
- We solve the resuspension equation for each substep $i = 1 \dots N_r$.

The splitting method with finite elements for the resuspension

Like in the second method, between t^n and t^{n+1}, we solve the equation (3.15) using finite volumes to predict a solution $f_s^{n+1/4}$. Then we solve (3.16) using continous piecewise linear finite elements. The semi-implicit scheme writes:

$$\frac{f_s^{n+\frac{1}{2}} - f_s^{n+\frac{1}{4}}}{\Delta t} - \nabla \cdot \left(F_l(f_s^{n+\frac{1}{4}}) A(f_s^{n+\frac{1}{4}}, \mathbf{v}^{n+\frac{1}{2}}) \nabla f_s^{n+\frac{1}{2}} \right) = 0, \tag{3.32}$$

Let ${f_s}_H^{n+\frac{1}{4}} \in Q_H$ be a piecewise linear approximation of $f_s^{n+\frac{1}{4}}$ using the finite element space Q_H defined by (3.12). The resuspension problem is solved with $\mathbb{P}_1$ finite elements. It consists in finding the solid fraction ${f_s}_H^{n+\frac{1}{2}} \in Q_H$ satisfying the Neumann conditions and such that:

$$\frac{1}{\Delta t} \int_{\Omega_H} {f_s}_H^{n+\frac{1}{2}} f \, d\mathbf{x} + \int_{\Omega_H} F_{lH}({f_s}_H^{n+\frac{1}{4}}) A_H({f_s}_H^{n+\frac{1}{4}}, \mathbf{v}_H^{n+\frac{1}{2}}) \nabla {f_s}_H^{n+\frac{1}{2}} \cdot \nabla f \, d\mathbf{x} = \frac{1}{\Delta t} \int_{\Omega_H} {f_s}_H^{n+\frac{1}{4}} f \, d\mathbf{x} \tag{3.33}$$

for all $f \in Q_H$ and where A_H and F_{lH} are defined by

$$A_H({f_s}_H^{n+\frac{1}{4}}, \mathbf{v}_H^{n+\frac{1}{2}})\Big|_K = \frac{1}{4} \sum_{J=1}^{4} A({f_s}_H^{n+\frac{1}{4}}(\mathcal{P}_J), \mathbf{v}_H^{n+\frac{1}{2}}(\mathcal{P}_J)),$$

and

$$F_{lH}({f_s}_H^{n+\frac{1}{4}})\Big|_K = \frac{1}{4} \sum_{J=1}^{4} F_l({f_s}_H^{n+\frac{1}{4}}(\mathcal{P}_J)),$$

where $\mathcal{P}_J$, $J = 1,\dots,4$ are the vertices of the element K of the finite element mesh. The sediment fraction ${f_s}_H^{n+\frac{1}{2}}$ is then interpolated on the structured grid $\mathcal{C}_h$, as described in Subsection 3.2.2, to obtain values $(f_s)_{ijk}^{n+1/2}$.

In all cases, the deposition and resuspension step allows us to obtain a prediction $(f_s)_{ijk}^{n+1/2}$ of

the sediment fraction, which is corrected in the advection step below.

3.2.4 Advection step

Equations (3.8)–(3.10) are implemented on the structured grid $\mathcal{C}_h$ as in [117], using constant values within each cell C_{ijk}. The advection step for a cell C_{ijk} consists in advecting all the quantities in the cell along $\Delta t^n \mathbf{v}_{ijk}^{n+1/2}$, which means that the quantities φ_{ijk}^n, $f_{s_{ijk}}^{n+1/2}$ and $\mathbf{v}_{ijk}^{n+1/2}$ are transported from the center $\mathbf{x}_{ijk}$ of C_{ijk} to the point $\mathbf{x}_{ijk} + \Delta t^n \mathbf{v}_{ijk}^{n+1/2}$. However, since the final position of the cell trajectory does not necessarily coincide with a cell's center, we conservatively redistribute the transported quantities into the overlapped cells (proportionally to the volume intersected by the transported cell and the overlapped cells). An example of cell advection and projection is presented in Figure 3.5 in two space dimensions. This method can be interpreted as a forward characteristics method with projection and therefore is unconditionally stable (no Courant-Friedrichs-Lewy (CFL) condition) and convergent.

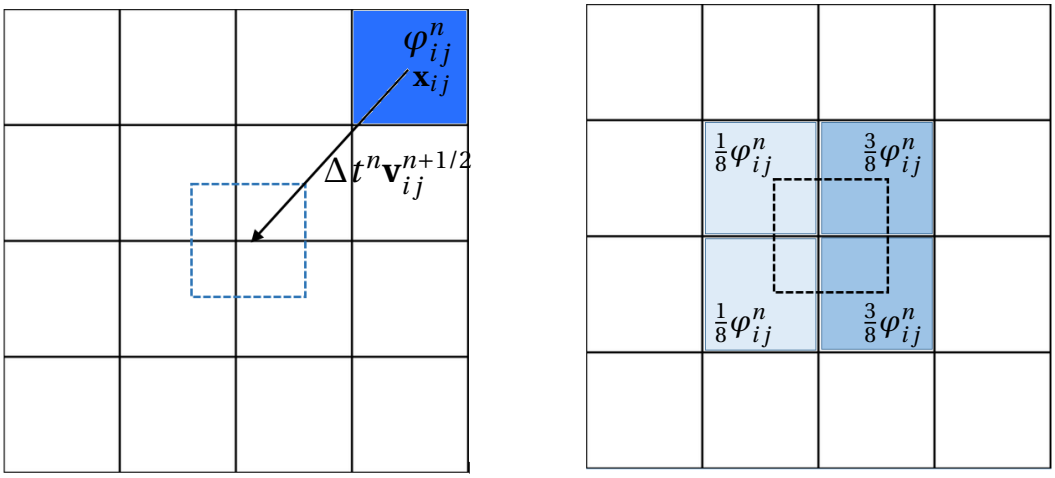


Figure 3.5 – An example of two-dimensional advection of φ_{ij}^n by $\Delta t^n \mathbf{v}_{ij}^{n+1/2}$ (left) and projection on the grid (right). The advected cell is represented by the dashed lines. The four cells containing the advected cell receive a fraction of φ_{ij}^n according to the position of the advected cell (ij).

Since the volume fraction of liquid φ is a step function, numerical diffusion is introduced when the values contained in the advected cells are projected on the grid. In order to reduce the numerical diffusion, we use a variation of the heuristic SLIC (Simple Linear Interface Calculation) algorithm inspired by [116]. Details can be found in [117]. Moreover a post-processing procedure avoids the numerical (artificial) compression induced by two cells

arriving at the same place or the loss of liquid/sediment mass due to cells transported outside the cavity Λ. More details are given in the next subsection.

3.2.5 Redistribution and decompression algorithms

Two situations can occur after the advection:

1. A cell can be advected outside the computational domain. The carried quantities φ and f_s are lost consequently.
2. Two cells can arrive at the same position. If this happens, after the projection, the destination cell can end-up with values $\varphi > 1$ and/or $f_s > f_{s_{CR}}$.

In order to assure the conservation of φ and f_s, we perform two post-processing algorithms:

1. Redistribution of lost quantities, inside the computational domain.
2. Decompression techniques, related to global and local repair algorithms [146], to produce final values φ^{n+1} (resp. f_s^{n+1}) which are between zero and one (resp. zero and $f_{s_{CR}}$).

In order to describe these techniques, we denote that,

$$\varphi = f_s + f_l,$$

where f_l is the liquid fraction. Moreover, we define the following apellations: an inflow surface, is a surface from which sediment and/or fluid can enter the computational domain. An outflow surface, is a sufrace from which sediment and fluid are allowed to exit the domain.

Redistribution algorithm

As already mentioned, the goal of this algorithm is to redistribute the quantities that are lost outside the domain. Lost quantities belong to cells that have been wrongly convected outside the computational domain. We refer to these cells by lost cells in the sequel.

The algorithm is the following: For each lost cell c_{out} we first construct a cube $\zeta_{c_{out}} \subset \Omega_h^{n+1}$ of its neighbouring cells. The neighbouring cells, at the first level, are the twenty-six cells that are adjacent to c_{out} in all three space directions (including corners). The neighbours are then reduced to those admissible (i.e. the neighbours inside the domain, and containing some f_s). We distribute the lost quantities (f_s and φ) on these cells, recursively. If no admissible cells were found, or if there is some f_s left to be distributed, the level of redistribution can be extended to neighbours of neighbours or more. In Figure 3.6, we illustrate the described algorithm in two dimensions.

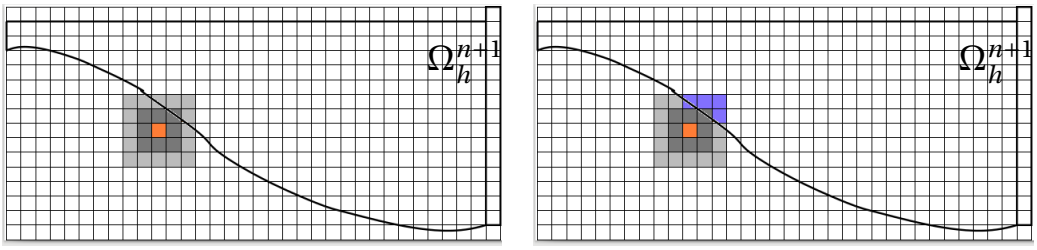


Figure 3.6 – Neighbours distribution in two dimensions. On the left: The algorithm constructs a square of neighbouring cells around the lost cell, marked in orange. The first neighbours are marked in gray, and the neighbours of neighbours are marked in light gray. On the right: The admissible neighbours, marked in purple, receive some quantity from the lost cell.

For instance, in two dimensions, as illustrated in Figure 3.7, in the level two distribution, the neighbouring cells are the eight adjacent cells and the 16 subadjacent cells, making up a total of 34 neighbours. The maxium level of redistribution is defined by a variable $level_m$ and is generally chosen to be 4.

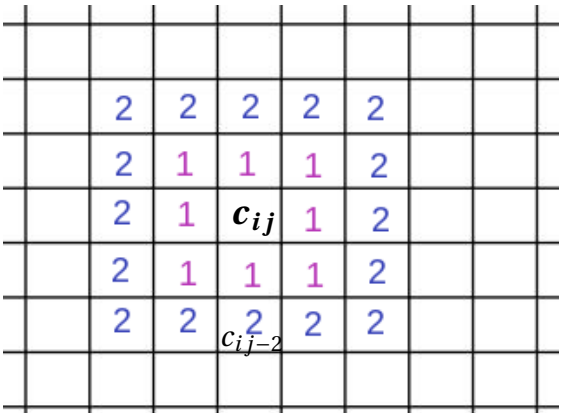


Figure 3.7 – Neighbours distribution in two dimensions around a cell c_{ij}, $0 < i \leq N_x$, $0 < j \leq N_y$. The neighbours of c_{ij} are marked "1", the neighbours of neighbours are of c_{ij} are marked "2".

The steps are also summarized in Algorithm 1 hereafter.

Algorithm 1 Neighbour-based redistribution

$b_s, b_l \leftarrow 0$ { buffer is initially empty}
for all cell c_{out} lost outside **do**
$\varphi_d = f_s^{n+1}(c_{out}) + f_l^{n+1}(c_{out})$ { φ_d is the total lost value to distribute, liquid and sediment }
$f_{s_d} = f_s^{n+1}(c_{out})$ { Save the quantity of sediment to distribute in f_{s_d} }
$f_{l_d} = f_s^{n+1}(c_{out})$ { Save the quantity of liquid to distribute in f_{l_d} }
while $\varphi_d > 0$ and $level < level_m$ **do** { $level$ is the current neighbours level, $level_m$ is a preset maximal distribution level}
$\zeta_{c_{out}} \leftarrow \{c \in \Omega^{n+1} s.t\ c$ is a neighbour of $c_d\}$ {Construct a cube of neighbouring cells}
$S_{c_{out}} \leftarrow \{c \in \zeta_{c_{out}} s.t \quad 0 < f_s^{n+1}(c) < f_{s_{CR}},\}$ {Restrict neighbouring cells to admissible neighbouring cells, sediment-wise}
$L_{c_{out}} \leftarrow \{c \in \zeta_{c_{out}} s.t \quad 0 < f_l^{n+1}(c) < 1,\}$ {Restrict neighbouring cells to admissible neighbouring cells, liquid-wise}
if $f_{s_d} > 0$ **then**
for all neighbour c in $S_{c_{out}}$ **do**
add $\min(f_{s_d}, f_s^{n+1}(c) - f_{s_{CR}})$ to $f_s^{n+1}(c)$
add $\min(f_{s_d}, f_s^{n+1}(c) - f_{s_{CR}})$ to $\varphi^{n+1}(c)$ { If f_s increases, so does φ}
$f_{s_d} \leftarrow f_{s_d} - \min(f_{s_d}, f_s^{n+1}(c) - f_{s_{CR}})$
$\varphi_d \leftarrow \varphi_d - \min(f_{s_d}, f_s^{n+1}(c) - f_{s_{CR}})$
end for
end if
if $f_{l_d} > 0$ **then**
for all neighbour c in $L_{c_{out}}$ **do**
add $\min(f_{l_d}, f_l^{n+1}(c) - 1)$ to $f_l^{n+1}(c)$
add $\min(f_{l_d}, f_l^{n+1}(c) - 1)$ to $\varphi^{n+1}(c)$ { If f_l increases, so does φ}
$f_{l_d} \leftarrow f_{l_d} - \min(f_{l_d}, f_l^{n+1}(c) - 1)$
$\varphi_d \leftarrow \varphi_d - \min(f_{l_d}, f_l^{n+1}(c) - 1)$
end for
end if
$level \leftarrow level + 1$ { Increase distribution level}
end while
$b_s \leftarrow b_s + f_{s_d}$ { Keep non-distributed sediment loss in buffer b_s}
$b_l \leftarrow b_l + f_{l_d}$ { Keep non-distributed liquid loss in buffer b_l}
end for

The algorithm stops when we have either redistributed all the lost quantities or when we have reached the maximal level of redistribution. The latter case happens when the CFL number is very large, or in experiments where the boundaries are curved and/or not very regular. If this happens, we store the remaining quantities in a buffer to deal with after we perform the decompression algorithms, in order to ensure the conservation.

Decompression algorithms

Examples of heuristic decompression algorithms used to accomplish this task for φ^{n+1} , are given in [118] and [117]. They consist of choosing a priority function $\theta : \Omega_h^{n+1} \to \mathbb{R}$ then redistributing the total excess of fluid to the cells in the domain, in the decreasing order of their θ values. In Algorithm 2, we formally describe this process.

Algorithm 2 Decompression algorithm with priority function θ

Initialization: $b \leftarrow 0$ {b is the buffer }
for all cell c in Ω_h^{n+1} **do**
  if $\varphi^{n+1}(c) > 1$ **then**
    $b \leftarrow b + (\varphi^{n+1}(c) - 1)V(c)$ { V is the volume of the cell c}
    $\varphi^{n+1}(c) \leftarrow 1$
  end if
end for
$A_c \leftarrow \{c \in \Omega_h^{n+1} s.t \quad 0 < \varphi^{n+1}(c) < 1\}$ {Admissible receiver cells}
while do
  choose c in A_c which maximizes $\theta(c)$
  remove c from A_c
  $s \leftarrow 1 - \varphi^{n+1}(c)$
  $\varphi^{n+1}(c) \leftarrow \min(1, \varphi^{n+1}(c) + \frac{b}{V(c)})$
  $b \leftarrow \max(0, b - sV(c))$
end while

In [117], θ is chosen equal to φ^{n+1}. This means redistributing the fluid first to cells with highest φ^{n+1} values. In [118], θ is chosen to be a smoothed version of φ^{n+1}. In other words, the excess fluid is distributed on the cells which have the highest φ^{n+1} values and whose neighbours also have high values.

In our case, we have an additional quantity f_s to decompress. Our decompression algorithm has thus, two main steps, executed in the following order:

1. The decompression of the solid fraction f_s, for which we use a different heuristic algorithm, described below.
2. The decompression of φ, for which we use the same algorithm as in [117].

We present here two heuristic variations of decompression algorithms used to decompress f_s.

The first variation is the one we use as default. It has proven to be very efficient in most cases. However, an additional (problem-dependent) variation (variation 2) has been developed. This variation is used when we have a lot of symmetry for an experiment, or if we have a considerable mass loss due to curved boundaries, provided the gravity follows the vertical direction (See example in Subsection 4.2.6). It provides a faster and less random effect of the distribution.

- **Variation 1:** For every cell c_d such that f_s is above the maximal value $f_{s_{CR}}$, we construct a cube $\zeta_c \subset \Omega_h^{n+1}$ of its neighbouring cells. This is done the same way as in the redistribution part. The difference is that we now randomize these neighbours. Then, we restrict the randomized neighbours to the admissible ones (neighbours that have $f_s < f_{s_{CR}}$). We finally distribute the excess value from the cell c into its neighbours. If we are unable to distribute all excess in current neighbours, we can increase recursively the level of distribution to two (to include neighbours of neighbours) or more. The higher the level of distribution is, the higher the chances are we distribute all excess. Algorithm 3 is a pseudo-code of the process.

Algorithm 3 Neighbour-based decompression

for all cell c_d in Ω_h^{n+1} **do**
  if $f_s^{n+1}(c_d) > f_{s_{CR}}$ **then**
    $d \leftarrow (f_s^{n+1}(c_d) - f_{s_{CR}})$ { d is the excess value from the cell c}
    $f_s^{n+1}(c_d) \leftarrow f_{s_{CR}}$
    $\varphi^{n+1}(c_d) \leftarrow \varphi^{n+1}(c_d) - d$ { Since $\varphi = f_s + f_l$, the decompression of f_s help the decompression of φ.}
  end if
  while $d > 0$ and $level < level_m$ **do** { l is the current distrubution level, l_m is a preset maximal distribution level}
    $\zeta_{c_d} \leftarrow \{c \in \Omega^{n+1} s.t\ c$ is a neighbour of $c_d\}$ {Construct a cube of neighbouring cells}
    $A_{c_d} \leftarrow \{c \in \zeta_{c_d} s.t \quad 0 < f_s^{n+1}(c) < f_{s_{CR}},\}$ {Restrict neighbouring cells to admissible neighbouring cells}
    for all neighbour c in A_{c_d} and $d > 0$ **do**
      add $\min(d, f_s^{n+1}(c) - f_{s_{CR}})$ to $f_s^{n+1}(c)$
      $d \leftarrow d - \min(d, f_s^{n+1}(c) - f_{s_{CR}})$
    end for
    $level \leftarrow level + 1$ { Increase distribution level}
  end while
  $b_s \leftarrow b_s + d$ { Keep non-distributed excess sediment in buffer b_s}
end for

- **Variation 2:** In this variation of the decompression of f_s, we do not rely on the neighbours distribution. Instead, we proceed with the following: The first step is to collect the excess sediment quantity over all active cells in a buffer b_s. Then, in a three-dimensional setup, where the gravity is along the vertical axis, we consider that the cells at position $z = 0$ are prioritary for the distribution. Thus, we start by distributing the quantities to

these cells homogeneously. This means that all cells should receive the same quantity of liquid/sediment. This is why we name this process the Homogeneous distribution. We present here a pseudo-algorithm for the distribution of the sediment in Algorithm 4.

Algorithm 4 Homogeneous distribution

{b_s contains the sediment quantity to be distributed }
while $z < N_z$ and $b_s > 0$ **do** {z is the vertical position of a cell c }
  Find the minimal solid fraction available m_{f_s} in level z
  if (m_{f_s} is 0 {No cell is available to receive quantity } **then**
    $z \leftarrow z+1$ {Go to next level }
  else
    Compute uniform value to distribute $u_{f_s} \leftarrow \frac{b_s}{n_a}$ {n_a is the number of available cells}
    $d_{f_s} = \min(m_{f_s}, u_{f_s})$ {d_{f_s} is the quantity to distribute}
    for all cell c in level z **do**
      $f_s \leftarrow f_s + d_{f_s}$
      $b_s \leftarrow b_s - d_{f_s}$
    end for
  end if
end while

We emphasize the fact that, when looking for the minimum f_s available, we consider that the available quantity is given by $f_{s_{CR}} - f_s$. This garantees that $f_s \in [0, f_{s_{CR}}]$. When applying the same algorithm for the liquid distribution, the available quantity is $1-\varphi$.

After the decompression of sediments, we proceed with the decompression of the liquid using Algorithm 2. As already mentioned earlier, under very specific conditions (large CFL, non-regular domains) the total decompression and/or redistribution are not guaranteed. We proceed with additional **conservation algorithms**: In this case, the initial sediment buffer b_s contains the sediment remainders of the redistribution and the decompression parts. The remaining quantities are distributed homogeneously on all admissible cells, using Algorithm 4. The same is done to distribute the remaning liquid quantity (if any), where the initial liquid buffer b_l contains the liquid remainders of the redistribution and the decompression parts.

3.2.6 Projection on the finite elements triangulation

Once the values φ_{ijk}^{n+1}, $f_{s_{ijk}}^{n+1}$ and $\mathbf{v}_{ijk}^{n+1}$ have been computed on $\mathcal{C}_h$, the approximated fields are interpolated at the vertices (and also at the barycenter when projecting the velocity) of each tetrahedra K of the mesh $\mathcal{T}_H$, in order to restart the next time step with the Stokes problem. The volume fraction of liquid at any vertex $\mathcal{P}$ is computed by considering all the cells whose

barycenter is in the tetrahedra that are adjacent with $\mathcal{P}$:

$$\varphi_H^{n+1}(\mathcal{P}) = \frac{\sum\limits_{\substack{K\in\mathcal{T}_H\\ \mathcal{P}\in K}} \sum\limits_{C_{ijk}\in K} \varphi_{ijk}^{n+1}\cdot \mathrm{dist}(C_{ijk},\mathcal{P})^{-2}}{\sum\limits_{\substack{K\in\mathcal{T}_H\\ \mathcal{P}\in K}} \sum\limits_{C_{ijk}\in K} \mathrm{dist}(C_{ijk},\mathcal{P})^{-2}}. \tag{3.34}$$

Figure 3.8 illustrates, in the two-dimensional case, the cells to be considered when projecting on node $\mathcal{P}$. If the point considered is the barycenter of a tetrahedron K, the formula is modified to account only for the element K and not the adjacent tetrahedra.

When the values of φ_H^{n+1} are available at the vertices of $\mathcal{T}_H$, the liquid region Ω_H^{n+1} is defined as follows: an element of the finite element mesh $\mathcal{T}_H$ is said to be *liquid* if (at least) one of its vertices P has a value $\varphi_P^{n+1} > 0.5$. The computational domain Ω_H^{n+1} is defined as the union of all liquid elements.

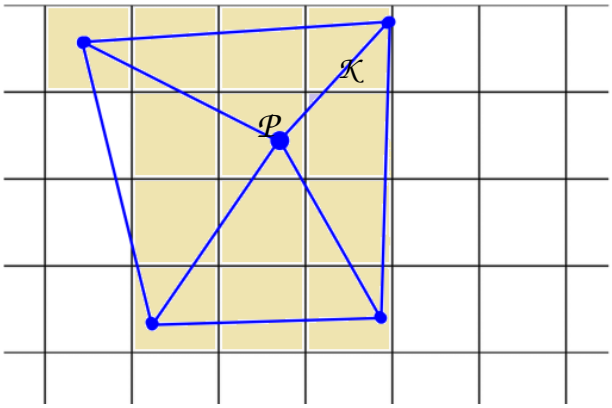


Figure 3.8 – All tetrahedra (here triangles) in $\mathcal{T}_H$ containing the vertex $\mathcal{P}$ are shown. The cells whose barycenter is intersecting with these tetrahedra, shown in light yellow, are involved in the interpolation.

The approximations $\rho_H^{n+1}, \mu_H^{n+1}$ and α_H^{n+1} are computed on the finite element mesh in order to solve (3.13). More precisely, those quantities are first computed on the structured grid $\mathcal{C}_h$ following (2.4) (2.5) and (2.6), based on the piecewise constant approximations of the sediment fraction that are obtained on the grid of cells after the advection step. These values are then interpolated on the finite element mesh to obtain piecewise constant approximations of the physical properties. For instance, the approximation of the density is given by:

$$\rho_H^{n+1}\big|_K = \frac{\sum\limits_{C_{ijk}\in K} \rho_{ijk}^{n+1}\cdot \mathrm{dist}(C_{ijk},\mathcal{B})^{-2}}{\sum\limits_{C_{ijk}\in K} \mathrm{dist}(C_{ijk},\mathcal{B})^{-2}}, \tag{3.35}$$

where $\mathcal{B}$ is the barycentric center of element K. Similar relationships apply for μ_H^{n+1} and α_H^{n+1}, to obtain piecewise constant approximations of those quantities on the finite element mesh $\mathcal{T}_H$.

3.2.7 Additional remarks and global algorithm flowchart

Computation of the inflow/outflow mass rates

In order to verify the mass conservation for the sediments, an algorithm has been implemented in order to compute the mass of the liquid and the sediment flowing in and out of the computational domain as well as the mass staying inside the domain.

Before we introduce the algorithms, we formally give the physical definitions of the volume and the mass flow-rate: The volumic flow rate Q (in $[m^3/s]$) through a surface Γ with velocity $\mathbf{v}$ is given by:

$$Q = \int_\Gamma \mathbf{v}\, d\Gamma,$$

The mass flow-rate Q_m (in $[kg/s]$) is given by:

$$Q_m = \int_\Gamma \rho_s f_s \mathbf{v}\, d\Gamma.$$

Thus, in order to compute the total mass m (in $[kg]$) that flew through a surface S over a period of time $[0, T]$, we integrate Q_m over time:

$$m = \int_0^T Q_m dt.$$

In practice, we have two options in order to compute the flow rates. The first option is to directly use the definitions above, by computing the rates on the finite elements. However, since the advection step is completed on the structured grid cells, we choose to compute the rates on the grid. This allows us to avoid unnecessary interpolation errors. The computation of the inflow and outflow mass rate is implemented along the advection part. This means that for each time step t^n of the global algorithm, $t^0 = 0 \leq t^n \leq t^N = T$, after the advection step, we compute the convected mass (the mass carried by the convected cells).

We denote $m_{in}(t^n)$ as the sediment mass flowing into the domain between t^n and $t^n + \Delta t$, $m_{out}(t^n)$ as the sediment mass flowing out of the domain between t^n and $t^n + \Delta t$ and m_{t^n} as the sediment mass inside the domain at time t^n, $n = 0 \dots N$. We have some balance if:

$$\sum_{n=0}^{N} m_{in}(t^n) + m_{t^0} = \sum_{n=0}^{N} m_{out}(t^n) + m_{t^N} \tag{3.36}$$

Apart from checking the conservation, we can use (3.36) for other applications such as measur-

ing the mass of sediments that settle down through the computational domain over a period of time.

Global Algorithm flowchart:

The code for free-surfaces flow with sediment dynamics has been created based on the notions and algorithms described before. The steps of computation are summarized in the algorithm flow chart illustrated in Figure 3.9.

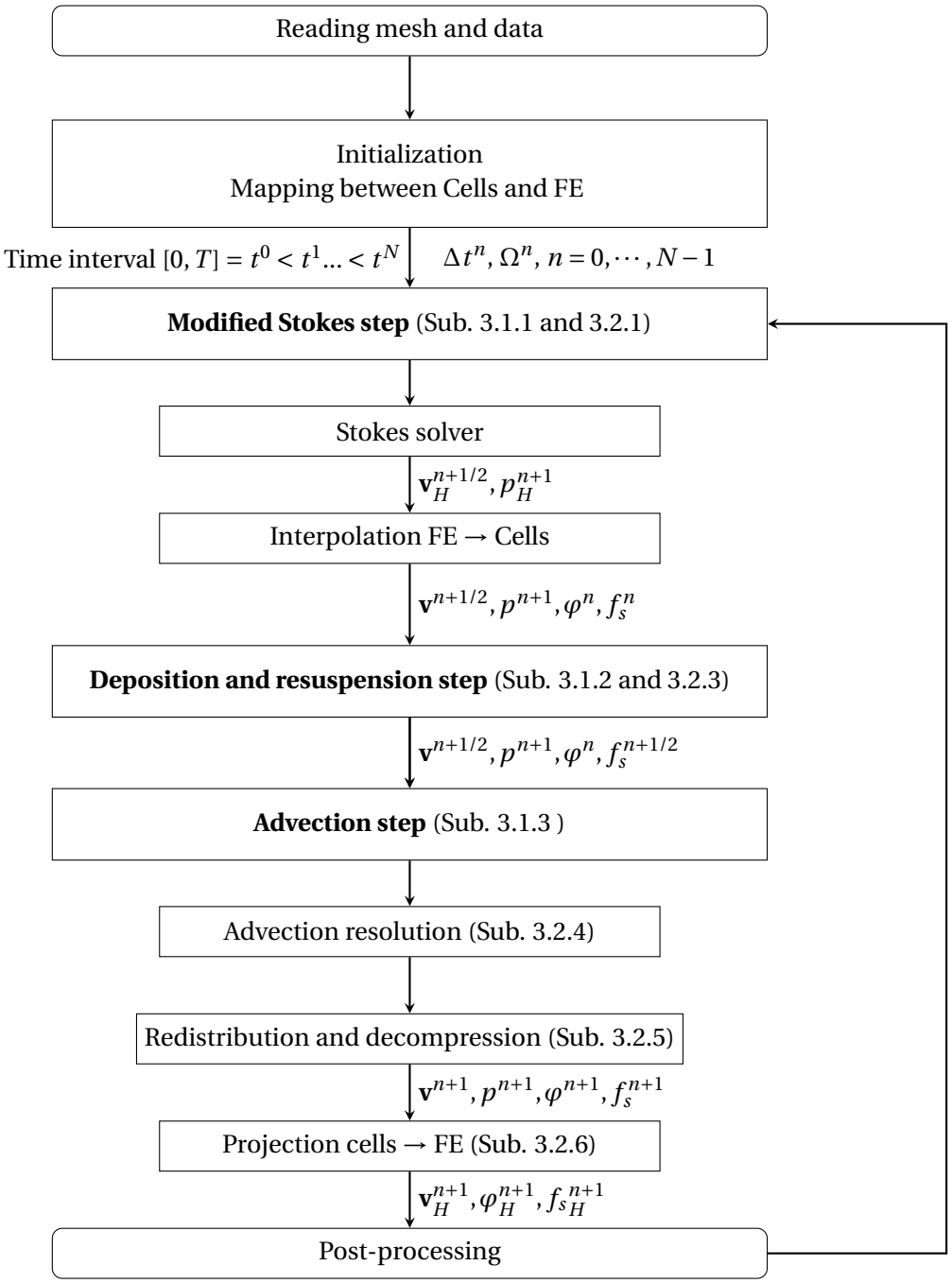


Figure 3.9 – Flow chart for the resolution steps

4 Numerical results

In the previous chapters, we have presented the mathematical model for sediment dynamics, along with the numerical methods to solve it. In this chapter, we present numerical results in order to validate the numerical approach as well as the physical model by:

- Performing convergence studies.
- Performing sensitivity analyses of the model parameters.
- Benchmarking the computational results with real experimental results.

This is done in two steps: First, in section 4.1, we validate and benchmark the model with deposition effects only. In fact, we consider processes with pure sedimentation or with erosion and impinging jets, where resuspension effects are not essential. Most of Section 4.1 results have already been published in [121]. Second, in Section 4.2, we validate and benchmark the full model (with deposition and resuspension effects). Finally, we discuss the limitations of the underlying physical model. We recall that the sedimentation equation is

$$\frac{\partial f_s}{\partial t} + \mathbf{v}\cdot\nabla f_s + \nabla\cdot\left(-f_s\Big(1-\frac{f_s}{f_{s_{CR}}}\Big)\Big(k\nu_{Stokes}\frac{\mathbf{g}}{\|\mathbf{g}\|} + K_r\Big(\frac{\tau(f_s,\mathbf{v})-\tau_{CR}}{\tau_{CR}}\Big)_+\nabla f_s\Big)\right) = 0, \tag{4.1}$$

and choosing $K_r = 0$ cancels the resuspension effects.

4.1 Results of sedimentation with deposition only

The steps of this part are as follows:

1. Validation of the Finite volume scheme

2. Validation of the deposition model with analytical and experimental data

4.1.1 Engquist-Osher and Godunov schemes: a comparison in the one-dimensional case

The main goal of this part is to validate the finite volumes implemented schemes for the deposition equation: the Godunov scheme and the Engquist-Osher scheme. Moreover, the Godunov being known as exact [76] when the conservation equation flux is convex (such is the case of the deposition flux), we wanted to see how the Engquist-Osher scheme compares to it. We consider the parabolic flux (2.9), and the corresponding analytical prototypical (and one dimensional) problem for $f_s \in [0, f_{s_{CR}}]$,

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(\alpha f_s\left(f_s - f_{s_{CR}}\right)\right) = 0, \qquad \text{in} \quad \mathbb{R}, \tag{4.2}$$

with $\alpha = k v_{Stokes}/f_{s_{CR}}$, and with a discontinuous initial solution that reads (Riemann problem):

$$f_s(t=0) = \begin{cases} f_s+, & z \geq z_0 \\ f_s-, & z_0 \geq z \geq z^- \\ f_{s_{CR}}, & z^- \geq z. \end{cases}$$

We consider $f_s+ = 0$, $f_s- = 0.48$, $f_{s_{CR}} = 0.6$, $z_0 = 0.055$ and $z^- = 0$. The notations defined in previous chapters remain valid. We denote by T the final time. Let us denote $F_D(f_s) = k v_{Stokes} f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right)$ as the deposition flux function.

The finite volumes scheme used to solve (4.2) writes as follows:

$$(f_s)_i^{n+1} = (f_s)_i^n - \frac{\Delta t^n}{\Delta z}\left((FV_Z)_{i+\frac{1}{2}}^n - (FV_Z)_{i-\frac{1}{2}}^n\right), \quad 1 \leq i \leq N_z \tag{4.3}$$

where (FV_Z) is the numerical flux given by:
Godunov scheme [76]

$$(FV_Z)_{i+\frac{1}{2}}^n = \begin{cases} \max\limits_{(f_s)_i^n \leq f_s \leq (f_s)_{i+1}^n} \left[-F_D(f_s)\right] & \text{if } (f_s)_{i+1}^n \geq (f_s)_i^n, \quad 1 \leq i \leq N_z \\ \min\limits_{(f_s)_{i+1}^n \leq f \leq (f_s)_i^n} \left[-F_D(f_s)\right] & \text{if } (f_s)_{i+1}^n < (f_s)_i^n. \quad 1 \leq i \leq N_z \end{cases} \tag{4.4}$$

Engquist-Osher scheme [143]

$$(FV_Z)_{i+\frac{1}{2}}^n = \frac{1}{2}\left[F_D((f_s)_{i+1}^n) + F_D((f_s)_i^n) - \int_{(f_s)_i^n}^{(f_s)_{i+1}^n} |F_D'(f_s)| df_s\right]. \tag{4.5}$$

In such a case (Riemann problem), an analytical solution f_{s_e} of the problem (4.2) can be obtained explicitly [76] and at $t = T$ the exact solution is given by:

$$f_s(z, t=T) = \begin{cases} 0.6 & \text{if} (x \leq 0.044), \\ 0 & \text{otherwise.} \end{cases}$$

where $T \simeq 1180$ [s]. In Figure 4.1, we show the exact solution at time T along with the numerical solutions for Godunov and Engquist-Osher schemes when we vary the mesh size Δz.

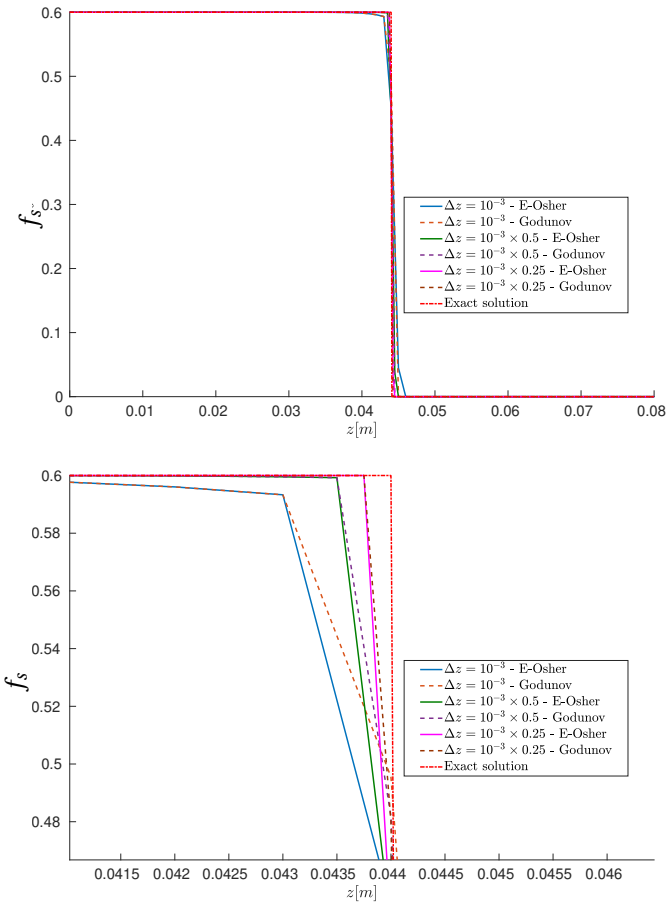


Figure 4.1 – Numerical and exact solutions for Godunov and Engquist-Osher schemes: unzoomed (top) and zoomed for $z \in [0.041, 0.046]$ (bottom). We take $\Delta t = \Delta z$.

Figure 4.1 shows that the solutions given by both schemes are very similar, the Godunov scheme being slightly better. In order to further assess this, we define the relative error E_{1_R} by

$$E_{1_R} = \frac{\sum_{i=1}^{N_z} |f_{s_i} - f_s(z_i, T)|}{\sum_{i=1}^{N_z} |f_s(z_i, T)|}. \tag{4.6}$$

Figure 4.2 illustrates the convergence order for the relative error (defined by (4.6)) for the Godunov and the Engquist-Osher schemes, along with the line corresponding to convergence order one.

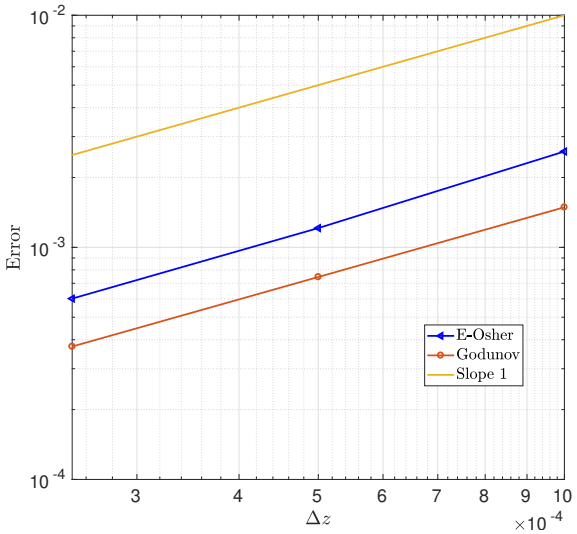


Figure 4.2 – Error (E_{1_R}) comparison for Godunov and Engquist-Osher schemes.

As expected, both schemes are of order 1. Moreover, we know that the Godunov scheme is exact for Riemann problems with convex fluxes [76, 77]. Thus, since the deposition flux F_D is strictly convex, the Godunov scheme gives better results than the Engquist-Osher scheme. The latter, however, shows appropriate convergence properties.

4.1.2 Sedimentation of polystyrene particles in a still fluid

The goal of this experiment is to further validate the numerical treatment of the sediment operator in a simplified multiphysics problem, and confirm the convergence of the method when the discretization parameters tend to zero.

In order to do so, let us consider the sedimentation of suspended, monodisperse, polystyrene particles in a tank of silicon oil. The particles properties are $d_* = 290$ [μm] and $\rho_p = 1.05$ [g cm^{-3}]; the silicon oil has $\mu_l = 0.02$ [Pa s] and $\rho_l = 0.95$ [g cm^{-3}]. The dimensions of the tank are $2.5 \times 0.05 \times 10$ [cm^3]. A mixing procedure is used to obtain a well-mixed, very dense, liquid mixture of uniform distribution $f_s(0) = 0.48$ in the bottom half of the domain ($z \leq 5.55$ [cm]). The maximal solid fraction is $f_{s_{CR}} = 0.6$, the cohesion parameter is $f_{s_{CO}} = 0.599$, $\varepsilon = 10^{-9}$, and $K = 1$.

This sedimentation process has been investigated in [127] with a comparison between a 1D model and experimental results, and in [147] with a multiphase 2D model. We consider the equation (4.2) with the same initial conditions as the ones given in Subsection 4.1.1. We recall that $\alpha = k v_{Stokes} / f_{s_{CR}}$.

As mentioned earlier, in such a case, an analytical solution can be obtained explicitly, and we obtain the development of two shockwaves, with shock speeds respectively equal to -0.12

and +0.48. The intersection of the two shocks takes place at $(t^\star, z^\star)$, with

$$z^\star = z^- + \alpha 0.48 t^\star = \alpha(-0.12) t^\star,$$

which leads to $(t^\star, z^\star) = (1084.0, 0.044)$ in the case of this particular experiment.

Figure 4.3 shows the snapshots of the sediment concentration at different times for our model with the parabolic flux ($\kappa = 1$). The numerical results show indeed the evolution of the two interfaces: one that separates the clear fluid at the top and the mixed fluid and another one between the mixed fluid and the fluid saturated with sediments at the bottom. Figure 4.4 illustrates the positions of these interfaces with, in particular: i) a comparison with the analytical solution derived previously, and ii) the evolution of the positions of these interfaces when the discretization parameters tends to zero. For the convergence study, we consider three discretizations, respectively, with:

(i) **Coarse mesh**: $H = 0.006638$ [m], $h = 0.001$ [m], and $\Delta t = 0.01$ [s]

(ii) **Intermediate mesh**: $H = 0.004978$ [m], $h = 0.00075$ [m], and $\Delta t = 0.0075$ [s]

(iii) **Fine mesh**: $H = 0.003319$ [m], $h = 0.0005$ [m], and $\Delta t = 0.005$ [s]

These results show not only a very good agreement of the computational results with the analytical ones but also the convergence of the method when the discretization parameters tend to zero.

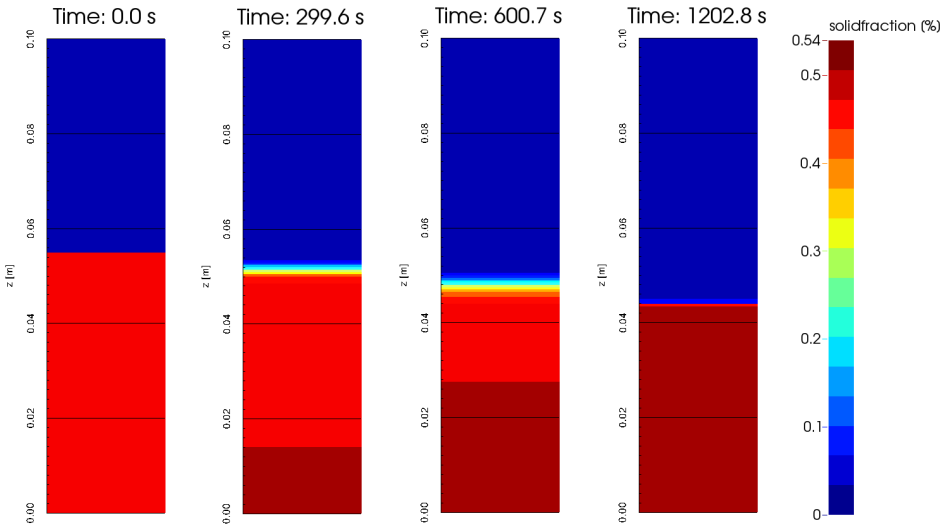


Figure 4.3 – Numerical simulation of the sedimentation of polystyrene particles. Snapshots of the solid fraction f_s.

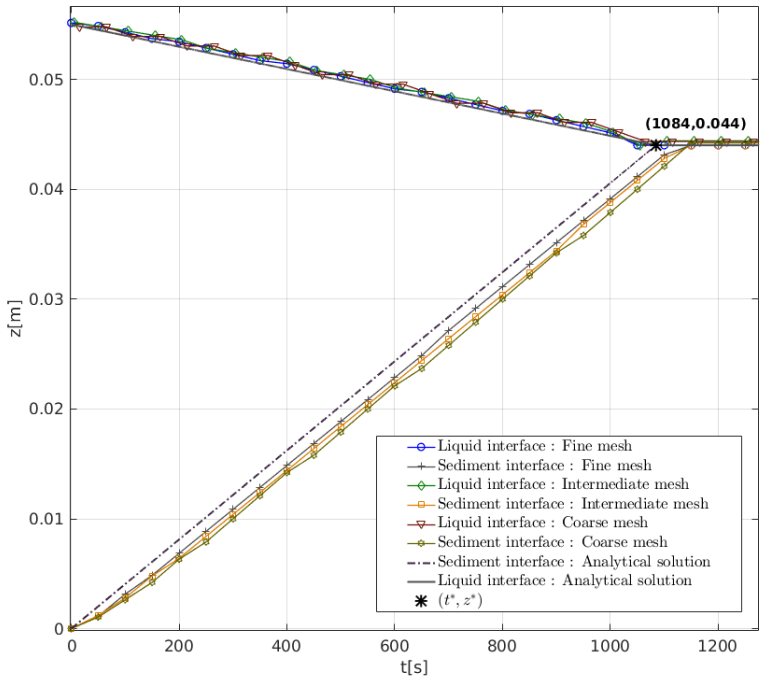


Figure 4.4 – Numerical simulation of the sedimentation of polystyrene particles. Time evolution of the two interfaces, including i) a comparison with the analytical solution, and ii) a convergence study when the discretization parameters tends to zero.

In order to quantitatively assess the convergence of the approximated solutions, we define two error estimates. The first estimate is based on the approximation error on the time trajectories of the interfaces (and the final time); the second is based on the L^1 error on the solution of the hyperbolic equation. Namely:

$$\begin{aligned} E_1 &= ||z_1 - z_{1h}||_{L^2(0,t^\star)} + ||z_2 - z_{2h}||_{L^2(0,t^\star)} + \left|t^\star - t_h\right|, \\ E_2 &= \left|\left|f_s - f_{s_h}\right|\right|_{L^1(\Lambda)}, \end{aligned}$$

where $z_1(t) = z^- + \alpha 0.48t$ (resp. $z_2(t) = \alpha(-0.12)t$) denote the time evolution of the lower and upper interfaces, z_{1h}, z_{2h} their respective numerical approximations, and $t^\star$ the exact time of intersection of the two trajectories. The position of the interfaces $z_{1h}(t)$ and $z_{2h}(t)$ are computed on the grid $\mathcal{C}_h$ as follows. The position $z_{1h}(t)$ is given by the vertical position of the center of the first cell (swept from bottom to top) such that $f_{s_{ijk}}^{n+1} > f_{s_{CR}} - \varepsilon_P$, where $\varepsilon_P = 10^{-5}$. Also, the position $z_{2h}(t)$ is given by the vertical position of the first cell (swept from top to bottom) such that $f_{s_{ijk}}^{n+1} < \varepsilon_P$. Figure 4.5 illustrates the convergence of these estimates when the discretization parameters h, H and Δt^n tend to zero, and shows an appropriate first order convergence order.

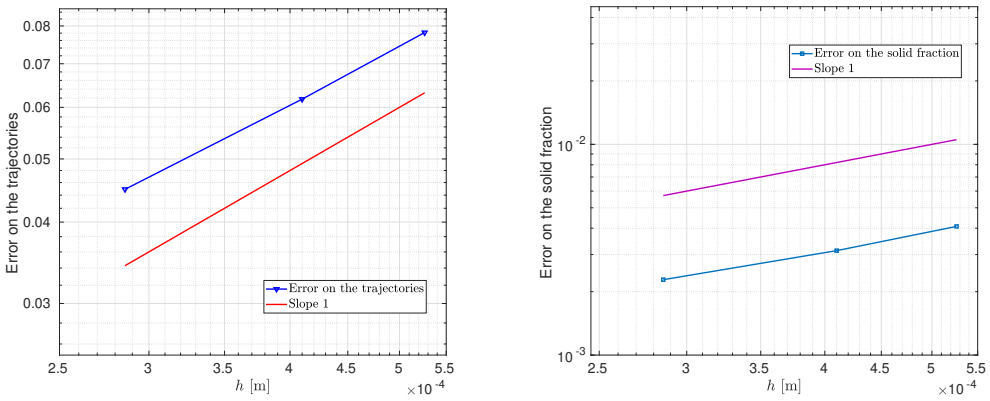


Figure 4.5 – Numerical simulation of the sedimentation of polystyrene particles. Convergence study when the discretization parameters tend to zero. Convergence of the error estimates E_1 (left) and E_2 (right) when $h \to 0$.

4.1.3 Erosion by an impinging liquid jet

In order to validate the numerical algorithm for the multiphysics model, we consider the benchmark simulation of the erosion of an immersed granular bed by an impinging liquid jet perpendicular to the surface of the consolidated sediment. We consider this erosion process as described in [148, 149]. The experiment consists of a bed load of non-cohesive sediments, initially at rest at the bottom of a parallelipidedic domain of size $H_e \times W_e \times W = 0.495 \times 0.032 \times 0.2$ [m^3], as illustrated in Figure 4.6. The bed load on non-cohesive sediment is composed by sand (with particles of diameter $4.0 \cdot 10^{-4}$ [m]); the dry density of the sand particles is 2500 [kg m^{-3}], while the bulk density is that of water, i.e. 1000 [kg m^{-3}]. The computational domain is actually smaller than the physical domain, and formed by a parallelipidedic domain of size $0.2 \times 0.004 \times 0.135$ [m^3]. The initial conditions consist of a domain entirely filled with liquid. The sediment concentration is maximal (i.e. $f_{s_{CR}} = 0.63$) when $z \leq 0.05$ [m] and zero otherwise. The water is vertically injected on the top of the domain with a velocity following a Poiseuille profile

$$\mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ 1.85 \cdot 10^5 (x - 0.102)(x - 0.098) \end{pmatrix} \text{ [m/s]},$$

on the inlet of size 0.004×0.004 [m^2] (so that the maximal velocity is $7.4 \cdot 10^{-1}$ [m/s]). This corresponds to a Reynolds number of $Re \simeq 630$, where $Re = \frac{\rho_l U_J b}{\mu_l}$, and U_J denotes the mean velocity of the inflow velocity, b is the diameter of the injection tube, and μ_l (resp. ρ_l) is the pure water viscosity (resp. density). The liquid can exit at the top of the domain (to compensate for the liquid injected), so that the domain is full at all times. The boundary conditions on the remaining part of the boundary of the domain are homogeneous Dirichlet boundary conditions, except for slip boundary conditions on the lateral walls along the pseudo-2D

direction. The water jet erodes the sand bed to create a hole of width D, and depth H_b. This numerical experiment is used to study the properties of the algorithm, to validate the mesh convergence of the algorithm, and to calibrate the numerical parameters of the model.

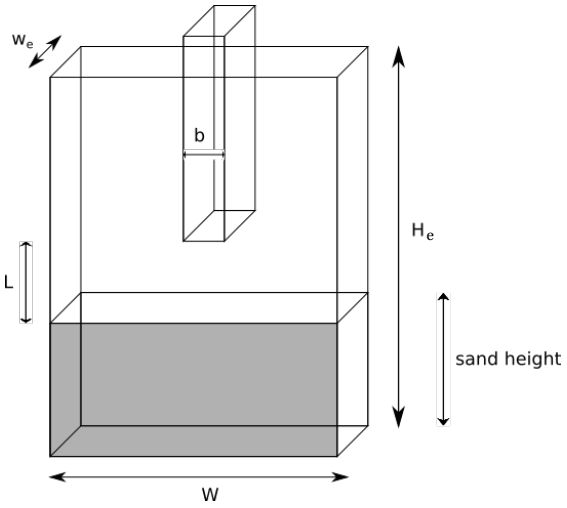


Figure 4.6 – Erosion by an impinging liquid jet. Sketch of the geometrical domain and numerical setup.

First, let us consider the grids containing 267,520 finite elements and 48,132 vertices in $\mathcal{T}_H$, and 1,071,360 cubic cells in $\mathcal{C}_h$. We calibrate the value of the time step to define a CFL number for the simulation that both allows us to maintain stability conditions, and minimize the computational effort. Numerical experiments have shown that a CFL number equal to 6 ($\Delta t = 0.003$ [s]) is appropriate to balance those effects. Figure 4.7 illustrates that this value leads to a less diffusive solution than when the CFL number is equal to 8.

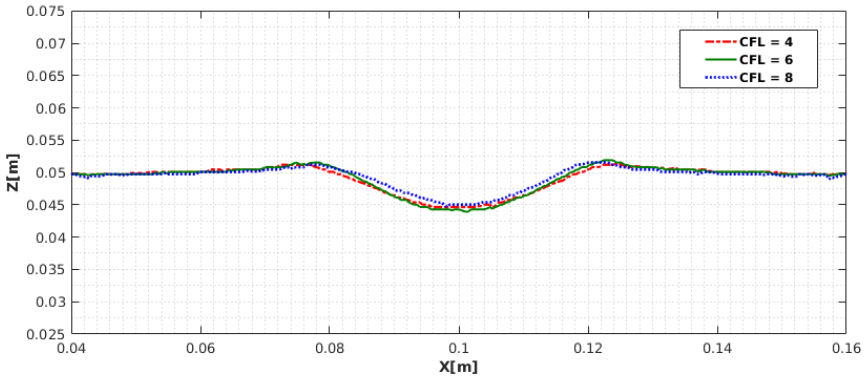


Figure 4.7 – Erosion by an impinging liquid jet. Effect of the CFL number on the solution.

Figure 4.8 shows the time evolution of the depth H_b and width D of the eroded hole respectively, and confirms that the erosion process has reached a stationary state after roughly

$T = 6$ [s]. We will use $T = 6$ [s] for all simulations in the sequel.

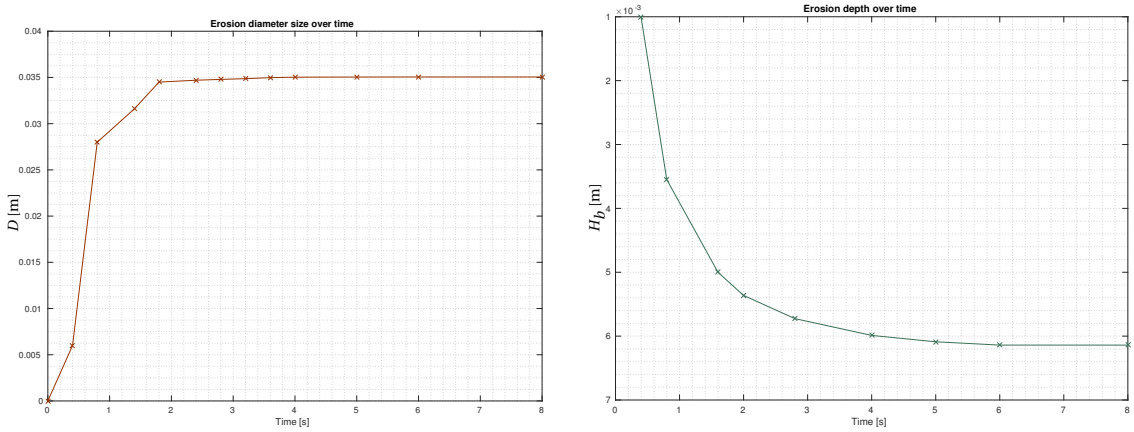


Figure 4.8 – Erosion by an impinging liquid jet. Left: time evolution of the numerical approximation of the hole width D. Right: time evolution of the numerical approximation of the hole depth H_b.

In order to study the convergence of the solution when the discretization parameters tend to zero, we consider three discretizations:

(i) **Coarse mesh:** 157,052 finite elements and 28,115 vertices in $\mathcal{T}_H$ ($H = 0.00135$ [m]), and 634,880 cubic cells in $\mathcal{C}_h$ ($h = 0.00054$ [m]), $\Delta t = 0.01$ [s]

(ii) **Intermediate mesh:** 490,866 finite elements and 86,051 vertices in $\mathcal{T}_H$ ($H = 0.0009$ [m]), and 2,142,720 cubic cells in $\mathcal{C}_h$ ($h = 0.00036$ [m]), $\Delta t = 0.006$ [s]

(iii) **Fine mesh:** 1,133,868 finite elements and 196,506 vertices in $\mathcal{T}_H$ ($H = 0.000675$ [m]), and 5,079,040 cubic cells in $\mathcal{C}_h$ ($h = 0.00027$ [m]), $\Delta t = 0.005$ [s]

Figure 4.9 illustrates the profile of the water-sand interface after $T = 0.8$ [s], and shows a convergence of the numerical approximations when the mesh sizes tend to zero.

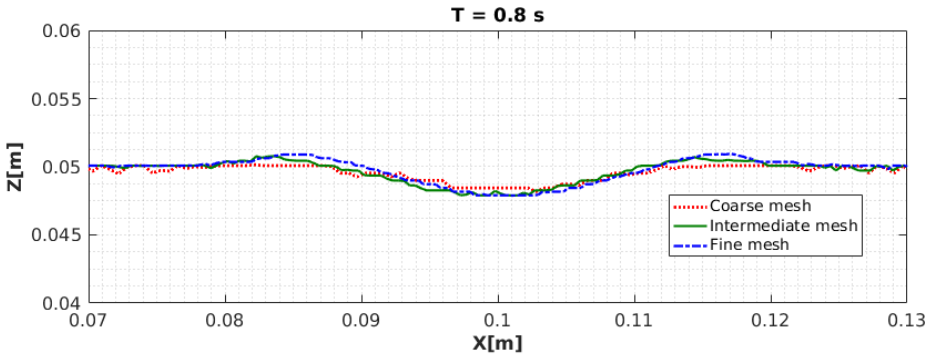


Figure 4.9 – Erosion by an impinging liquid jet. The convergence of the approximated solution when the discretization parameters tend to zero. Illustration of the sediment-water interface.

From now on, we consider the intermediate mesh, illustrated in Figure 4.10 and proceed to the calibration of the parameters involved in the model. The calibration of the model relies on determining the values of the numerical parameters ε (the regularization parameter in the Carman-Kozeny coefficient), and $f_{s_{CO}}$ (the cohesion threshold).

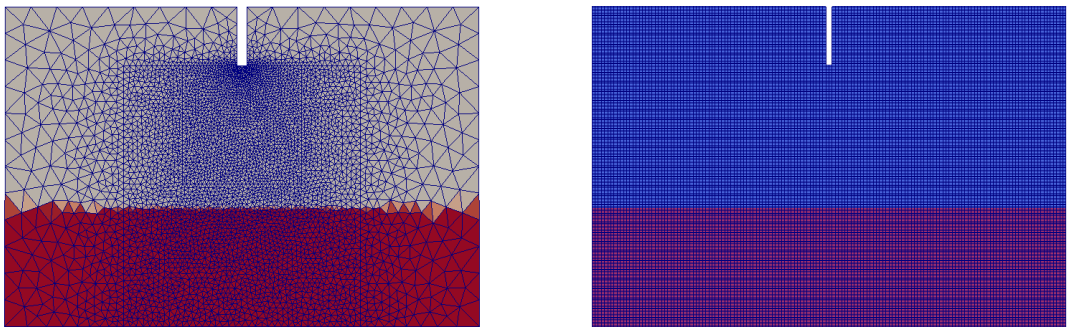


Figure 4.10 – Erosion by an impinging liquid jet. Intermediate mesh: Finite elements (left) and structured grid (right).

Numerical experiments, reported in Figures 4.11 and 4.12, have shown that optimal values are given by $\varepsilon = 10^{-9}$ and $f_{s_{CO}} = 0.62$. The choice of these values is based on the best fit (in terms of the sediment profile height) with experiments provided in [148, 149], as illustrated in the sequel for these optimal values. The results in Figures 4.12, **??** and 4.14, are presented in the same fashion as in [149], namely by expressing the normalized quantities $D/(L-\lambda)$ and $H_b/(L-\lambda)$, where $\lambda = 10b$ is a suggested correction of the distance L between the inlet and the sand interface, to account for the virtual origin, spreading angle and decay of the submerged

plane jet, as explained in [148], as a function of the erosion number E_c defined as

$$E_c = U_J\,(b/(L-\lambda))^{1/2}\left(\left(\frac{\rho_s}{\rho_l}-1\right)||\mathbf{g}||_2 d_*\right)^{-1/2}.$$

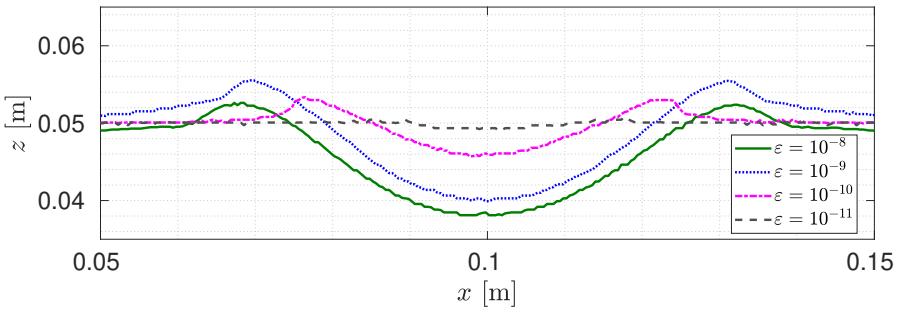


Figure 4.11 – Erosion by an impinging liquid jet. Sediment profile at $t = 3.5$ [s] for various values of the penalization parameter ε ($U_J = 0.28$ [m/s], $f_{s_{CO}} = 0.62$). For all cases, $K = 1$.

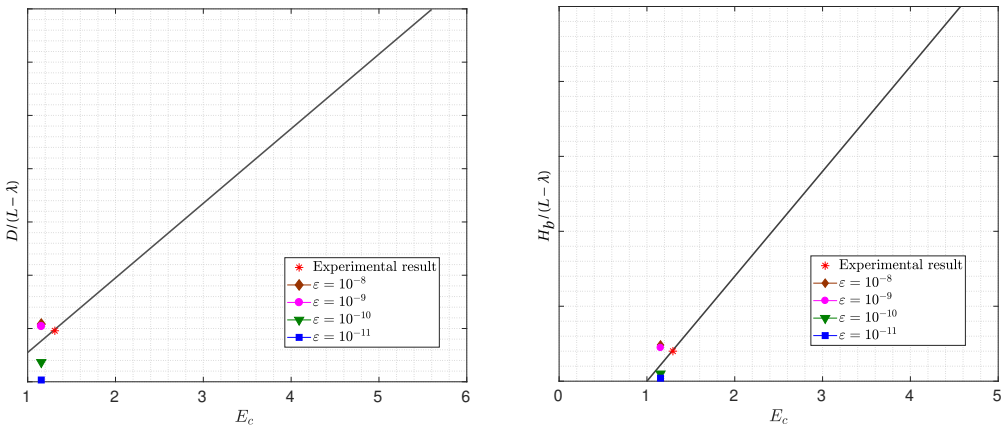


Figure 4.12 – Erosion by an impinging liquid jet. Numerical approximation of the width D and depth H_b of the eroded hole, as a function of the inlet velocity magnitude for various ε values; the regression lines are extracted from [149]. ($\lambda = 10b$, $U_J = 0.28$ [m/s], $f_{s_{CO}} = 0.62$).

Figure 4.13 illustrates snapshots of the solution, in particular, the sediment profile at various times for $U_J = 0.28$ [m/s], and illustrates the stationary regime of the flow velocity. The numerical results obtained with the proposed method are compared with experimental results from [148, 149]. Figure **??** illustrates the values of the width D and depth H of the eroded hole as a function of the inlet velocity magnitude. In agreement with [149], we consider inflow velocities inducing Reynolds numbers between $Re = 630$ and $Re = 1890$. Results in Figure 4.14 show that the higher the Reynolds number, the deeper and wider the eroded

hole. The regression lines $D/(L-\lambda) = 0.55 + 1.4(E-1)$ and $H_b/(L-\lambda) = 0.7(E-1)$, which are illustrated on the figures, are extracted from [149], compare well with the computational results, and validate our approach for such experiments. Furthermore, it was mentionned in [149] that data points remain close to the master curves for $\lambda = 10b$. Figure 4.15 shows the results when $\lambda = 10b$ is taken as the correction factor, which shows the sensitivity of the post-processed results with respect to post-processing choices.

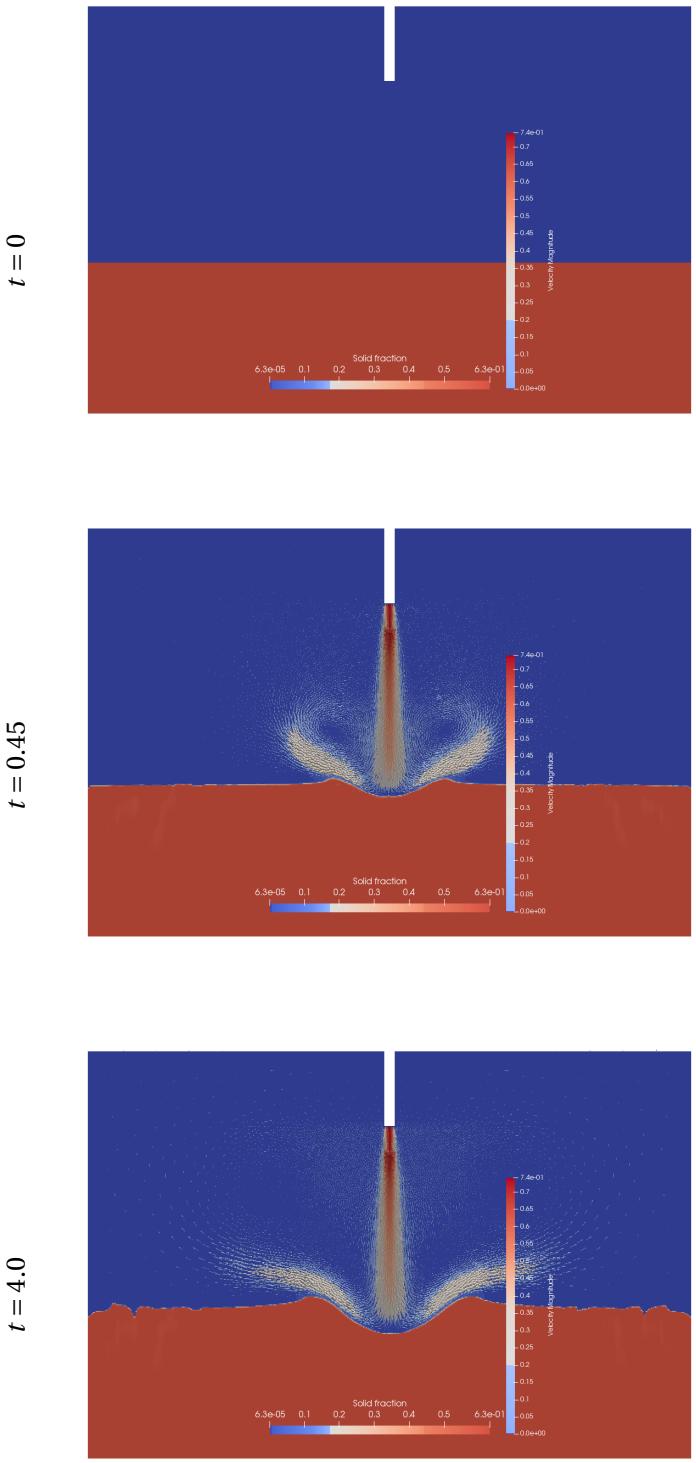


Figure 4.13 – Erosion by an impinging liquid jet. Snapshots of the numerical solution (sediment position and velocity field) at times $t = 0, 0.45$ and 4.0 [s], for $U_J = 0.45$ [m/s].

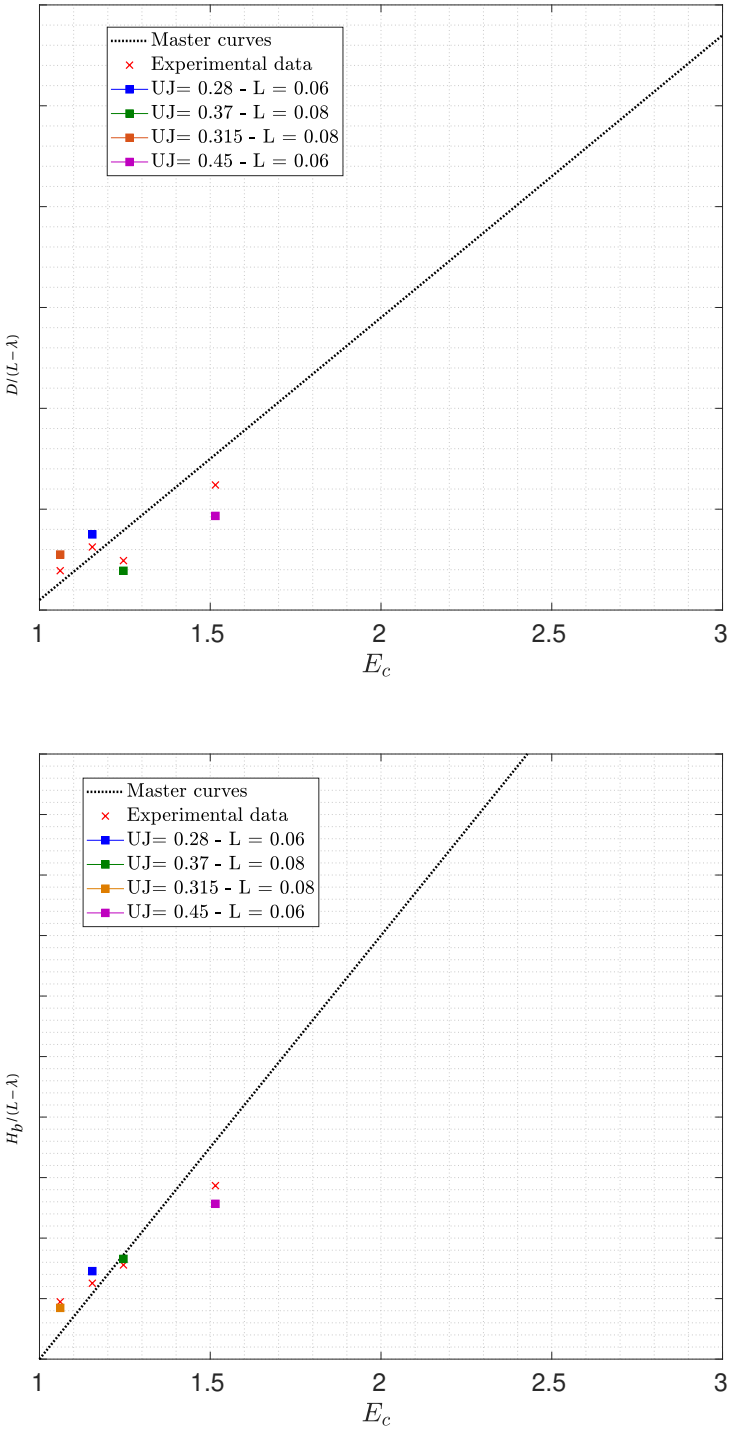


Figure 4.14 – Erosion by an impinging liquid jet. Numerical approximation of the width D (top) and depth H_b (bottom) of the eroded hole, as a function of the inlet velocity magnitude; the regression lines and experimental data (red crosses) are extracted from [149]. $\lambda = 5b$.

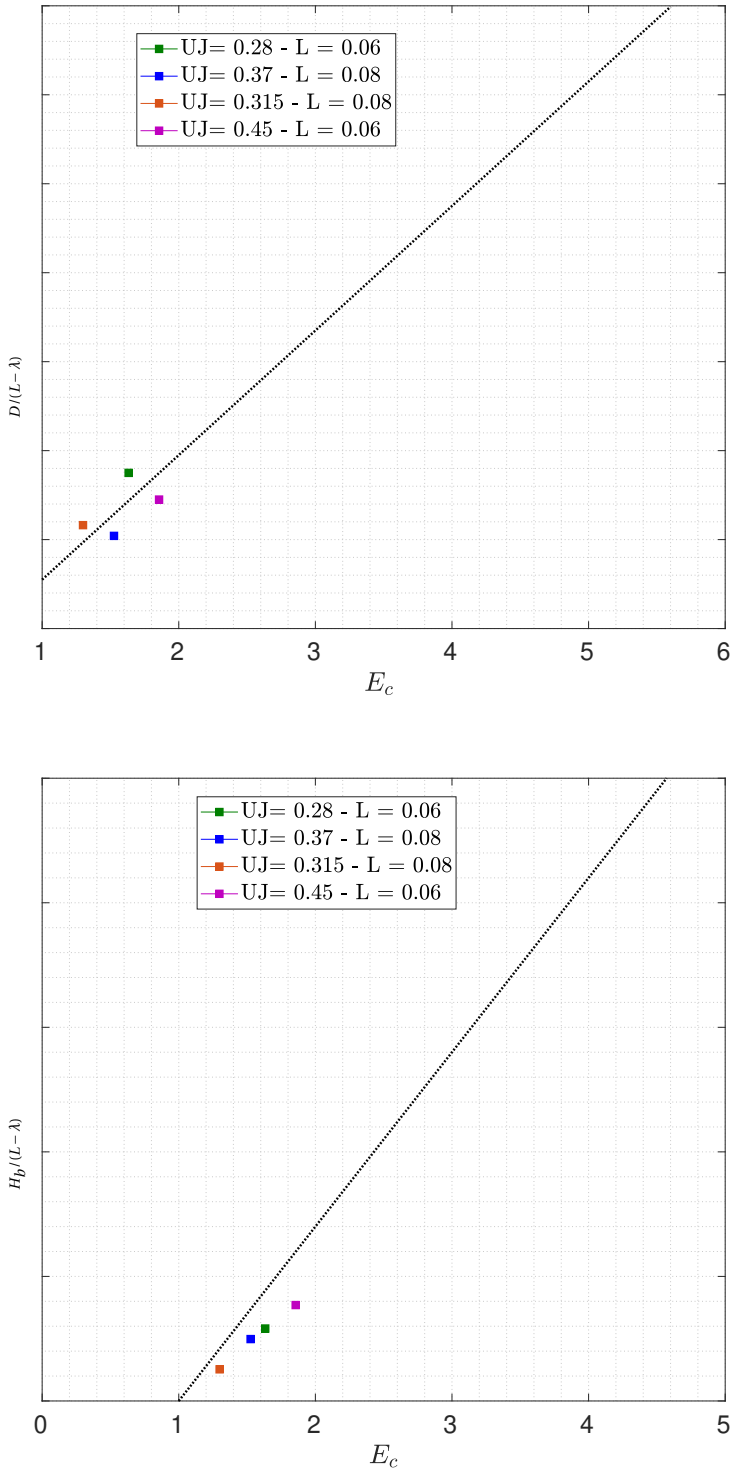


Figure 4.15 – Erosion by an impinging liquid jet. Numerical approximation of the width D (top) and depth H_b (bottom) of the eroded hole, as a function of the inlet velocity magnitude; the regression lines are extracted from [149]. $\lambda = 10b$.

4.1.4 Plunge pool scouring

The second experiment is the time evolution of plunge pool scour, caused by an inclined impinging jet of water on a sediment bed. The impact of the inclined jet into the sediments creates a pool scour and a ridge with the transported sediments. As highlighted in [150–152], such jets provoke turbulence in the flow but eventually converge to steady-state conditions for the scour depth and ridge height. Experimental results have been presented in [151, 152], while simulations with Flow3D have been validated in [150].

We consider the full 3D numerical setup used in [150] and illustrated in Figure 4.16. The experiment consists of a bed load of non-cohesive sediments, initially at rest at the bottom of a parallelipidedic domain of size $W \times w_e \times L = 1.9 \times 0.5 \times 0.65$ [m^3]. Let us consider three discretizations:

(i) **Coarse mesh:** 29,882 finite elements and 5336 vertices in $\mathcal{T}_H$, and 483,840 cubic cells ($H = 0.065$ [m], $h = 0.0036$ [m]), with $\Delta t = 0.012$ [s] .

(ii) **Intermediate mesh:** 42,433 finite elements and 7536 vertices in $\mathcal{T}_H$, 689,920 cubic cells ($H = 0.04$ [m], $h = 0.0024$ [m]), with $\Delta t = 0.01$ [s] .

(iii) **Fine mesh:** 76,380 finite elements and 13,549 vertices in $\mathcal{T}_H$, and 1,267,200 cubic cells in $\mathcal{C}_h$ ($H = 0.025$ [m], $h = 0.0016$ [m]), with $\Delta t = 0.0085$ [s] .

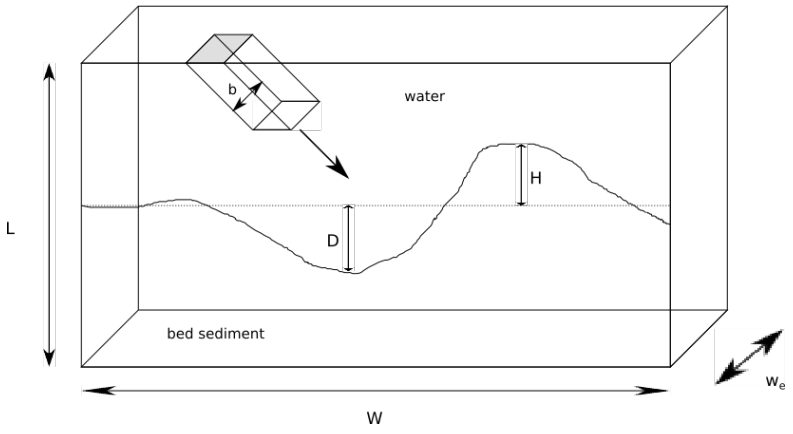


Figure 4.16 – Plunge pool scouring by an inclined impinging liquid jet. Sketch of the geometrical domain and numerical setup.

The bed load of non-cohesive sediment is composed of nearly uniform particles of diameter $d_* = 0.00125$ [m]; the dry density of the particles is $\rho_s = 1400$ [kg m^{-3}], while the bulk density is that of water, i.e. $\rho_l = 1000$ [kg m^{-3}]. The initial conditions consist of a domain entirely filled with liquid, with zero velocity. The sediment concentration is maximal (i.e. $f_{s_{CR}} = 0.6$) when $z \leq 0.3$ [m] and zero otherwise. The cohesion threshold is given by $f_{s_{CO}} = 0.599$, and $\varepsilon = 10^{-9}$.

The water is injected from the top of the domain with a jet dimension $b = 0.035[m]$ and a jet impact angle $\alpha = 45°$. The jet discharge Q is 3.5 [L/s]. This results in a diagonally injected water jet with velocity $\mathbf{u}_{in} = (4.04, 0, -4.04)^T$ [m/s]. The water level h_0 is kept such that the ratio $T_w = \frac{h_0}{b}$ (referred to by tail-water i.e. the average water depth)is equal to 5.9.

The boundary conditions on the remaining part of the boundary of the domain are homogeneous Dirichlet boundary conditions, except for slip boundary conditions on the symmetry plane as illustrated in Figure 4.17.

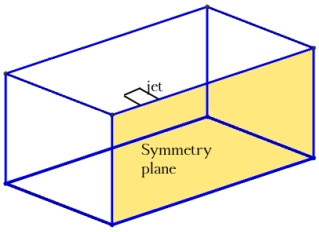


Figure 4.17 – Plunge pool scouring by an inclined impinging liquid jet. The symmetry wall is where slip conditions are imposed. Wall conditions are imposed on the rest of the surfaces (except the inflow).

We introduce the dimensionless time $\tau = (g' d_*)^{\frac{1}{2}}(t/b)$ where t denotes the time [s] and $g' = ||\mathbf{g}||_2(\rho_s - \rho_l)/\rho_l$ denotes the reduced gravitational acceleration, and d_* denotes the sediment particles diameter. The end of the calculations is not that of the experiments but whenever the steady-state of the scour characteristics is observed. Figure 4.19 illustrates snapshots of the solution obtained with the intermediate mesh, in particular, the sediment profile at various times. The scour depth D and the nearby ridge H_b are developing until a state where they only vary mildly after about 40 [s]. Figure 4.18 illustrated the intermediate mesh.

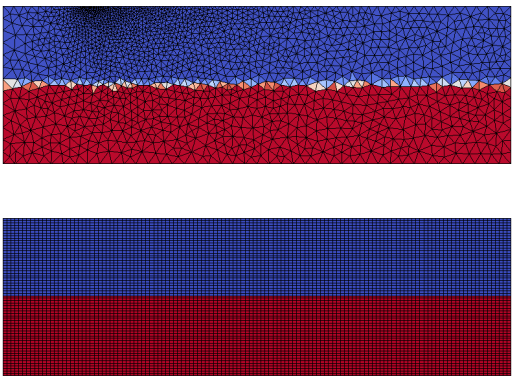


Figure 4.18 – Plunge pool scouring by an inclined impinging liquid jet. Intermediate mesh: finite elements (top), structured grid (bottom).

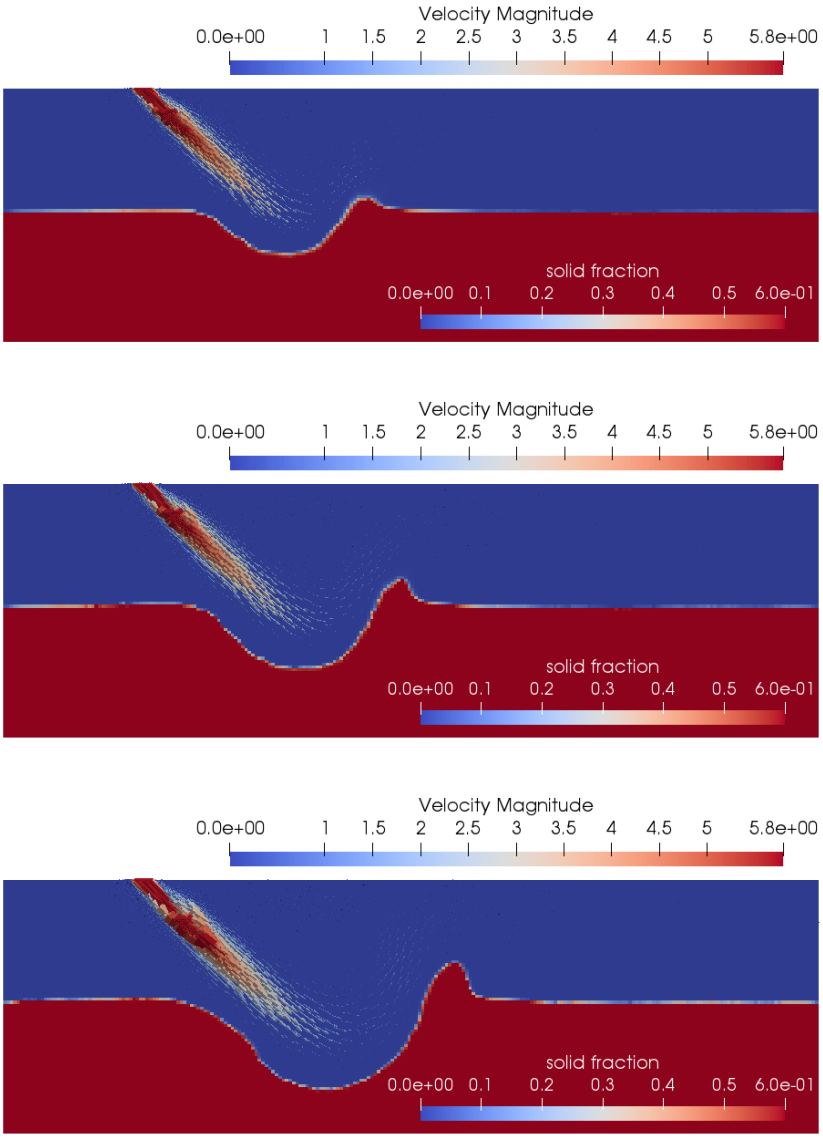


Figure 4.19 – Plunge pool scour by an inclined impinging liquid jet. Snapshots of the numerical solution (sediment position and velocity field) at times $t = 5, 10$ and 30 [s] (top to bottom).

Figure 4.20 illustrates the time evolution of the stationary dimensionless depth $D_d = D/b$, and the dimensionless ridge height $H_d = H_b/b$, as a function of the dimensionless time τ. Results are post-processed as in [150], and compared with experimental results. We observe that, albeit some numerical diffusion appears (especially for coarser meshes, which is not surprising for 3D calculations), the slope of the graph is adequately reproduced; furthermore, the numerical results improve when the mesh discretization parameters tend to zero.

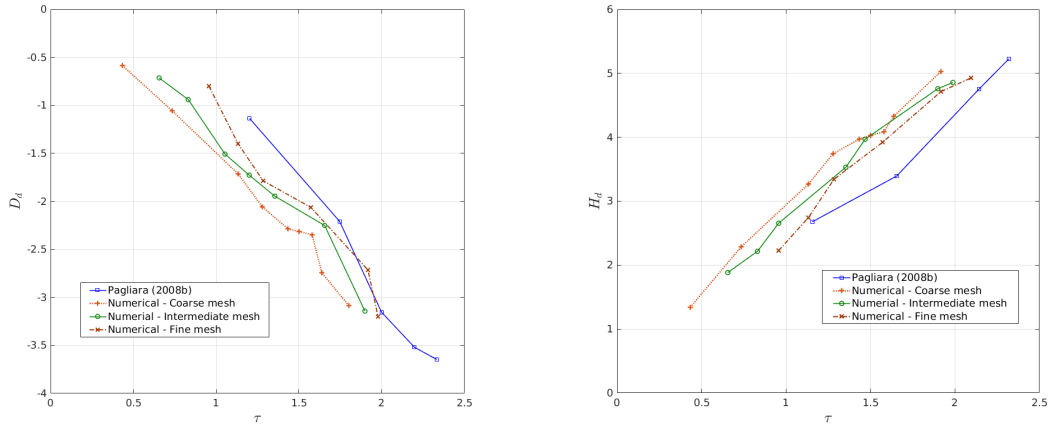


Figure 4.20 – Plunge pool scour by an inclined impinging liquid jet. Comparison of the time evolution of dimensionless scour depth D_d and ridge height H_d derived from the experiments and those derived from numerical simulations for a coarse, intermediate, and fine meshes.

4.1.5 Sediment flushing

We discuss a last numerical experiment, in order to define the range of capability of the physical model with deposition only, but also illustrate its limits. We consider a flushing process as described in [153] (and references therein). The importance of flushing techniques has been described, e.g., in [154], for both industrial applications in dams and natural configurations in rivers and lakes. The experiment consists of a bed load of non-cohesive sediments, initially at rest at the bottom of a channel, as illustrated in Figure 4.21. It is realized in a laboratory flume of width 0.2 [m]. The bed load of non-cohesive sediment is composed by sand (with particles of diameter $d_* = 7.6 \cdot 10^{-4}$ [m]); the dry density of the sand particles is $\rho_s = 2650$ [kg m^{-3}], while the bulk density is $\rho_l = 1750$ [kg m^{-3}]. Under vertical gravity forces, the liquid flows out of the domain via the valve on the right, and the sediment is flushed by the rapid flow.

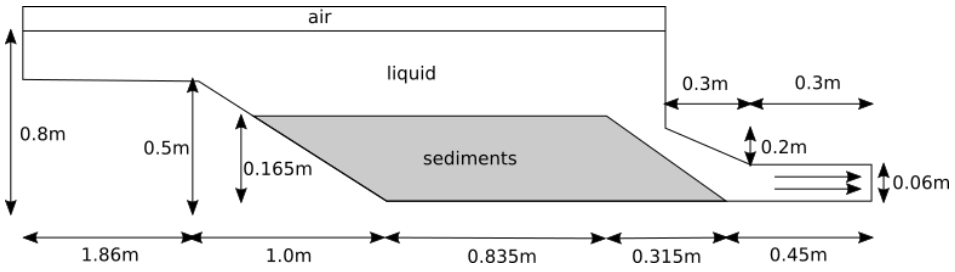


Figure 4.21 – Flushing of sediments. Sketch of the geometrical domain and numerical setup (similar as in [153]).

No inflow condition is enforced, but an outflow condition, by imposing a discharge of $q_0 =$ 0.0079 [m^3/s] on the bottom right part of the domain (which corresponds to an outflow velocity

of 0.132 [m/s]). A free surface between liquid and air lies at the top of the domain, such that the domain is initially full but the liquid level decreases as the liquid flows out. Slip boundary conditions are imposed everywhere else. Initially, the liquid and sediments are at rest.

We first evaluate the convergence of the numerical method, when the discretization parameters (time step and mesh size) decrease. In order to do so, we run a simulation over 2.5 [s]. We consider three discretizations:

(i) **Coarse mesh:** 5592 elements, 1,144 nodes, 646,800 cells ($H = 0.03$ [m], $h = 0.0058$ [m]) and $\Delta t = 0.01$ [s]

(ii) **Intermediate mesh:** 32,710 elements, 6,237 nodes, 5,174,400 cells ($H = 0.01$ [m],$h = 0.002$ [m]) and $\Delta t = 0.003$ [s]

(iii) **Fine mesh:** 218,554 elements, 39,430 nodes, 41,395,200 cells ($H = 0.005$ [m],$h = 0.001$ [m]) and $\Delta t = 0.001$ [s]

Figure 4.22 shows the convergence of the solution when the time step and mesh sizes decrease. The top figure illustrates the sediment profile after $T = 2.5$ [s] for the three meshes, while the bottom figure visualizes the position of the sediment front (position of the first particle of sand flushed to the right) for the three meshes. Based on both figures, numerical experiments show some convergence of the numerical solution, for both the sediment profile and the position of the front, when the discretization parameters tend to zero.

In a second step, numerical results are compared with experimental results in [153]. We extend the simulation until $T = 48.0$ [s]. Figure 4.23 illustrates a comparison between computed and measured sediment profiles. Computed results do not provide profiles with exactly the same slope for the sediment bed as that of experimental results.

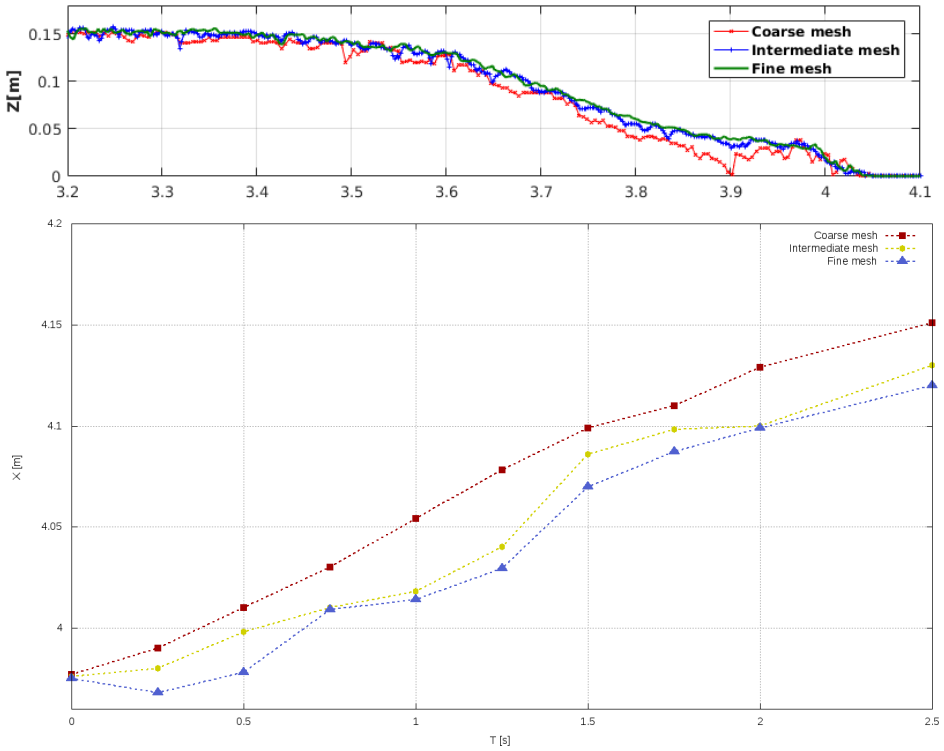


Figure 4.22 – Sediments flushing. Convergence of the numerical solution when the discretization parameters tend to zero. Top: sediment profiles at time $T = 2.5$ [s] for various mesh sizes. Bottom: time evolution of the position of the sediment front.

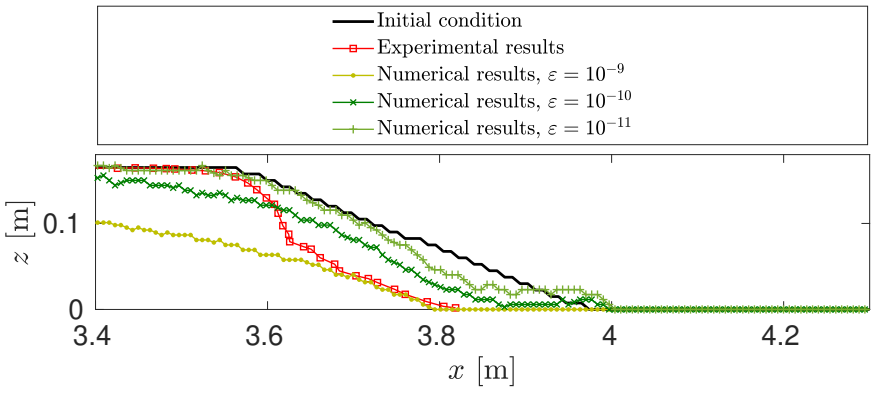


Figure 4.23 – Sediments flushing. Comparison of sediment profiles at time $T = 42.0$ [s] between the computed solution for various ε and $f_{s_{CO}} = 0.46$, and the experimental results from [153].

We infer that this difference comes from the limited physical model considered here, which does not include the resuspension effects of the sediments in the liquid. These effects have been highlighted in [153], when considering the so-called Shields model, and have been deemed to be necessary to match experimental results. This is when the resuspension effects become necessary. In fact, a sensitivity analysis study has been performed without resuspension effects for this experiment, but we elect to illustrate the results with resuspension effects in Subsection 4.2.4. The sensitivity analysis shows that the model with deposition effects alone does not suffice for this experiment.

To conclude this part, the operator splitting strategy, and the appropriate mix of finite elements, finite volumes, and structured grids, has proved to be very flexible to incorporate various numerical solvers. In this first part, the model has been calibrated versus experiments. The numerical results agree with experimental measurements for pure sedimentation, erosion processes, and impinging jets. However, a more complete physical model for the sediment resuspension is missing to adequately model flushing experiments for instance. Future computational results will thus include the extension of the model with the resuspension of cohesive particle bed. In the next section, we validate the numerical methods used to solve the full sedimentation equation (including the resuspension) then we proceed with the benchmarking process through numerical experiments.

4.2 Results of sedimentation with deposition and resuspension

Our goal is now to validate the resuspension flux F_r in (2.8). It includes the following steps:

1. To validate the computation of the shear stress.
2. To validate and compare the methods proposed to solve the complete model. (first with resuspension only, then with deposition and resuspension effects)
3. To validate the complete physical model with experimental data.

The goal is to better match some physical phenomena that require resuspension effects, for instance the sediment flushing experiment discussed in Subsection 4.1.5.

4.2.1 Validation of the shear stress computation

As specified in (2.10), the resuspension only takes place when the bedload shear stress $\tau(f_s, \mathbf{v})$ is larger than a critical value τ_{CR}. This makes the computation of shear stress crucial in the resuspension model. In this section, we want to validate the computation of the shear stress, and, to do so, we consider a simplified situation:

- We consider a rectangular domain of size $(L_x, L_y, L_z) = (0.05, 0.005, 0.025)$, initially full of water (with viscosity $\mu_l = 10^{-3}$ [Pa.s]).
- We impose a velocity profile such that $\mathbf{v} = 6z\mathbf{e_x}$ as illustrated in Figure 4.24,
- In the expression of the shear stress (2.11), we take $\mathbf{n}(f_s) = \mathbf{e_z}$.
- We consider $f_s = 0$ (no sediments), which leads to many simplifications: Indeed, when $f_s = 0$, the mixture viscosity $\mu_m(f_s)$ given by (2.5) is equal to the liquid viscosity μ_l, and the penalization parameter $\alpha_m(f_s)$ given by (2.6) becomes zero. This also means that we end-up not solving the sedimentation equation (4.1), but we still compute the expression of the shear stress for validation purposes.
- In this domain, we solve the volume-of-fluid equation (2.1) (the volume fraction of liquid being one everywhere) and the Navier-Stokes equations (2.2), (2.3) as described in Section 2.1.2.

Using the simplifications above ($\mathbf{v} = 6z\mathbf{e_x}$ [m/s]) and the shear stress expression (2.11), we can compute the exact value of the shear stress which is constant and equal to 6×10^{-3} [N/m^2]. This value is compared to the computed one. Figure 4.25 shows that the computed shear stress magnitude oscillates around the exact value.

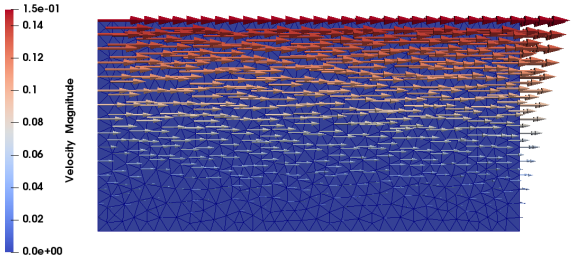


Figure 4.24 – Imposed velocity profile : $\mathbf{v} = 6z\mathbf{e_x}$ [m/s].

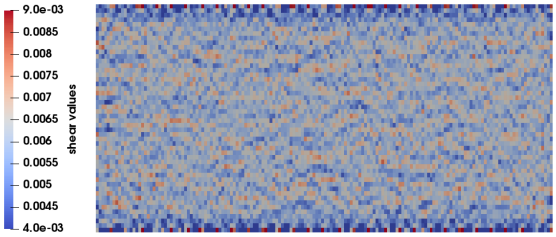


Figure 4.25 – Shear stress magnitude displayed on the strctured grid cells such that $\mathbf{v} = 6z\mathbf{e_x}$ [m/s], when the velocity is computed (interpolated from the finite elements to the cells grid). The exact value is 0.006 [N/m^2].

In order to confirm that these oscillations are due to numerical errors, we run a similar test case with the same velocity values. The only difference is that we impose the exact values of the velocity on the cells using the expression $\mathbf{v} = 6z\boldsymbol{e_x}$, instead of using the interpolated values from the finite elements. We can observe in Figure 4.26 that, in this case, the shear stress magnitude is precisely the expected constant.

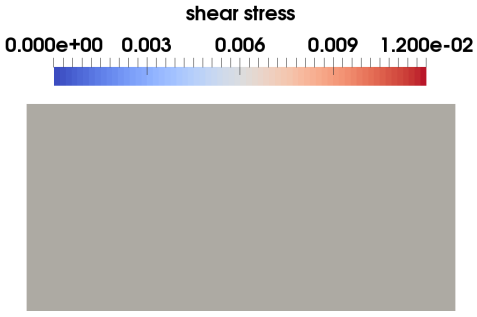


Figure 4.26 – Shear stress magnitude values when the velocity on the cells is imposed instead of interpolated from the finite elements.

Then, we run the experiment on three different meshes to perform a convergence study:

(a) Coarse mesh: 540 elements, 218 nodes (Average size $H = 0.00586$), 4000 cells ($h = 0.00083$) and $\Delta t = 0.016$.

(b) Intermediate mesh: 4121 elements, 1612 nodes (Average size $H = 0.00275$), 32000 cells ($h = 0.00038$) and $\Delta t = 0.008$.

(c) Fine mesh: 32590 elements, 11360 nodes (Average size $H = 0.0015$), 256000 cells ($h = 0.000212$) and $\Delta t = 0.004$.

In Figure 4.27, we plot the shear stress values for the three meshes over the vertical line (at $x = x_{max}/2$ and $y = y_{max}/2$). We illustrate the snapshots of the shear stress on each mesh in Figure 4.29. In Figure 4.28, we plot the L^2 error for the coarse, intermediate and fine meshes. These figures show that, even though refining the mesh does not reduce much the shear stress oscillations, the error indeed decreases with a convergence order equal to one.

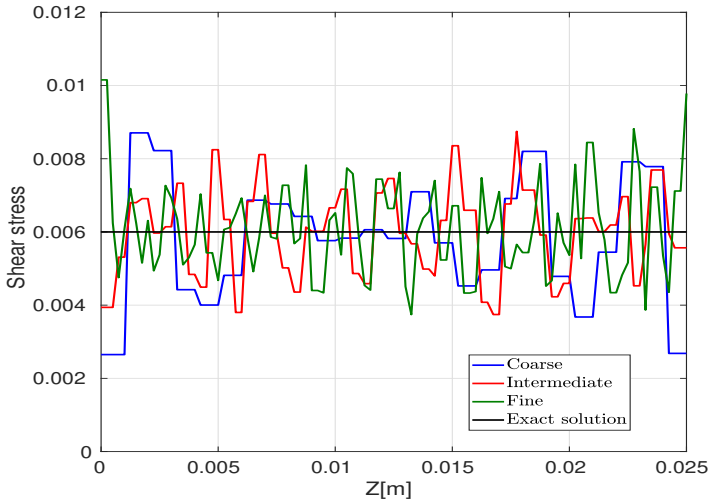


Figure 4.27 – Shear values along the vertical axis at $x = x_{max}/2$ and $y = y_{max}/2$) for the coarse, intermediate and the fine meshes. The exact solution is constant and represented by a black line at 0.006.

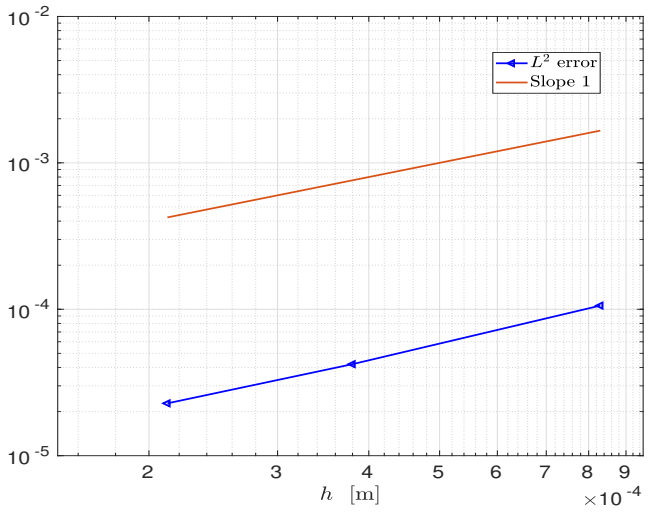
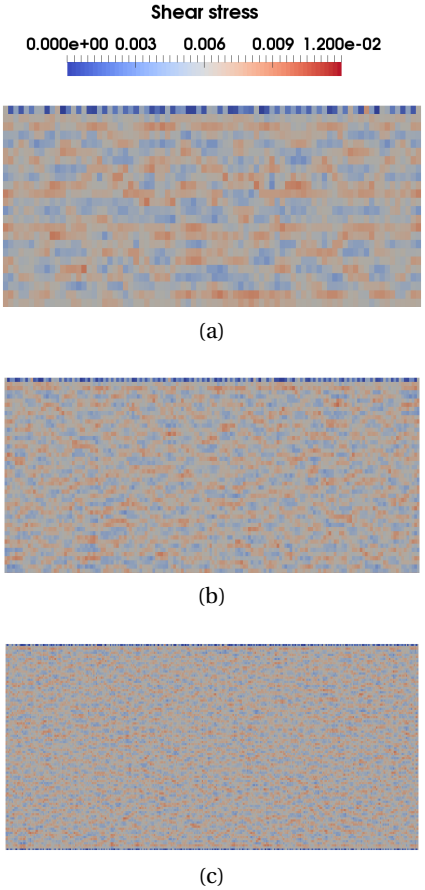


Figure 4.28 – L^2 error of the shear stress computation for the coarse, intermediate and fine meshes.



(a)

(b)

(c)

Figure 4.29 – Snapshots of the shear stress on a : (a) Coarse mesh, (b) Intermediate mesh and (c) Fine mesh.

As a result, we implemented an algorithm to smooth the shear stress and reduce these oscillations. In order to smooth the shear stress τ^n_{ijk}, computed using (3.17), we consider the shear stress quantities from the twenty-six neighboring cells i.e. we consider a cube of twenty-seven cells, where the center of the cell C_{ijk} is the center of the cube. Then, we make a local average over the twenty-seven cells with different weights. The weights are attributed such that, the closer the neighbouring cell is to C_{ijk}, the more significant the weight is. This results in the restriction operatpr (4.7) to compute a smoothened shear $\tau^n_{s_{ijk}}$ (we refer to [155, pages 64-65]. We illustrate the described process in a two-dimensional setup in Figure 4.30.

$\frac{1}{16}$ C_{i-1j+1}	$\frac{2}{16}$ C_{ij+1}	$\frac{1}{16}$ C_{i+1j+1}
$\frac{2}{16}$ C_{i-1j}	$\frac{4}{16}$ C_{ij}	$\frac{2}{16}$ C_{i+1j}
$\frac{1}{16}$ C_{i-1j-1}	$\frac{2}{16}$ C_{ij-1}	$\frac{1}{16}$ C_{i+1j-1}

Figure 4.30 – An illustration of the shear stress smoothing process in two dimensions. The smoothened shear stress at cell C_{ij} is an averaged quantity using the illustrated neighbouring cells with the specified weights.

$$\begin{aligned}\tau^n_{s_{ijk}} = \frac{1}{64}\Big[&8\tau^n_{ijk} + 4\Big(\tau^n_{ij-1k} + \tau^n_{ij+1k} + \tau^n_{ijk-1} + \tau^n_{ijk+1} + \tau_{i-1jk} + \tau_{i+1jk}\Big)\\ &+ 2\Big(\tau^n_{i+1jk+1} + \tau^n_{i+1jk-1} + \tau^n_{i-1jk-1} + \tau^n_{i-1jk+1} + \tau^n_{ij+1k+1} + \tau^n_{ij-1k+1} + \tau^n_{ij-1k-1} + \tau^n_{ij+1k-1}\\ &+ \tau^n_{i+1j-1k} + \tau^n_{i+1j+1k} + \tau^n_{i-1j-1k} + \tau^n_{i-1j+1k}\,) + \Big(\tau^n_{i+1j-1k+1} + \tau^n_{i+1j-1k-1} + \tau^n_{i+1j+1k-1}\\ &+ \tau^n_{i+1j+1k+1} + \tau^n_{i-1j-1k-1} + \tau^n_{i-1j-1k+1} + \tau^n_{i-1j+1k+1} + \tau^n_{i-1j+1k-1}\,)\,\Big]. \end{aligned} \tag{4.7}$$

In two space dimensions (as illustrated in Figure 4.30), the formula becomes:

$$\tau^n_{s_{ij}} = \frac{1}{16}\Big(4\tau^n_{ij} + 2\Big(\tau^n_{ij-1} + \tau^n_{ij+1} + \tau^n_{i+1j} + \tau^n_{i-1j}\Big) + \Big(\tau^n_{i-1j-1} + \tau^n_{i+1j-1} + \tau^n_{i+1j+1} + \tau^n_{i-1j+1}\Big)\Big).$$

On the boundaries $i = 0, i = N_x$, $j = 0, j = N_y$ or $k = 0, k = N_z$, we use a modified smoothing

formula that requests values from available neighbouring cells only. For example, for $i = 0$,

$$\tau^n_{s_{0jk}} = \frac{1}{12}\left(4\tau^n_{0jk} + 2\tau^n_{0jk+1} + 2\tau^n_{0jk-1} + 2\tau^n_{1jk} + \tau^n_{1jk+1} + \tau^n_{1jk-1}\right) \tag{4.8}$$

Furthermore, if we want to enhance the smoothing effect, we can apply recusively the described process more than once.

In Figure 4.31, on the intermediate mesh, we plot the exact shear values as well as the smoothened shear stress when we apply the smoothing process once, twice, and three times. We can perceive that the smoothing process reduces the oscillations around the exact values. However, we can quickly observe that the effect of smoothing decreases after smoothing twice. Later on, in the benchmarking phase, we smooth the shear stress on average only once.

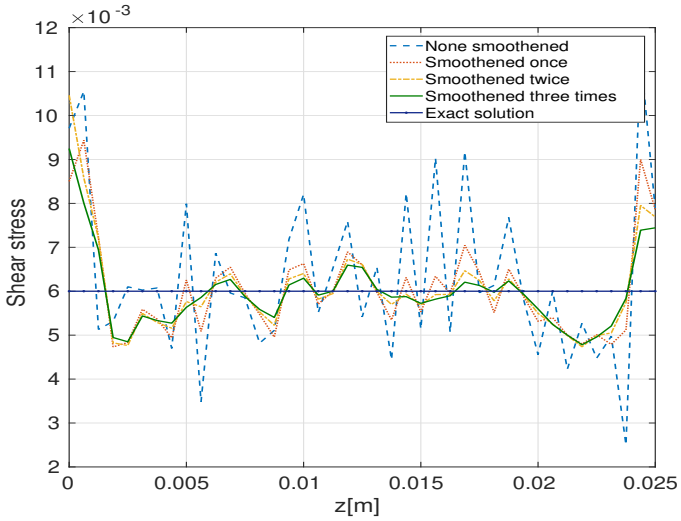


Figure 4.31 – Shear values along the vertical axis at $x = x_{max}/2$, $y = y_{max}/2$ on the intermediate mesh: effect of multipe times smoothing in comparison with the exact values.

4.2.2 Deposition and resuspension: one-dimensional test examples

In this part, we want to test the numerical methods that we used to solve the sedimentation problem (which we presented in Subsection 3.2.3). The deposition has already been tested earlier in this chapter. Thus, we dedicate this part to validate the resuspension in a simplified one-dimensional setup. First, we consider a resuspension problem (without deposition effects).

We look for $f_s \in [0, f_{s_{CR}}]$ solution of

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1 - \frac{f_s}{f_{s_{CR}}}\right)\left(-\gamma\frac{\partial f_s}{\partial z}\right)\right) = 0, \tag{4.9}$$

where γ is a positive constant playing the role of resuspension. To solve (4.9), we first use a finite differences scheme, then a finite volumes scheme. We already described the schemes in Subsection 3.2.3, but we recall them here for this simplified case (we use the same notation introduced in Chapter 2):

The finite differences scheme We denote $F_l(f_s) = f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right)$. The scheme writes

$$\frac{(f_s)_i^{n+1} - (f_s)_i^n}{\Delta t^n} = (FD_Z)_i^n,\ 1 \le i \le N_z, \tag{4.10}$$

where

$$(FD_Z)_i^n = \frac{\gamma F^n_{i+\frac{1}{2}} - \gamma F^n_{i-\frac{1}{2}}}{(\Delta z)^2},$$

and

$$F^n_{i+\frac{1}{2}} = F_l\left(\frac{(f_s)^n_{i+1} + (f_s)^n_i}{2}\right)\left((f_s)^n_{i+1} - (f_s)^n_i\right),\ 1 \le i \le N_z.$$

The finite volumes scheme The scheme writes

$$(f_s)_i^{n+1} = (f_s)_i^n - \frac{\Delta t^n}{\Delta z}\left((FV_Z)^n_{i+\frac{1}{2}} - (FV_Z)^n_{i-\frac{1}{2}}\right),\ 1 \le i \le N_z \tag{4.11}$$

where the numerical flux (FV_Z) is given by:

Godunov scheme:

$$(FV_Z)^n_{i+\frac{1}{2}} = \begin{cases} \max\limits_{(f_s)^n_i \le f_s \le (f_s)^n_{i+1}} \left[-f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right)\left(\gamma {G_Z}^n_{i+\frac{1}{2}}\right)\right] \text{if} (f_s)^n_{i+1} \ge (f_s)^n_i,\ 1 \le i \le N_z, \\ \min\limits_{(f_s)^n_{i+1} \le f \le (f_s)^n_i} \left[-f_s\left(1 - \dfrac{f_s}{f_{s_{CR}}}\right)\left(\gamma {G_Z}^n_{i+\frac{1}{2}}\right)\right] \text{if} (f_s)^n_{i+1} < (f_s)^n_i,\ 1 \le i \le N_z, \end{cases} \tag{4.12}$$

where ${G_Z}^n_{i+\frac{1}{2}} = \dfrac{(f_s)^n_{i+1} - (f_s)^n_i}{\Delta z}$.

Engquist-Osher scheme:

$$(FV_Z)^n_{i+\frac{1}{2}} = \frac{1}{2}\left[F_R((f_s)^n_{i+1}) + F_R((f_s)^n_i) - \int_{(f_s)^n_i}^{(f_s)^n_{i+1}} |F_R'(f_s)| df_s\right],\ 1 \le i \le N_z, \tag{4.13}$$

where $F_R(f_s) = f_s(1 - \dfrac{f_s}{f_{s_{CR}}})\left(-\gamma {G_Z}^n_{i+\frac{1}{2}}\right)$, for all $1 \le i \le N_z$.

We would like to study these algorithms on simple test cases. In order to do so, we

construct the appropriate right-hand side g such that an exact solution to

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1 - \frac{f_s}{f_{s_{CR}}}\right)\left(-\gamma\frac{\partial f_s}{\partial z}\right)\right) = g(z,t),$$

is:

- Example 1: $f_{s_{exact}}(z,t) = \dfrac{z^2+t}{2}$.
- Example 2: $f_{s_{exact}}(z,t) = \sin^2(\omega_1\pi z)\sin^2(\omega_2\pi t)$, with $\omega_1, \omega_2 > 0$.

We start with Example 1. First, we use the finite differences method and start by conducting a mesh convergence study. For every variation of N_z, Δt is also updated accordingly in order to verify the stability condition stated in Section 3.2.3, which is

$$\Delta t \leq \frac{\Delta z^2}{2}.$$

In Figure 4.32 (left), we show the exact solution and the computed solutions at $t = 150s$, for various meshes. On the right, we show the same plots zoomed in the region where $z \in [0.5985, 0.6015]$. We observe that the solution converges when the discretization parameters go to zero.

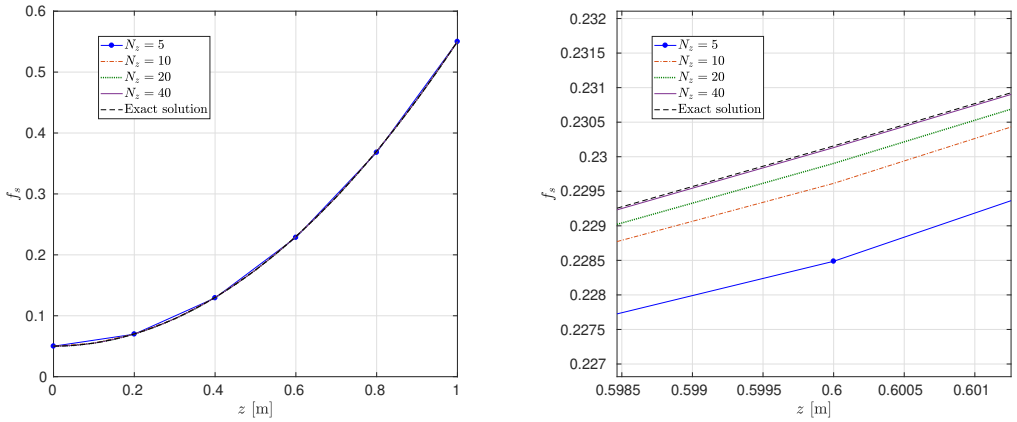


Figure 4.32 – Example 1: Exact solution and approximated solutions computed using the finite differences method at $t = 150$ [s], for various discretizations. Unzoomed (left), zoomed around $z = 0.6$ [m] (right).

For the same example, with the finite volumes approach, the first observation is that, Godunov and Engquist-Osher schemes give exactly the same results. In fact, the flux is no more convex, and the Godunov scheme is not more accurate than the Engquist-Osher scheme anymore. In Figure 4.33, we plot the relative error (as defined by (4.6)) for the finite volumes method along

with the first order convergence slope as well as the relative error for the finite differences method along with the second order convergence slope.

We can see from the figures and from Table 4.1 that the finite differences method is better than the finite volumes method in terms of accuracy. The error is not only smaller, but also, the order of convergence of the finite differences scheme is higher than that of the finite volumes method.

Table 4.1 – Example 1: - Errors for finite volumes and finite differences methods while varying the number of grid points.

$N_z = 1/\Delta z$	10	20	40	80
Error for finite volumes method	$10^{-5} \times 61.79$	$10^{-5} \times 22.98$	$10^{-5} \times 8.351$	$10^{-5} \times 2.99319$
Error for finite differences method	$10^{-5} \times 21.26$	$10^{-5} \times 5.66$	$10^{-5} \times 1.46$	$10^{-5} \times 0.37$

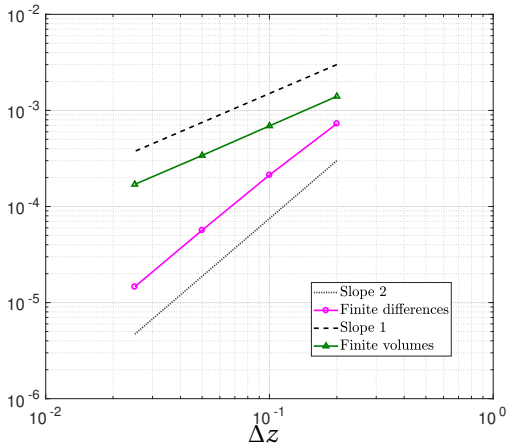


Figure 4.33 – Example 1: Finite volumes method: the scheme order is closer to 1. Finite differences method: the scheme order is closer to 2.

We now consider Example 2 and run a mesh convergence study while varying ω_1 and ω_2. We report the results in Table 4.2. Looking at the results in Table 4.2, we can see that the finite volumes method seems to fail to give any apparent order. This is true, essentially when the solution is oscillatory in both time and space. On the other hand, the finite differences method seem to perform very well: they still exhibit the order 2 for convergence, with considerably lower error values.

Table 4.2 – Example 2 - Errors for Finite volumes and Finite differences.

	Δz	Error for finite differences	Error for finite volumes
$\omega_1 = 1, \omega_2 = 1$	0.050000	0.000179	0.004255
	0.025000	0.000057	0.004065
	0.012500	0.000015	0.002108
	0.006250	0.000004	0.001604
$\omega_1 = 10, \omega_2 = 1$	0.050000	5.530749	4.459744
	0.025000	0.270610	0.261377
	0.012500	0.082205	0.151226
	0.006250	0.027722	0.138018
$\omega_1 = 1, \omega_2 = 10$	0.050000	0.014813	0.047236
	0.025000	0.005425	0.042571
	0.012500	0.002121	0.035003
	0.006250	0.000815	0.027735
$\omega_1 = 10, \omega_2 = 10$	0.025000	0.191291	1.057347
	0.012500	0.062620	0.787782
	0.006250	0.013769	0.43352
	0.003125	0.002156	0.42101

After testing the resuspension part alone, we want to do the same when we add the deposition effects. We look for $f_s \in [0, f_{s_{CR}}]$ solution of:

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)\Big(-k v_{stokes} - \gamma \frac{\partial f_s}{\partial z}\Big)\right) = 0 \tag{4.14}$$

where $\gamma > 0$. We want to compare the no-splitting method with finite volumes, and the splitting method with finite differences for the resuspension part (as described in Subsection 3.2.3) on this simplified one-dimensional case. We remind these methods briefly:

No splitting: Finite volumes method: We solve (4.14) with the finite volumes method (either Godunov or Engquist-Osher schemes).

Splitting: Finite differences method for the resuspension: We split the equation (4.14) into:

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)\Big(-k v_{stokes}\Big)\right) = 0 \tag{4.15}$$

and

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\Big(1 - \frac{f_s}{f_{s_{CR}}}\Big)\Big(-\gamma \frac{\partial f_s}{\partial z}\Big)\right) = 0 \tag{4.16}$$

We use the finite volumes method to solve (4.15) and the finite differences method to solve (4.16). However, since we want to see the effect of the splitting, we try an additional method for comparison purposes. This third method consists in splitting the equation (4.14) into (4.15) and (4.16), and using the finite volumes method to solve each equation separately. In order to

simplify the illustration of the results, we refer to these methods, respectively, by No-splitting, splitting FV-FD, and splitting FV-FV.

Example 1: time-dependent exact solution

We construct a source term g_1 such that a solution to

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(-k v_{stokes} - \gamma \frac{\partial f_s}{\partial z}\right)\right) = g_1(z,t),$$

is $f_{s_{exact}}(z,t) = t$. In this case, the solution is only a function of time. Thus, when we do not split the equation, we get the exact solution as a final result. We now want to see the effect of splitting the equation into (4.15) and (4.16). We add the source term to (4.16). In Figure 4.34, we can see that both methods give more or less the same error: the error curves are superimposed. Since the solution depends only on time, the error that we see is the splitting error.

Example 2: space-dependent exact solution

We construct a source term $g_2(z,t)$ such that a solution to

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(-k v_{stokes} - \gamma \frac{\partial f_s}{\partial z}\right)\right) = g_2(z,t),$$

is $f_{s_{exact}}(z,t) = z^2$. In Figure 4.35, we plot the L_2 errors for Example 2 for each method. We observe that using the finite differences method with a splitting scheme leads to a better solution that using the finite volumes method with a splitting scheme or without. The second order of the finite differences method in space is the reason behind this.

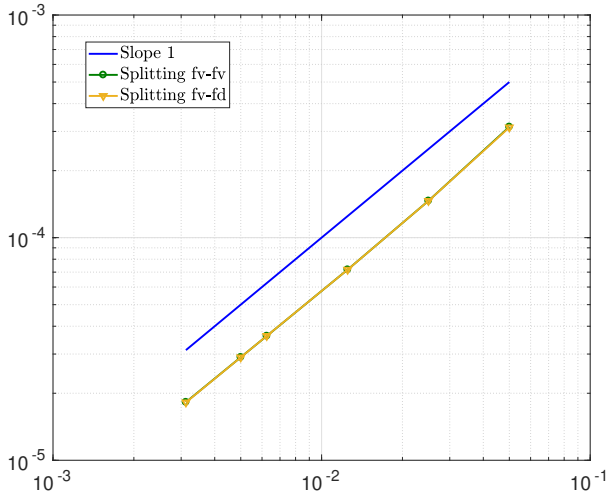


Figure 4.34 – Example 1: Errors for the splitting methods (FV-FD and FV-FV): the yellow and the green lines are superimposed.

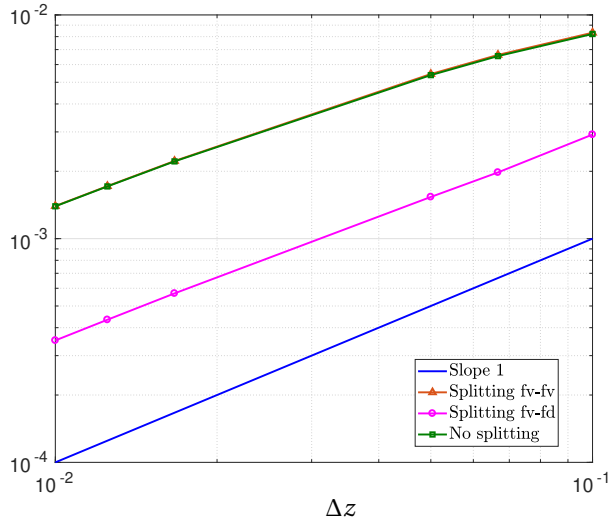


Figure 4.35 – Example 2: Errors for the No-splitting method (Finite volumes) and the splitting method (FV-FD and FV-FD). The green and the orange lines are superimposed.

Example 3

We construct a source term g_3 such that a solution to

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(-kv_{stokes}-\gamma\frac{\partial f_s}{\partial z}\right)\right) = g_3(z,t),$$

is $f_{s_{exact}}(z,t) = (z^2+t)/2$. In Figure 4.36, we illustrate the exact and the approximated solutions for different meshes.

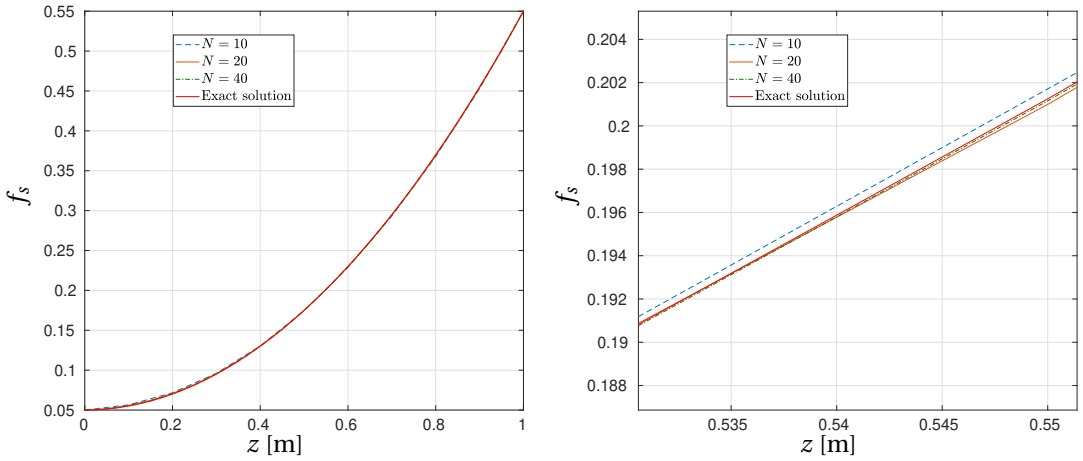


Figure 4.36 – Example 3: Numerical and exact solution when we solve the deposition with finite volumes, then the resuspension with finite differences: zoomed (right), non-zoomed (left). The source term is taken into account in the resuspension step.

In Figure 4.37, we plot the L^2 errors for Example 3 for each method. In fact, for the chosen discretization parameters, the splitting error has a big effect. However, even though the second order of the finite differences scheme is no longer observed, the splitting scheme with the finite differences method for the resuspension, is still better.

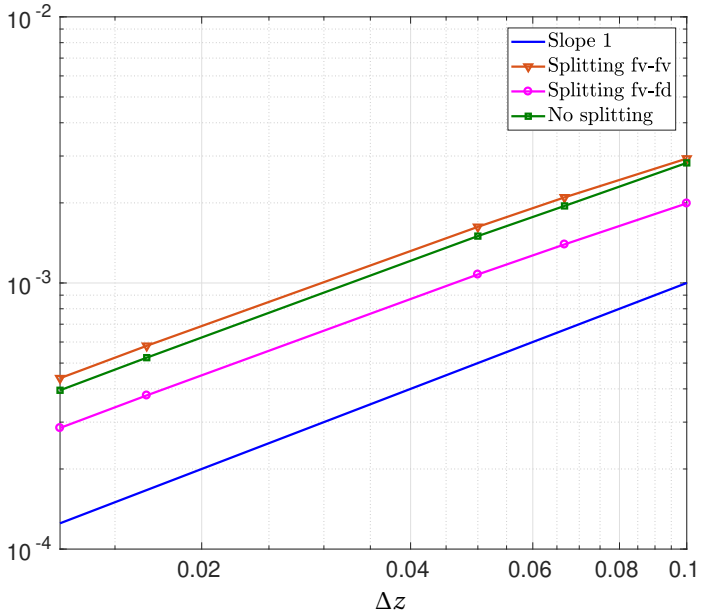


Figure 4.37 – Example 3: Errors for the no-splitting scheme, splitting scheme with the finite differences method for the resuspension (Splitting fv-fd) and the additional method (Splitting fv-fv).

Remark: Figure 4.38 shows that there is roughly no effect of the resolution order after the splitting, as well as of the equation where the source term is placed.

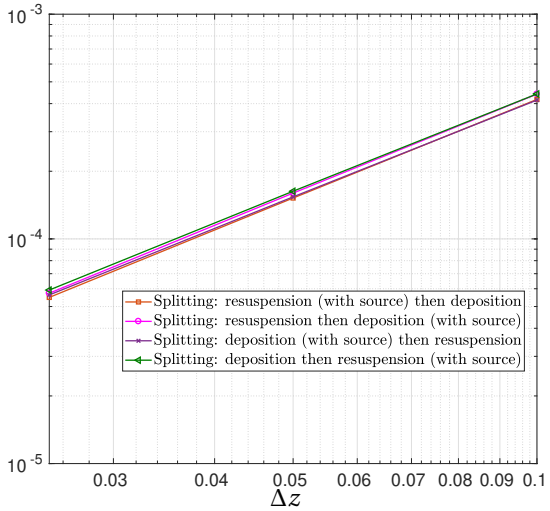


Figure 4.38 – Error comparison when we change the order of operators after the splitting, and when we change the placement of the source term g_3.

Example 4

We construct a source term $g_4(x,t)$ such that a solution to

$$\frac{\partial f_s}{\partial t} + \frac{\partial}{\partial z}\left(f_s\left(1-\frac{f_s}{f_{s_{CR}}}\right)\left(-kv_{stokes}-\gamma\frac{\partial f_s}{\partial z}\right)\right) = g_4(z,t),$$

is $f_{s_{exact}}(z,t) = \sin^2(\omega_1\pi z)\sin^2(\omega_2\pi t)$ where $\omega_1,\omega_2 > 0$. We vary ω_1 and ω_2 and display the results in Tables 4.3, 4.4 and 4.5.

Table 4.3 – Example 4: Errors for splitting and no splitting schemes for $\omega_1 = 1, \omega_2 = 1$.

Δz	No splitting	Splitting, FV-FV	Splitting, FV-FD
0.100000	0.000996	0.000913	0.000959
0.050000	0.000263	0.000353	0.000211
0.025000	0.000118	0.000144	0.000097
0.012500	0.000043	0.000062	0.000035
0.006250	0.000016	0.000023	0.000013

Table 4.4 – Example 4: Errors for splitting and no splitting schemes $\omega_1 = 1, \omega_2 = 10$.

Δz	No splitting	Splitting, FV-FV	Splitting FV-FD
0.100000	0.006029	0.007229	0.004215
0.050000	0.002270	0.002803	0.001123
0.025000	0.001011	0.001252	0.000344
0.012500	0.000398	0.000513	0.000128
0.006250	0.000152	0.000201	0.000049

Table 4.5 – Example 4: Errors for splitting and no splitting schemes $\omega_1 = 10, \omega_2 = 1$.

Δz	No splitting	Splitting, FV-FV	Splitting FV-FD
0.050000	0.212366	0.211758	0.850920
0.025000	0.006373	0.006475	0.006864
0.012500	0.001865	0.001978	0.001057
0.006250	0.000854	0.000915	0.000237

We observe that the splitting effect on the results is negligeable. Moreover, since the finite differences scheme is of order two in space, when the solution is more dependent on the space discretization, the order is roughly two, such is the case in Table 4.4.

4.2.3 Deposition and resuspension of polysterene particles in a still fluid

In this experiment, the goal is to qualitatively check the resuspension effect on a similar setup to that described in Subsection 4.1.2. We look for $f_s = f_s(z,t)$ by solving (4.14) while varying the resuspension constant γ. The stationary solution obtained in Subsection 4.1.2 is the initial condition that we consider here:

$$f_s(z,t=0) = \begin{cases} f_+, & z \geq z^* \\ f_-, & 0 \leq z < z^* \end{cases} \tag{4.17}$$

with $f_+ = 0$, $f_+ = 0.6$ and $z^* = 0.044$. We show the effect of the resuspension constant γ on the final solution in Figure 4.39. When $\gamma = 0$, only the deposition part is taken into account. The solution in this case remains the same over time. On the other hand, when the resuspension constant becomes large (in this case, $\gamma = 10^{-2}$), the final solution is homogeneous, i.e., all of the particles are in suspension.

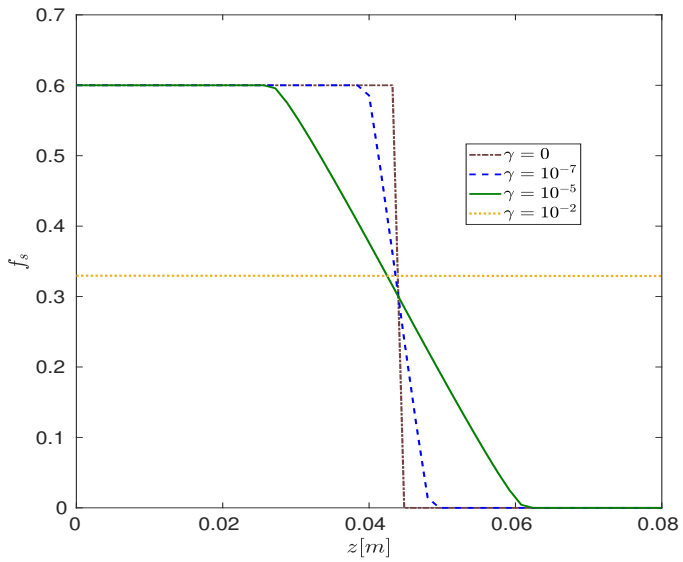


Figure 4.39 – Effect of resuspension constant on the stationary solution to the equation (4.14) at $t = 50$ [s].

After validating the method on simplified one dimensional cases, we now consider the one-dimensional case solution as a reference solution and want to validate the solution on the full 3D model where:

- The rest of equations are involved (i.e. Modified Stokes and convection equations.
- In the resuspension flux of the sedimentation equation, we take $K_r\left(\frac{\tau(f_s,\mathbf{v})-\tau_{CR}}{\tau_{CR}}\right)_+ = \gamma$.

We consider a rectangular domain of 0.08 [m] height and 0.01 [m] width and a tranversal depth

of 0.005 [m]. We run the experiment for the initial condition (4.17) and $\gamma = 10^{-5}$. The expected solution is stationary starting from $t = 50$ [s]:

$$f_s(t=50) = \begin{cases} f_s = 0, & z \geq z^+ \\ 0 < f_s < f_{s_{CR}}, & z^+ \leq z < z^- \\ f_s = f_{s_{CR}}, & 0 \leq z < z^- \end{cases}$$

with $z^- = 0.028$ and $z^+ = 0.06$. The reference solution, is visualized in Figure 4.40. The numerical solution, illustrated in Figure 4.41, shows the same solid fraction repartition and validates the model.

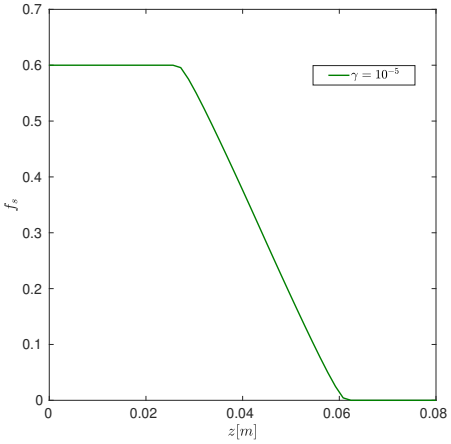


Figure 4.40 – Solid fraction repartition at $t = 50s$ for $\gamma = 10^{-5}$ using the finite difference method.

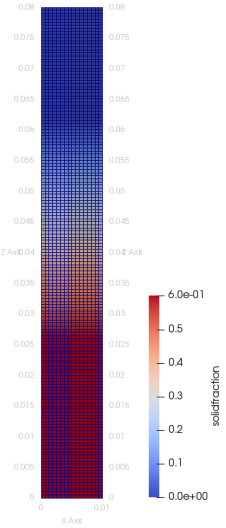


Figure 4.41 – Numerical solution: solid fraction snapshot at $t = 50$ [s] for $\gamma = 10^{-5}$ on the cells, using the finite differences method.

A comparison between methods: We consider the following initial condition:

$$f_s(z, t=0) = \begin{cases} f_+ & z \geq z^* \\ f_-, & 0 \leq z < z^* \end{cases} \tag{4.18}$$

with $f_+ = 0$, $f_- = 0.6$ and $z^* = 0.003$. The solution in this case is stationary starting from $t = 10$ [s].

Let us inverstigate the effect of the interpolation FE/Grid-cells when using the finite elements to solve the resuspension equation. When we solve the resuspension equation using the finite elements method, we first need to interpolate the solid fraction values on the finite elements, and then we have to interpolate the computed solution back on the structured grid cells. Thus, we need to evaluate the interpolation effects on this method. For $\gamma = 0$ and the initial condition (4.18), we show the results in Figure 4.42. The solution for this case is

expected to remain constant over time. Furthermore, in the same figure, we show the effect of finite-elements/grid-cells ratio H/h on the solution. First, we vary the ratio from $H/h \simeq 3$ to $H/h \simeq 5$, then $H/h \simeq 10$ and observe the difference. Figure 4.42 (right), shows that there is numerical diffusion caused by these interpolations no matter what the ratio is. The questions we would like to answer are: how much this diffusion penalizes the desired results, and if there is a way we can reduce it.

In Figure 4.43, we show the stationary solution when using the finite elements method for all three ratios and $\gamma = 10^{-5}$. We observe that, even though the numerical diffusion still exists, the induced error is negligible due to the higher resuspension γ.

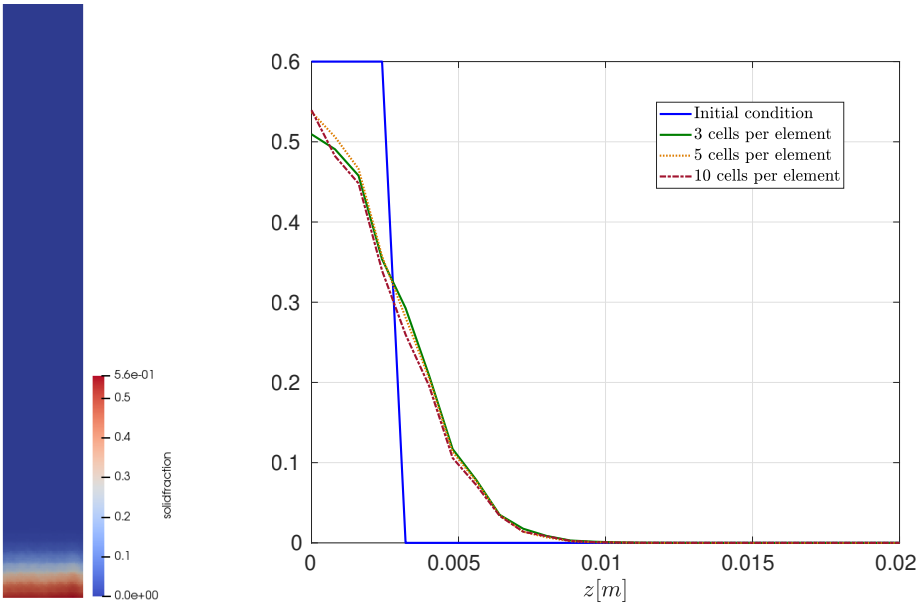


Figure 4.42 – Finite elements for resuspension, $\gamma = 0$: effect of the interpolations Finite-elements/Grid-cells on the solution for various ratios H/h (3 cells per element means $H/h = 3$). Left: snapshot of the solution for $H/h = 3$, right: solid fraction values along the vertical axis, at position $(x, y) = (0.005, 0)$ at $t = 10$ [s] for various ratios H/h.

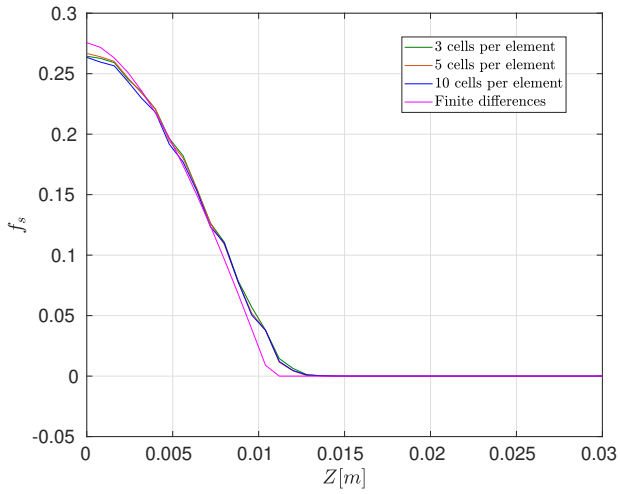
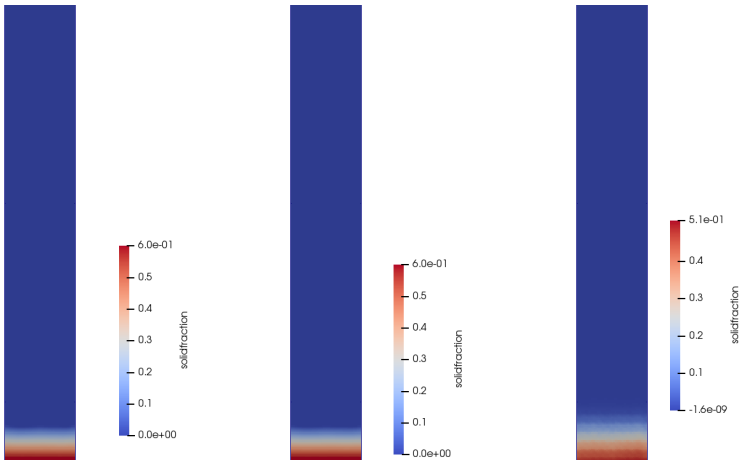


Figure 4.43 – Finite elements for resuspension, $\gamma = 10^{-5}$: solid fraction values along the vertical axis, at position (4.18) at $t = 10$ [s] and with different ratios H/h (3 cells per element means $H/h = 3$).

In the sequel, we take $H \approx 5h$. We compare the methods while varying the resuspension value, γ. For $\gamma = 10^{-6}$, $\gamma = 10^{-5}$ and $\gamma = 10^{-4}$, we show the snapshots of the solid fraction at $t = 10$ [s] for the three methods in Figure 4.44, Figure 4.46 and Figure 4.48, respectively. For all methods, at $t = 10$ [s] , we plot the sediment profile over the vertical axis at position $(x, y) = (0.005, 0)$ for $\gamma = 10^{-6}$, $\gamma = 10^{-5}$ and $\gamma = 10^{-4}$ respectively in Figure 4.45 , Figure 4.47 and Figure 4.49.



(a) (b) (c)

Figure 4.44 – Snaphots of the solution of (4.14) with the initial condition (4.18) at $t = 10$ [s] using: (a) the finite differences method, (b) the finite volumes method and (c) the finite elements ($\gamma = 10^{-6}$).

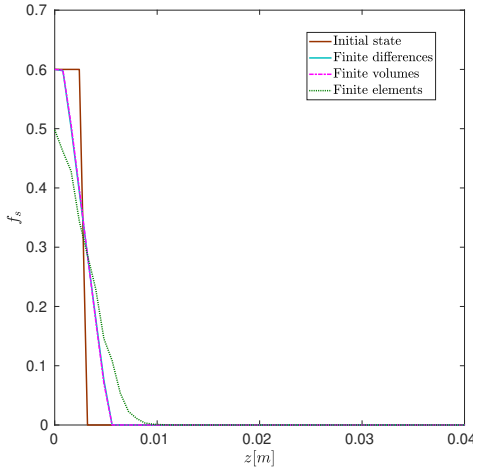
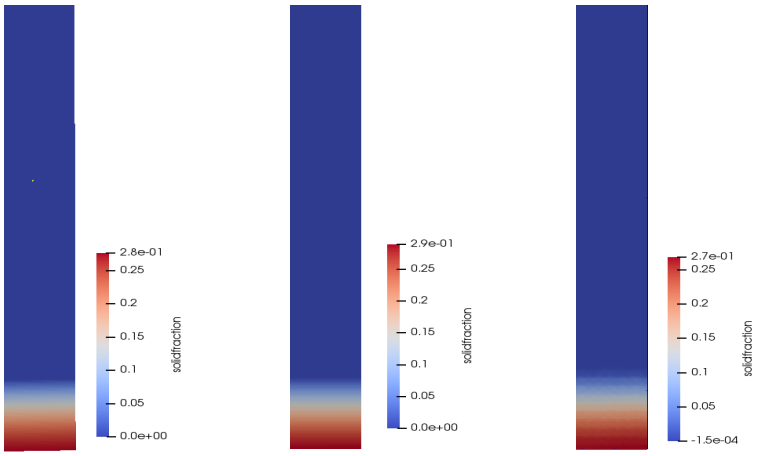


Figure 4.45 – Solid fraction at $t = 10$ [s] along the vertical axis at position $(x, y) = (0.005, 0)$ with the finite differences method, the finite volumes method and the finite elements method ($\gamma = 10^{-6}$).



(a) (b) (c)

Figure 4.46 – Snaphots of the solution of (4.14) with the initial condition (4.18) at $t = 10$ [s] using: (a) the finite differences method, (b) the finite volumes method and (c) the finite elements ($\gamma = 10^{-5}$).

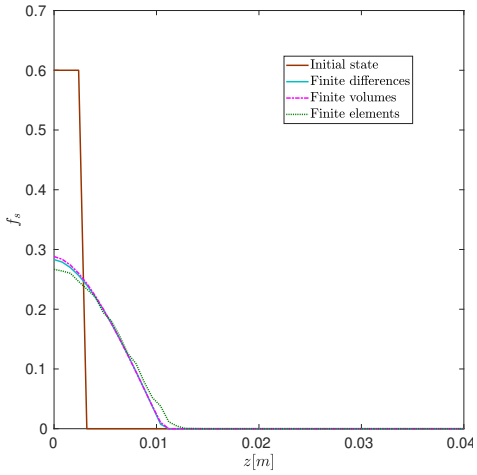
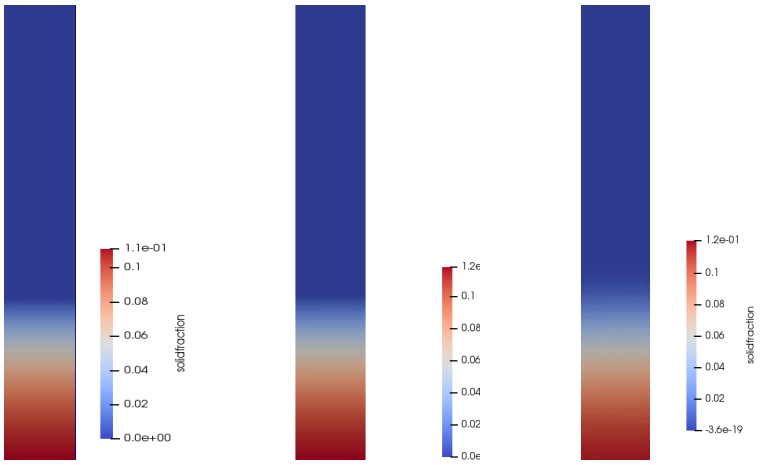


Figure 4.47 – Solid fraction at $t = 10$ [s] along the vertical axis at position $(x, y) = (0.005, 0)$ with the finite differences method, the finite volumes method and the finite elements method ($\gamma = 10^{-5}$).



(a) (b) (c)

Figure 4.48 – Snaphots of the solution of (4.14) with the initial condition (4.18) at $t = 10$ [s] using: (a) the finite differences method, (b) the finite volumes method and (c) the finite elements ($\gamma = 10^{-4}$).

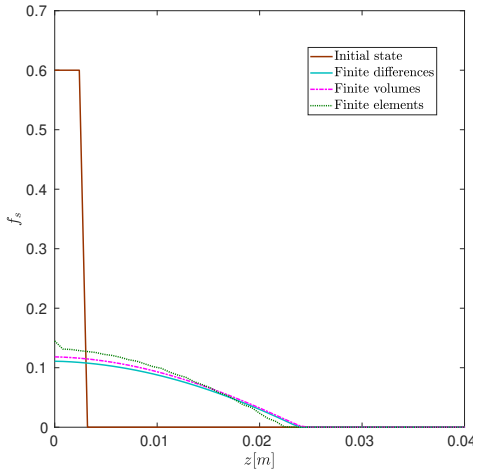


Figure 4.49 – Solid fraction at $t = 10$ [s] along the vertical axis at position $(x, y) = (0.005, 0)$ with the finite differences method, the finite volumes method and the finite elements method ($\gamma = 10^{-4}$).

While the finite elements method for the resuspension is not restrictive on the time step compared to finite differences or finite volumes, it remains very numerically diffusive. However, the higher the resuspension parameter is, the less apparent the diffusion due to the interpolation is. For instance, in Figure 4.45, where $\gamma = 10^{-6}$, the finite element method gives a solution that is worse than both the finite difference and the finite volume solutions. This is not the case when $\gamma = 10^{-4}$ as shows Figure 4.49. This means that, for experiments where we have enough physical diffusion (resuspension), the finite elements can still be suitable. This will be further studied in the next sections.

4.2.4 Sediment flushing

In Subsection 4.1.5, we have illustrated the limits of the physical model for sedimentation dynamics with deposition effects only. For the flushing experiment mentioned and described in [153], resuspension effects were inferred to be necessary. Thus, we revisit the results here, after including the resuspension effects to our model. The experimental setup is the same as that described in Subsection 4.1.5. Except for the transversal dimension that is about twice as small as that considered in Subsection 4.1.5. We first evaluate the convergence of the numerical method, when the discretization parameters (time step and mesh size) decrease. To do so, we run the experiment with a final time $T = 3$ [s]. We consider three discretizations:

1. **Very coarse mesh:** 2991 elements, 1144 nodes, 23760 cells ($H \simeq 0.07$ [m], $h \simeq 0.015$ [m]) and $\Delta t = 0.009$ [s]
2. **Coarse mesh:** 6684 elements, 2446 nodes, 51480 cells ($H \simeq 0.045$ [m], $h \simeq 0.01$ [m]) and $\Delta t = 0.006$ [s]
3. **Intermediate mesh:** 26079 elements, 9124 nodes, 205920 cells ($H \simeq 0.025$ [m], $h \simeq$

0.005 [m])and $\Delta t = 0.003$ [s]

4. **Fine mesh:** 45600 elements, 15772 nodes, 369600 cells ($H \simeq 0.02$ [m], $h \simeq 0.003$ [m]) and $\Delta t = 0.0022$ [s]

Figure 4.50 illustrates the numerical results obtained with and without resuspension effects for the coarse, intermediate and the fine meshes. Figure 4.51 and Figure 4.52 illustrate the snapshots of the sediment profile at $t = 3$ [s] for the coarse, the intermediate and the fine mesh, with resuspension ($K_r \neq 0$) and without resuspension ($K_r = 0$), respectively. These figures show the convergence of the method. Furthermore, from Figures 4.51 and 4.52 we can already observe that the sediment profile with and without resuspension effects are different: resuspension effects promote the flushing of the sediments.

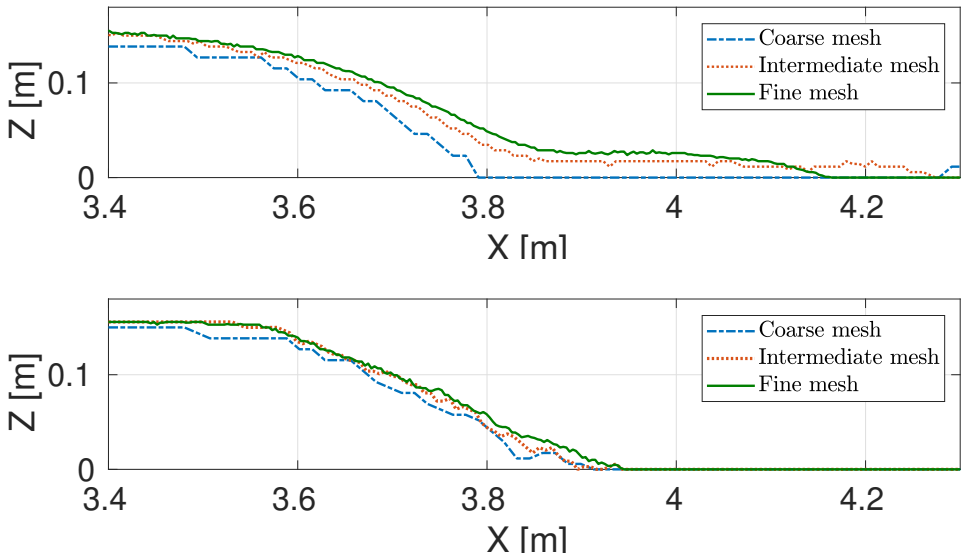
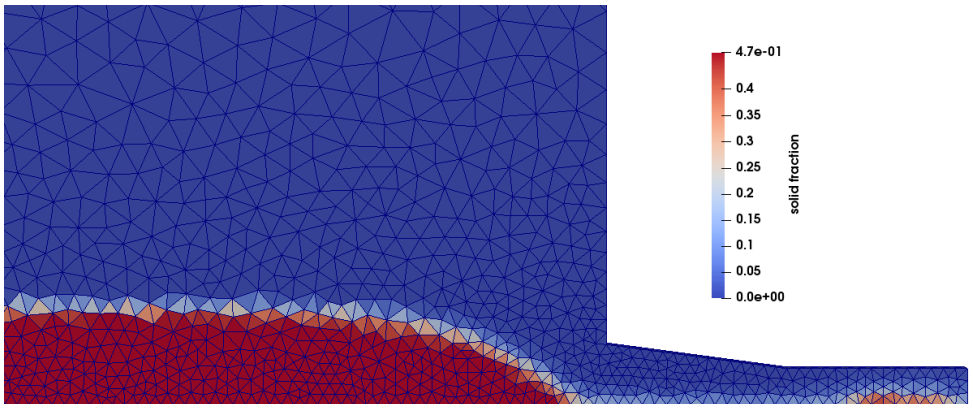
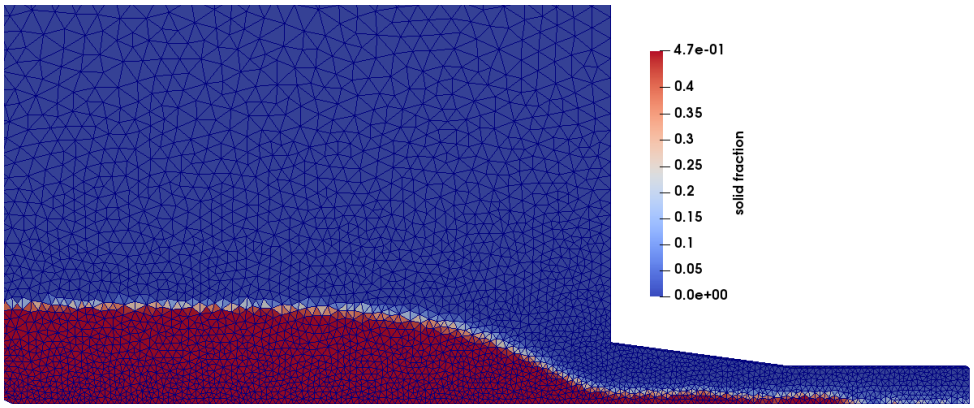


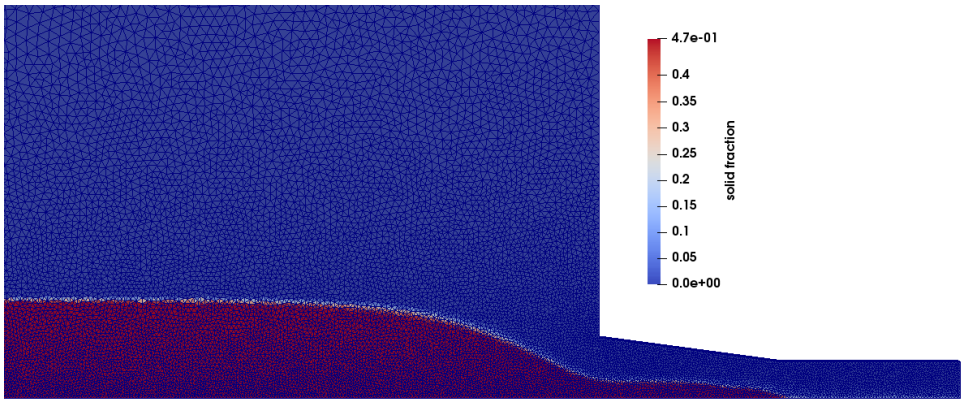
Figure 4.50 – Sediment flusing. Mesh convergence at $t = 3$ [s]. Top: without resuspension effects ($K_r = 0$). Bottom: with resuspension effects ($K_r \neq 0$), using the finite differences method for the resuspension.



(a)

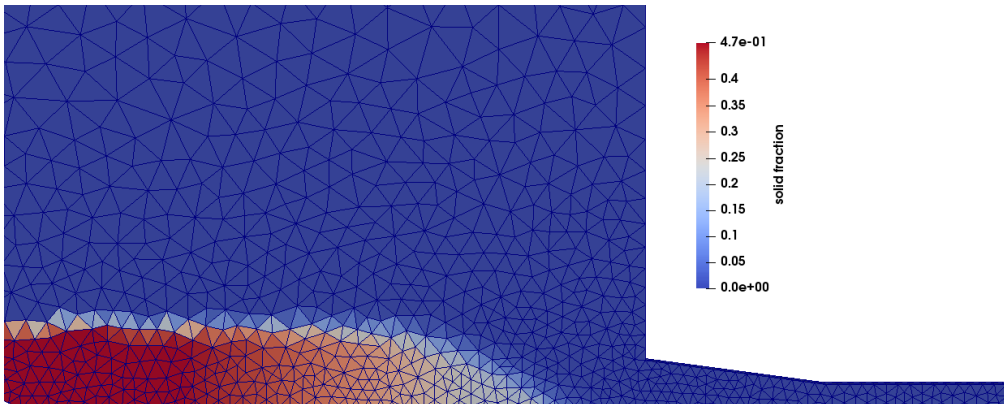


(b)

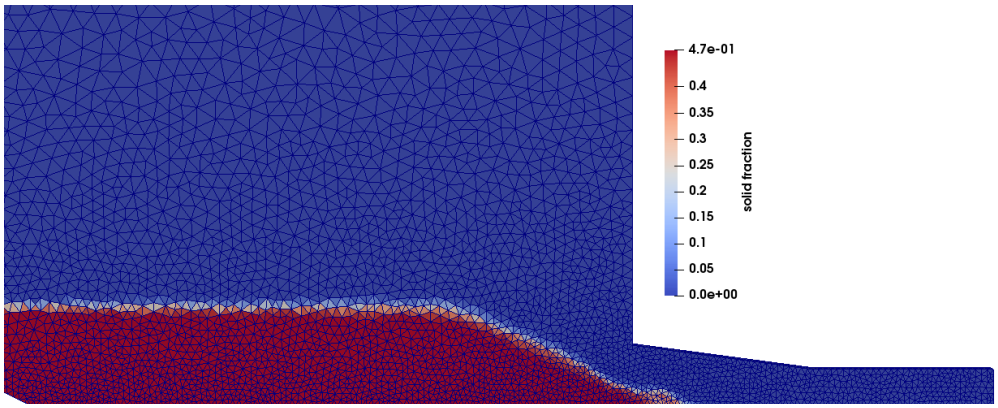


(c)

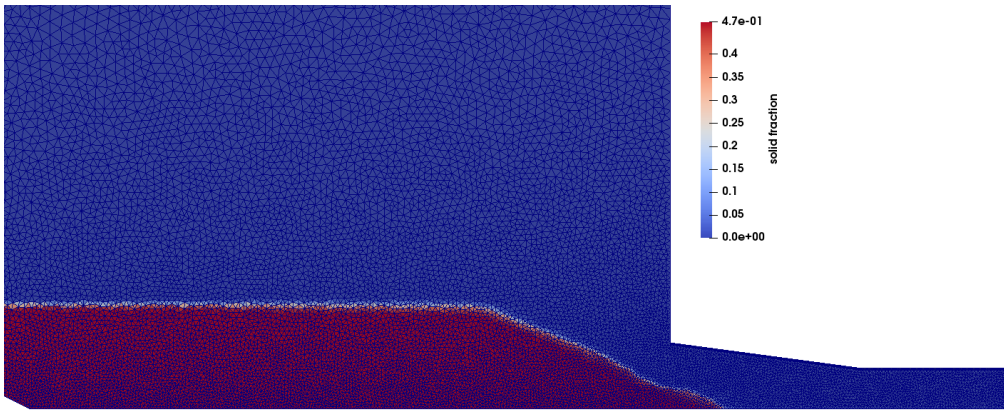
Figure 4.51 – Sediment flushing. Snaphots of sediment profite at $t = 3$ [s] for (a) coarse mesh (b) intermediate mesh (c) fine mesh, when $K_r = 0$ (no resuspension).



(a)



(b)



(c)

Figure 4.52 – Sediment flushing. Snaphots of sediment profite at $t = 3$ [s] for (a) coarse mesh (b) intermediate mesh (c) fine mesh, when $K_r \neq 0$ (with resuspension).

For the rest of the computations, we consider the so-called intermediate mesh. Moreover, unless mentioned otherwise, we rely on the splitting method with the finite differences approach, described in Subsection 3.2.3 for the sedimentation equation resolution. First, we set the resuspension constant $K_r = 0$ and the sediment cohesion parameter $f_{s_{CO}} = 0.4699$. We vary the penalization parameter ε that appears in (2.6). By setting the resuspension constant to zero, we are only allowing the deposition to occur. As explained in [121], this was not enough to acquire the desired bedload in comparison to the experiment. By omiting the resuspension effects first, we want to find another optimal penalization parameter, ε so that we can adjust the resuspension parameters (K_r and τ_{CR}) in order to match the experimental results. Snapshots of the obtained results with several values of ε are illustrated in Figure 4.53. For $\varepsilon = 10^{-9}$ (top figure), the penalization in the sand is too low that the bedload level decreases very fast over the 40 seconds of the experiment. When ε decreases, numerical results are getting in better agreement with experimental ones. With $\varepsilon = 10^{-11}$ (bottom figure), the penalization is high enough to keep the sand stable in the necessary regions. Thus, in the following tests, we consider $\varepsilon = 10^{-11}$.

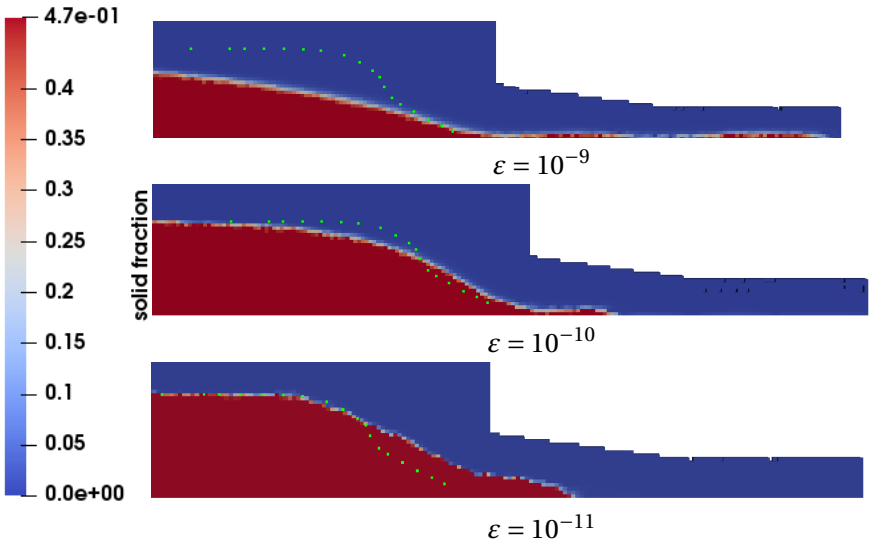


Figure 4.53 – Sediment flushing. Solution at $t = 40$ [s] for $K_r = 0$ and various penalization values (ε). Plotted dots represent the experimental results.

Setting $\varepsilon = 10^{-11}$, we introduce the resuspension effects by taking $K_r \neq 0$. For $K_r = 10^{-1}$ and $\tau_{CR} = 0.05$ [N/m^2], we display the snapshots of the shear stress on the structured cells for various times t in Figure 4.55. It shows that the shear stress appears in the slope where the sediments are expected to be flushed over time. Figure 4.55 shows the sediment profile with $K_r = 10^{-2}$ and various τ_{CR}. It shows that the optimal value for τ_{CR} is between 0.01 and 0.05 [N/m^2]. In Figure 4.56, we display the resuspension velocity in [m/s], computed as $A(f_s, \mathbf{v})\nabla f_s$ at various times. In Figure 4.57 we show the sediment bedload at $t = 40$ [s], with

$\tau_{CR} = 0.05$ [N/m^2] and $K_r = 10^{-3}$ (top figure), $K_r = 10^{-2}$ (middle figure) and $K_r = 10^{-1}$ (bottom figure). We observe that, with $K_r = 10^{-3}$, we do not have enough resuspension effects and the numerical results do not match the experiment. The results with $K_r = 10^{-2}$ and $K_r = 10^{-1}$ are very similar and close to the experimental results, at the stationary state. Figure 4.58 illustrates snapshots of the numerical solution (sediment bedload) and the experimental results over time for $K_r = 10^{-2}$ and $\tau_{CR} = 0.05$ [N/m^2]. It shows that our numerical results compare well to the experimental ones over time, and not just in the stationary state. Figure 4.59 compares the numerical solution obtained with and without smoothing. They both show that the results with smoothing are very similar to those without smoothing.

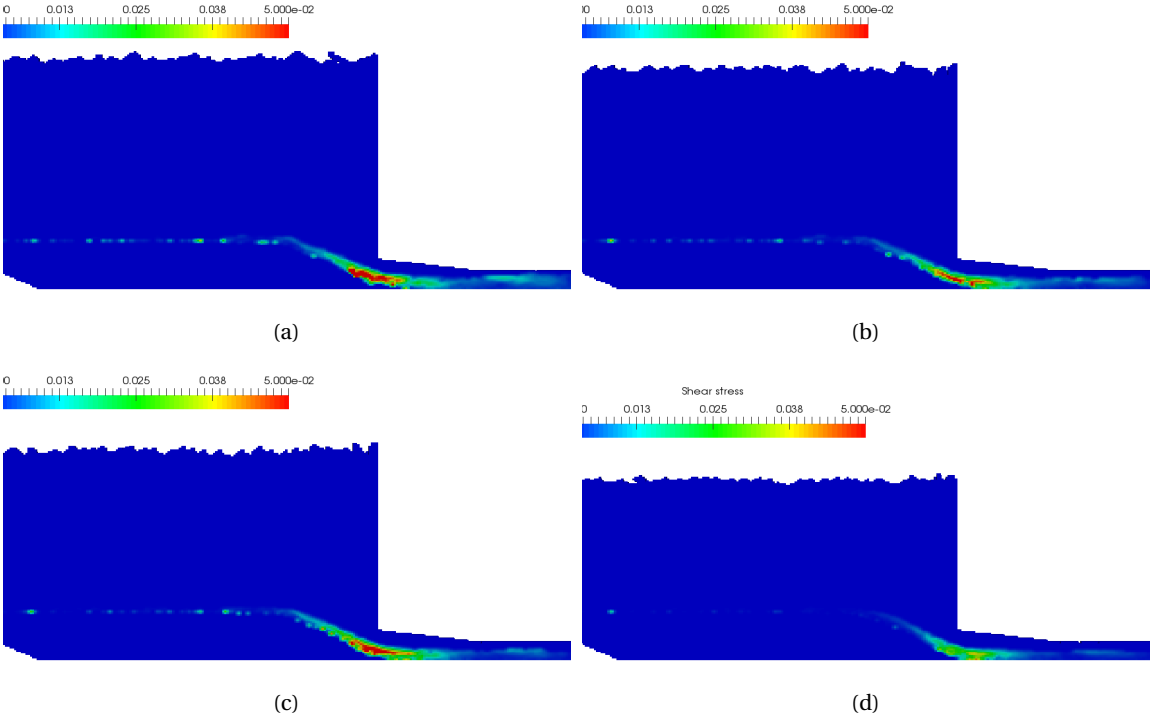


Figure 4.54 – Sediment flushing. Snaphots of the shear stress at: (a) $t = 2.5$ [s], (b) $t = 5$ [s], (c) $t = 10$ [s], (d) $t = 20$ [s].

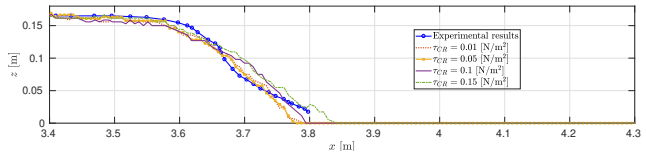


Figure 4.55 – Sediment flushing. Sediment profile at $t = 40$ [s], for $K_r = 10^{-2}$ and various values of τ_{CR} [N/m^2].

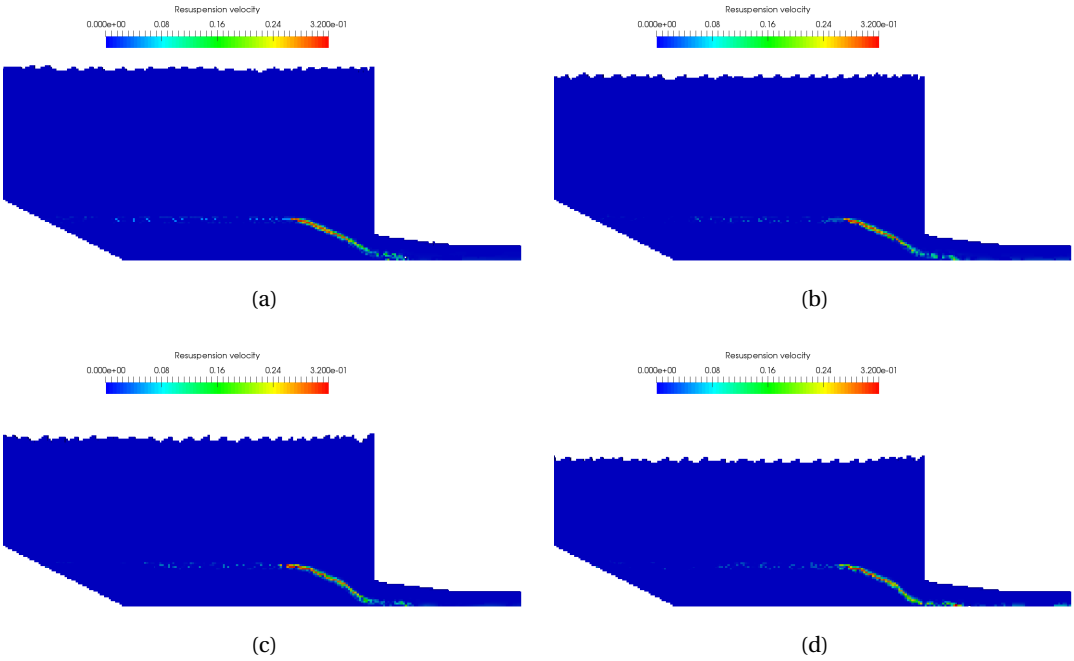


Figure 4.56 – Sediment flushing. Snaphots of the resuspension velocity, given by $A(f_s, \mathbf{v})\nabla f_s$, at: (a) $t = 3$ [s], (b) $t = 6$ [s], (c) $t = 12$ [s], (d) $t = 20$ [s].

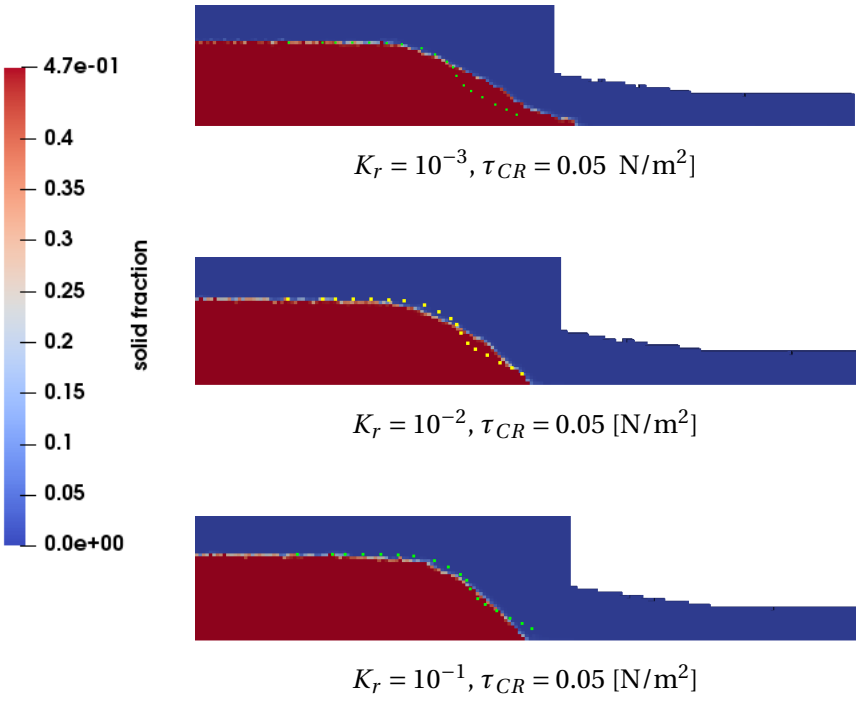


Figure 4.57 – Sediment flushing. Sediment profile at $t = 40$ [s] for $K_r \neq 0$, $\varepsilon = 10^{-11}$ and various resuspension parameters. Plotted dots represent the experimental results [153].

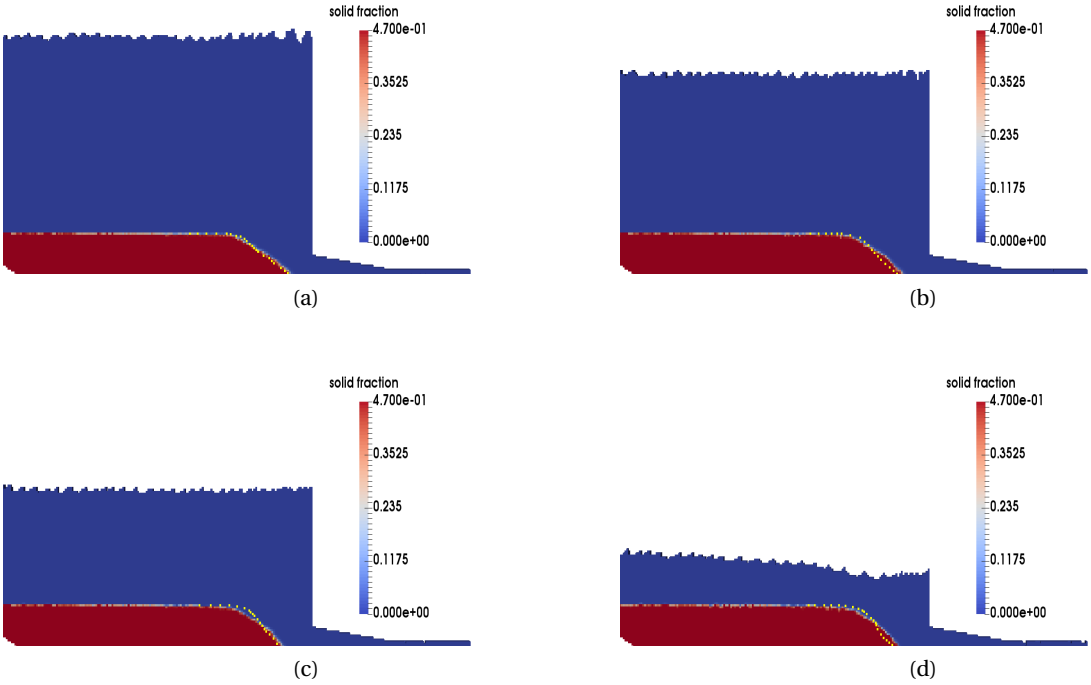


Figure 4.58 – Sediment flushing. Snaphots of the sediment profile at **(a)** $t = 10$ [s], **(b)** $t = 20$ [s], **(c)** $t = 30$ [s], **(d)** $t = 40$ [s] with $K_r = 10^{-2}$ and $\tau_{CR} = 0.05$ [N/m^2]. Plotted dots are the experimental results.

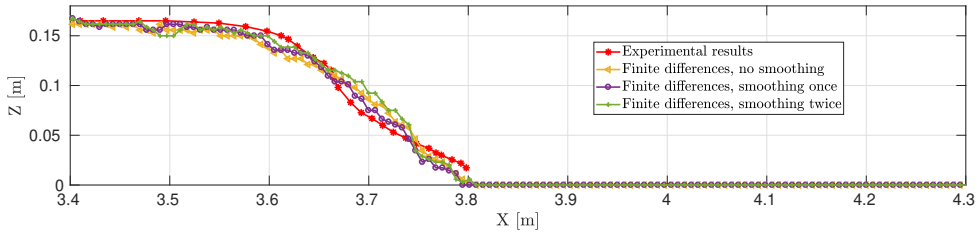


Figure 4.59 – Sediment flushing. Numerical solution (sediment bedload) at $t = 40$ [s]: Shear stress smoothing effect.

Once calibrating the parameters with an appropriate fit with the experimental results, we conduct an error analysis: we run the simulation for $K_r = 10^{-2}$ and $\tau_{CR} = 0.05$ [N/m^2] on the meshes presented previously. Figure 4.60 displays the sediment profile for $t = 10$ [s] obtained in the experiment as well as the numerical solution for all used meshes and shows, qualitatively, the convergence of the method.

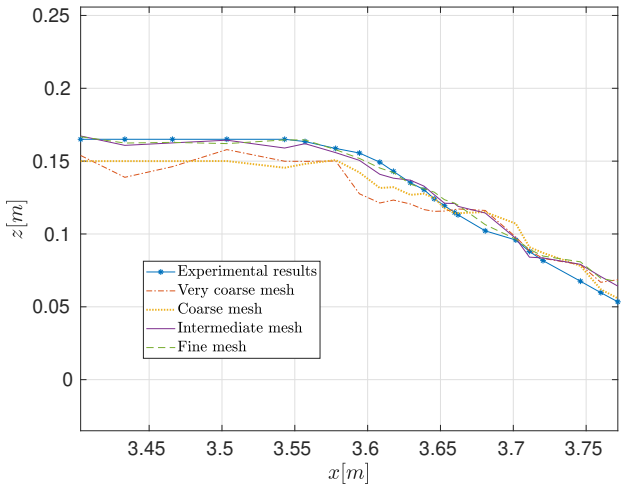


Figure 4.60 – Sediment flushing. At $t = 10$ [s]: Sediment bedload for the very coarse mesh, the coarse mesh, the intermediate mesh and the fine mesh in comparison with the experimental bedload profile.

We want to compute the error for the sediment profile between the experimental and the numerical solutions for each mesh. Let us denote z_{exp} the experimental bedload height. In fact, z_{exp} represents the experimental data points (assuming they are numbered from 1 to N): $(x_1, z_{exp_1}), \ldots, (x_N, z_{exp_N})$ where $x_1 < \cdots < x_N \in I$ where $I = [3.4, 3.85]$. On the other hand, let us denote by z_h the numerical bedload height. The numerical bedload is postprocessed such that, for each position x (defined by the cell positions on the horizonal axis (Ox)), we associate a vertical position of the bedload giving a piecewise constant approximation. In order to compute the error, we first interpolate the numerical solution z_h on the $x_1, \ldots, x_N$ such that the interpolated solution z_h is given by the points $(x_1, z_h), \ldots, (x_N, z_{hN})$. We compute the L^2 error between the numerical solution and the experimental solution:

$$\|z_h - z_{exp}\|_{L^2(I)} = \left(\sum_{k=1}^{N-1} (x_{k+1} - x_k) \frac{\left(z_{hk+1} - z_{exp_{k+1}}\right)^2 + \left(z_{hk} - z_{exp_k}\right)^2}{2} \right)^{1/2}. \tag{4.19}$$

Furthermore, we denote by z_{ref} the reference solution, which is the numerical solution obtained on the fine mesh. Denote by h_1, h_2 and h_3 the cell size for the very coarse mesh, the coarse mesh and the intermediate mesh, respectively. We want to compute the L_2 error between the reference solution and the solution obtained on each mesh i for $i = 1, 2, 3$. The structured grid sizes corresponding to these meshes are $h_1 = 0.02027$, $h_2 = 0.013515$, $h_3 = 0.00675757$ respectively. For instance, to compute the error between the numerical solution on the very coarse mesh and the reference solution, we first interpolate z_{ref} on the very coarse

mesh, to obtain Z_{ref}. Then we use the following formula:

$$\|z_h - Z_{ref}\|_{L^2(I)} = \left(\sum_{k=1}^{N_x} h_1 \frac{\left(z_{h\,k+1} - Z_{ref\,k+1}\right)^2 + \left(z_{h_1\,k} - Z_{ref\,k}\right)^2}{2} \right)^{1/2}, \tag{4.20}$$

where N_x is the number of the horizontal grid points on the very coarse mesh. Figure 4.61 shows the L^2 error between the experiment and the numerical solutions (computed with (4.19)) while Figure 4.62 shows the L^2 error between the reference solution and the rest of the numerical solutions (computed with (4.20) for the very coarse mesh and similarly for the other meshes) . We observe that our numerical scheme converges with an order that is approximately equal to one when the discretization parameters go to zero.

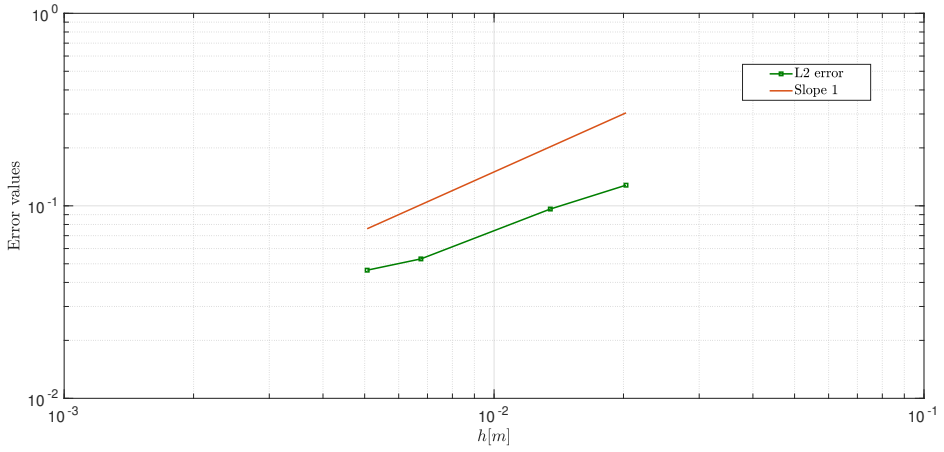


Figure 4.61 – Sediment flushing. L^2 error (computed as (4.19)) between the experiment and the numerical sediment heights for various meshes at $t = 10$ [s].

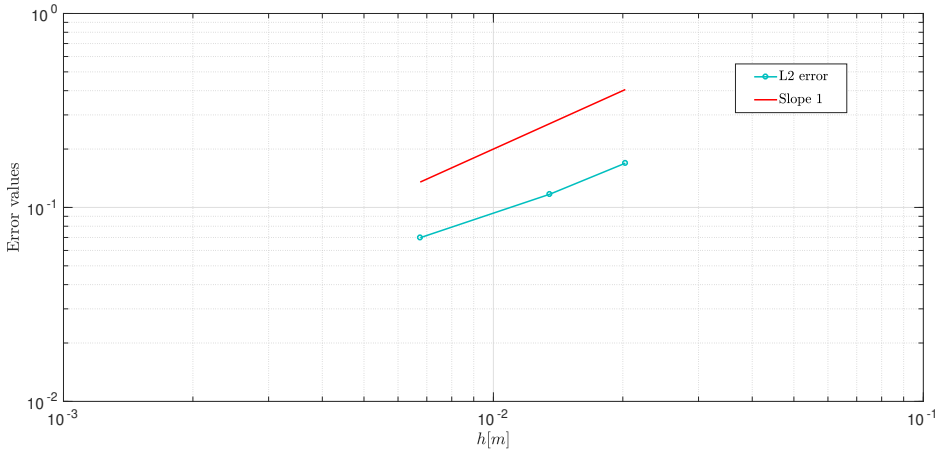


Figure 4.62 – Sediment flushing. L^2 error (computed as (4.20)) between the reference solution and the solutions for various meshes at $t = 10$ [s].

After comparing our numerical solution when we use the finite differences scheme for the resuspension with experimental results, we want to check the obtained results when we use the finite volumes approach and then the finite elements approach. To do so, we run the same experiment (using the same resuspension parameters $K_r = 10^{-2}$ and $\tau_{CR} = 0.05$ $[N/m^2]$) for each method. We plot the bedload profile obtained with the finite differences method and the finite volumes method in Figure 4.63, where the methods show very similar solutions.

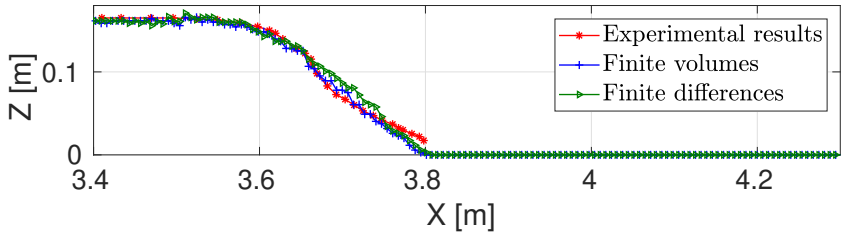


Figure 4.63 – Sediment flushing. At $t = 40$ [s]: snaphot of the sediment profile when we smooth the shear once (left), plot of the sediment profile with and without smoothing of the shear stress (right)

In Table 4.6, for each of the previously defined meshes, we present the average CPU time (in [s]) per time step Δt for this experiment, for each operator: the Stokes operator, the deposition/resuspension operator, and the advection operator.

Table 4.6 – Sediment flushing. Average CPU time per time step in [s] for each operator, for the coarse mesh($\Delta t = 0.006$ [s], intermediate mesh ($\Delta t = 0.003$ [s]) and fine mesh ($\Delta t = 0.0022$ [s]) all defined earlier. The results are given for the chosen parameters $\varepsilon = 10^{-11}$, $K_r = 10^{-2}$ and $\tau_{CR} = 0.05$ [N/m^2] and over a computation time of 42 [s].

	Stokes step	Deposition + resuspension step		Advection step	
		Deposition	Resuspension	Advection	Decomp + SLIC
Coarse	0.38	0.06	0.42	0.17	0.11
Intermediate	5.2	0.94	6.32	2.66	1.57
Fine	25.8	4.65	31.2	13.2	7.85

We can observe that the resuspension is time costly. The reason behind this is the strict stability condition given by (3.31). Thus, substeps have to be taken in order to solve the resuspension step: in this case, for each time step, 5 susbsteps are used to solve the resuspension, in average.
Remark: On this particular experiment, using the finite elements method instead of the finite differences to solve the resuspension equation (4.16) was proven to be very diffusive as Figure 4.64 shows. The diffusion is amplified by the fact that, for this particular experiment, because of the high values of the velocity (especially around the outlet), a very small time step is used (to ensure a small CFL number).

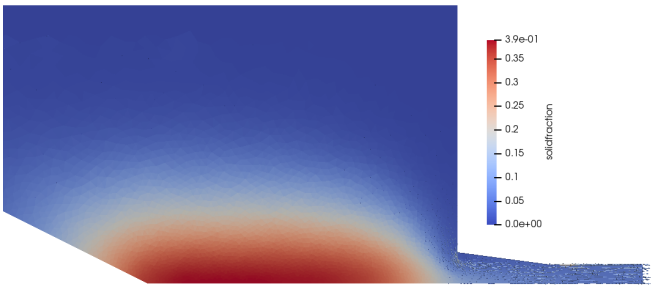


Figure 4.64 – Sediment flushing. Snapshot of the bedload profile at $t = 4$ [s] on the intermediate mesh, when we use the finite elements method to solve the resuspension. Numerical diffusion has a very large effect.

4.2.5 Wall-jet scouring

Now we consider a scouring experiment. This experiment, described in [156], is performed in a glass-walled horizontal flume of 0.74 [m] width and 0.2 [m] depth and a transversal dimension equal to 0.02 [m]. In the experimental setup, a rigid plate is installed in the flume to create an artificial reservoir on the left side of the setup. The water height in the reservoir is equal to 0.15 [m]. Between the liquid reservoir and the rest of the domain, is a gate of height 0.05 [m].

Opening the gate creates a bottom outlet for the reservoir. A layer of non-cohesive sand particles of diameter $d_* = 0.85 \times 10^{-3}$ [m] is laid downstream on the right side of the setup. The dry density of the sand particles is $\rho_s = 2650$ [kg m^{-3}], while the bulk density is $\rho_l = 1750$ [kg m^{-3}]. At $t = 0$, the gate is opened, and a scouring process is observed. We illustrate the computational domain in Figure 4.65. Free outflow conditions are applied on the top right part of the domain. Slip boundary conditions are applied on the lateral surfaces, and at the top surface, while no-slip boundary conditions are applied on the surfaces in contact with sediment. After about 3.5 [s], the formation of a scour hole and a ridge becomes stationary.

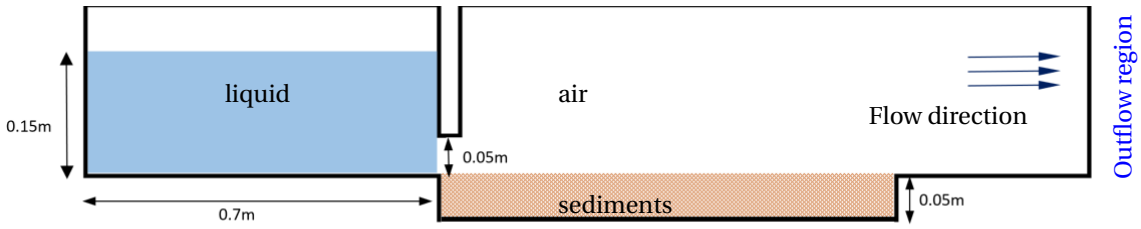


Figure 4.65 – Wall-jet scouring. Sketch of the geometrical domain and numerical setup (similar as in [156]).

We compare the experimental results and the numerical results over time by comparing the sand profiles and the water levels as well as the evolution of the scour hole diameter, illustrated in Figure 4.66, over time. For the sedimentation equation, we rely on the splitting method with the finite differences method for the resuspension.

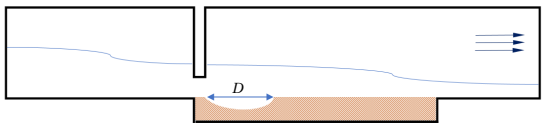


Figure 4.66 – Wall-jet scouring. Illustration of the scour hole diameter D.

First, we evaluate the convergence of the numerical method when the discretization parameters decrease (time step and mesh size). We take $K_r = 10^{-3}$ and $\tau_{CR} = 0.03$ [N/m^2]. We consider three discretizations:

(i) **Coarse mesh:** 6750 elements, 2442 nodes, 19500 cells ($H \simeq 0.037$, $h \simeq 0.0043$) and $\Delta t = 0.05$

(ii) **Intermediate mesh:** 25956 elements, 9028 nodes, 81900 cells ($H \simeq 0.02$, $h \simeq 0.0025$) and $\Delta t = 0.025$

(iii) **Fine mesh:** 102888 elements, 35040 nodes, 319800 cells ($H \simeq 0.011$, $h \simeq 0.0014$) and $\Delta t = 0.0125$

Figure 4.67 illustrates the sediment and the water profiles for each mesh at $t = 1$ [s] and shows the convergence of the method. The convergence is also observed for larger times. Figure 4.68 shows convergence results through the snapshots of the solution at 1.5 [s] for these meshes.

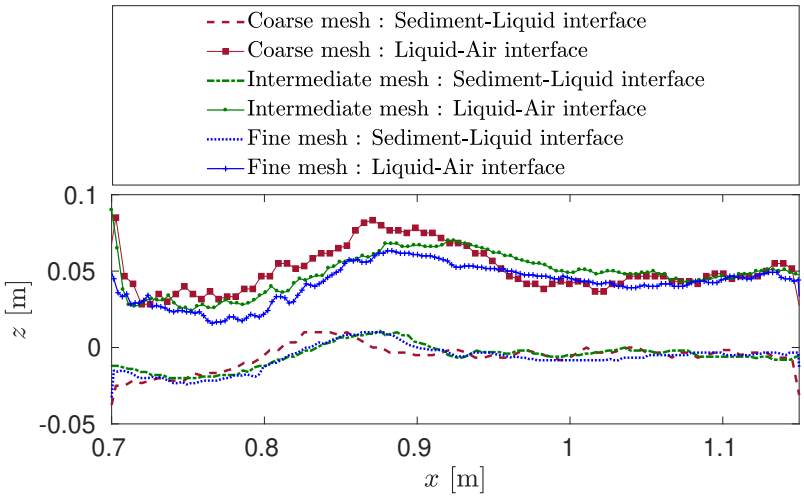
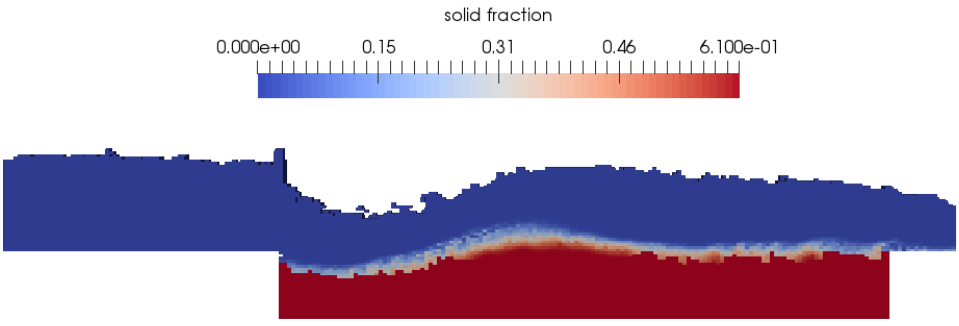
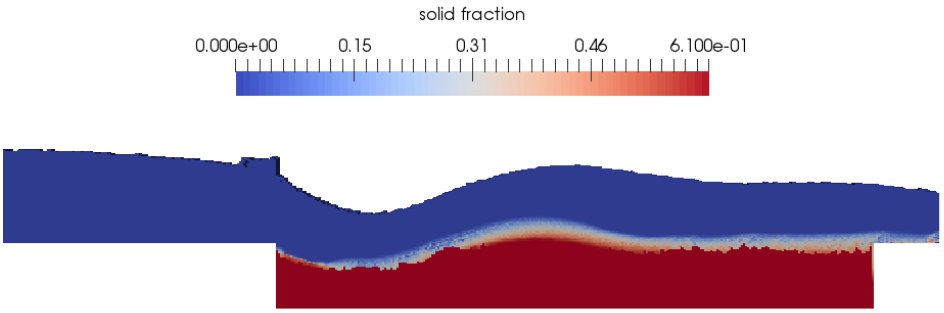


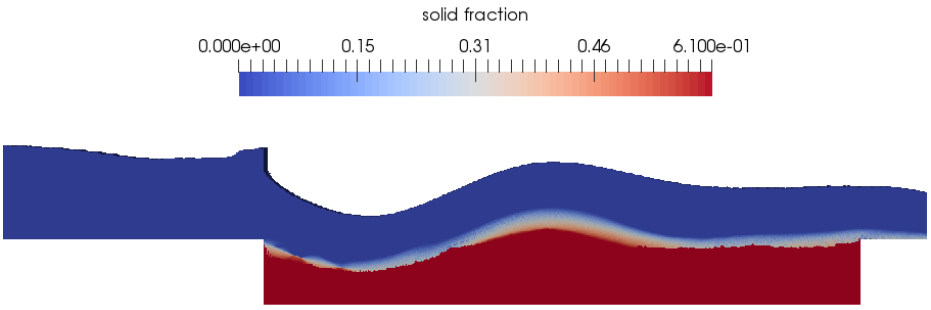
Figure 4.67 – Wall-jet scouring. Convergence of the numerical solution when discretization parameters go to zero: sediment and water profiles at $t = 1$ [s] for various mesh sizes.



(a)



(b)



(c)

Figure 4.68 – Wall-jet scouring. Snaphots of sediment profiles at $t = 1.5$ [s] for (a) Coarse mesh (b) Intermediate mesh (c) Fine mesh.

We consider the intermediate mesh for the rest of the computations. In a second step, we run the same experiment, without allowing the resuspension effects ($K_r = 0$). In Figure 4.69, we show a snapshot of the numerical solution at the final time $t = 3.5$ [s]. It shows that, without resuspension, the scouring does not occur. Instead, the liquid leaves the reservoir towards the outflow region, without carrying the sediment along. Thus, in this kind of experiment, resuspension effects are necessary to observe such phenomena. We introduce the resus-

pension effects again by taking $K_r = 10^{-3}$, $f_{s_{CO}} = 0.6099$ and $\varepsilon = 10^{-11}$, on the intermediate mesh and running the experiment with a constant critical shear $\tau_{CR} = 0.05$ [N/m^2]. In Figure 4.70, we illustrate a snapshot of the smothed shear stress at $t = 1.5$ [s]. In Figure 4.71 we show the results for the constant critical shear model at $t = 1.5$ [s] and $t = 3.5$ [s]. We observe that, the numerical results are comparable with the experimental results (described in [156]) in the stationary state (at 3.5 [s]). However, in the intermediate stage (e.g. at 1.5 [s]), the numerical solution does not match well with the experiment. We denote D_{final} as the final scour-hole diameter of the experiment. In Figure 4.72 we show the variation of the scour hole diameter over time normalized by D_{final}. It shows that the stationary state is obtained, but that the transition is faster than the actual experiment. In fact, the issue observed with our constant shear model was also observed in the results obtained with the Colomb-shear model mentionned in the same article [156].

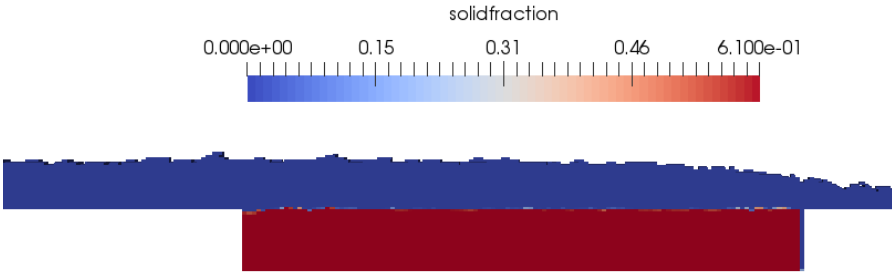


Figure 4.69 – Wall-jet scouring. Snapshot of the sediment profile at $t = 3.5$ [s] without resuspension ($K_r = 0$).

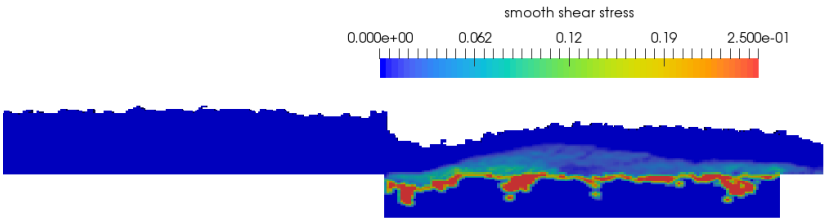


Figure 4.70 – Wall-jet scouring. Snapshot of smoothed shear stress at $t = 1.5$ [s].

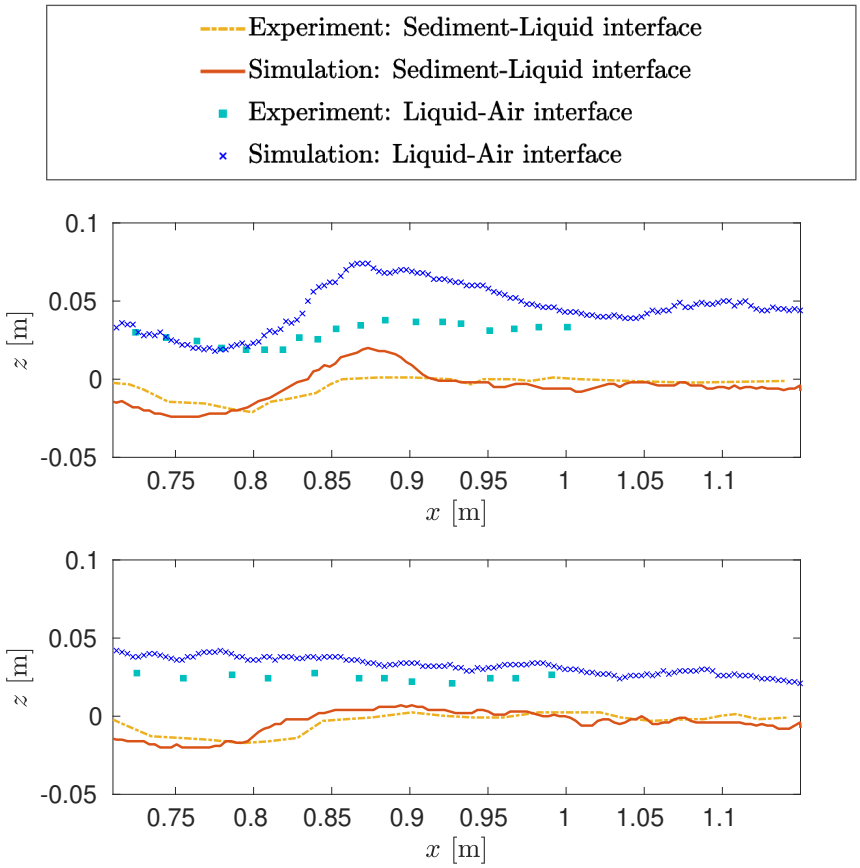


Figure 4.71 – Wall-jet scouring. Numerical results with the constant critical shear model: Sediment and water profiles in comparison with the experimental results. At $t = 1.5$ [s] (top) and at $t = 3.5$ [s] (bottom).

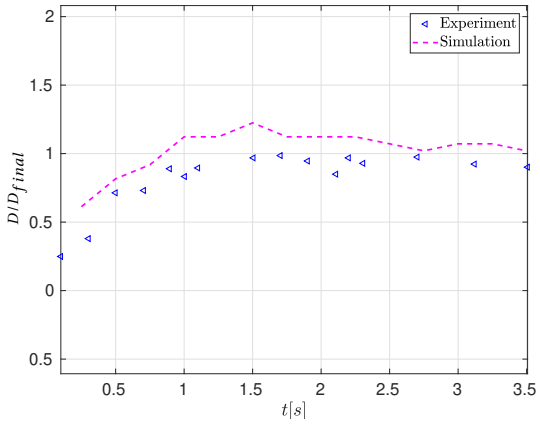


Figure 4.72 – Wall-jet scouring. Scour-hole width over time: numerical results with a constant critical shear model in comparison with the experimental results.

In an attempt to improve the result, especially during the intermediate phases, we developed a model to include the Shields shear, inspired by [156] (see Subsection 2.1.3). We run the experiment on the intermediate mesh with the Shields critical shear. Figure 4.73 shows the sediment and liquid profiles for the numerical results with the Shields critical shear and the experiment. Figure 4.74 illustrates that the maximal diameter of the scour hole varies the same way over time for both the experimental and the numerical solutions. In Figure 4.75, the snapshots at various times illustrate that the scouring process is simulated better with the Shields model. However, we can see that there is always an overestimation of the liquid level at all times. The same issue was also observed in the numerical results presented in [156] with respect to the experimental ones.

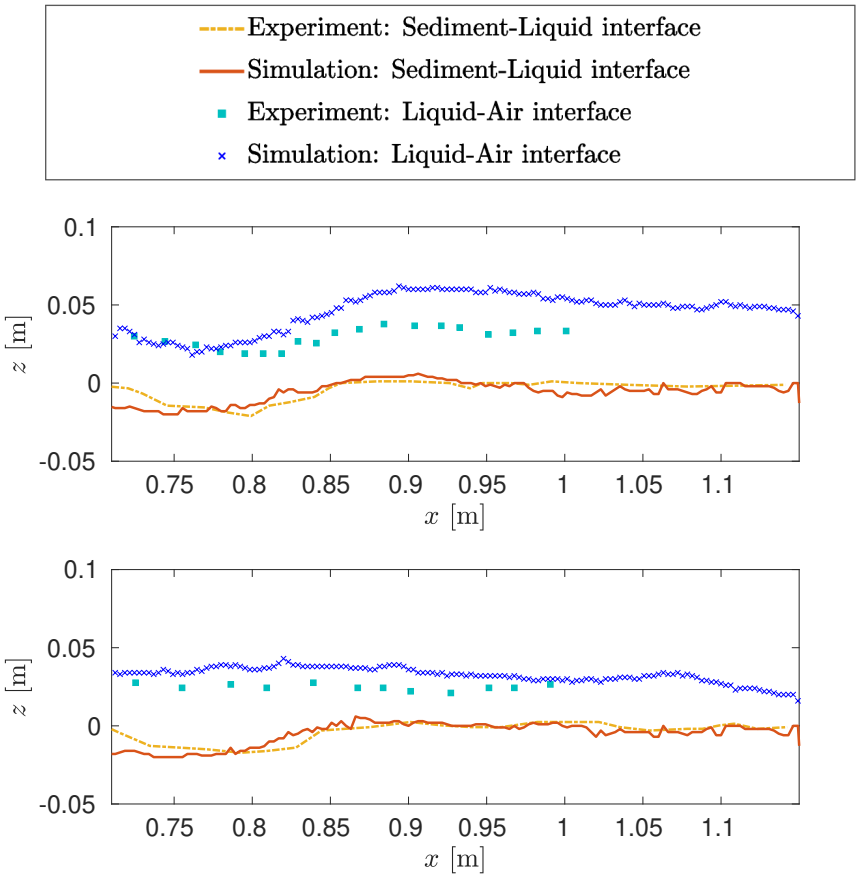


Figure 4.73 – Wall-jet scouring. Numerical results with the Shields critical shear model: Sediment and water profiles in comparison with the experimental results. (a) At $t = 1.5$ [s] (b) At $t = 3.5$ [s] .

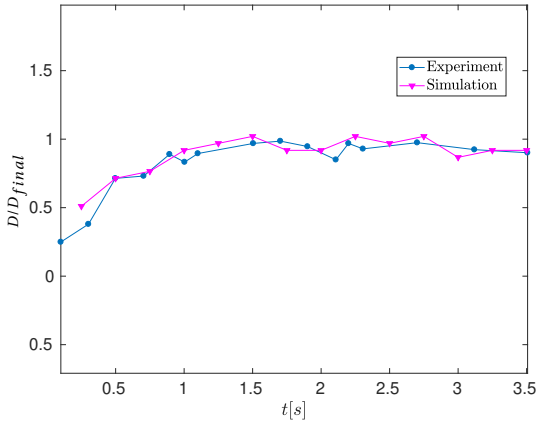


Figure 4.74 – Wall-jet scouring. Scour-hole width over time: numerical results with the Shields critical shear model in comparison with the experimental results.

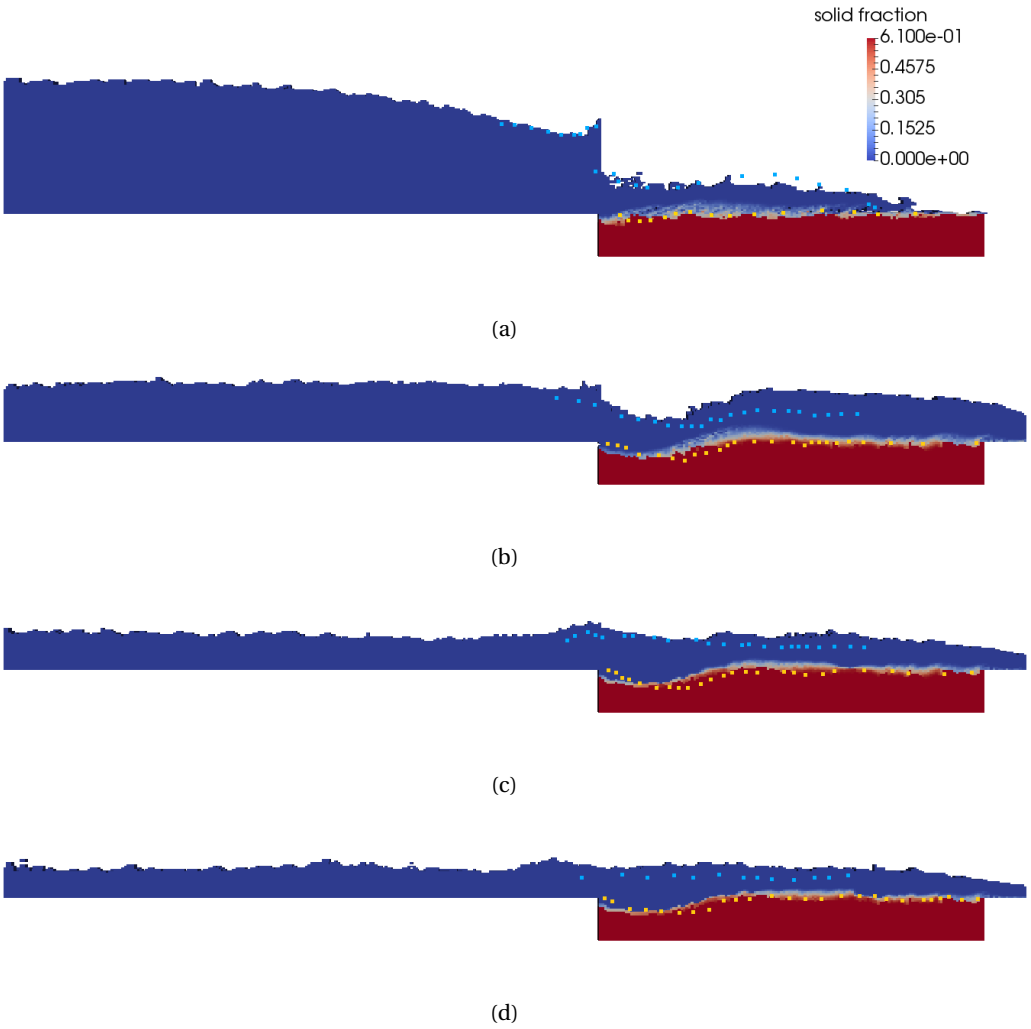
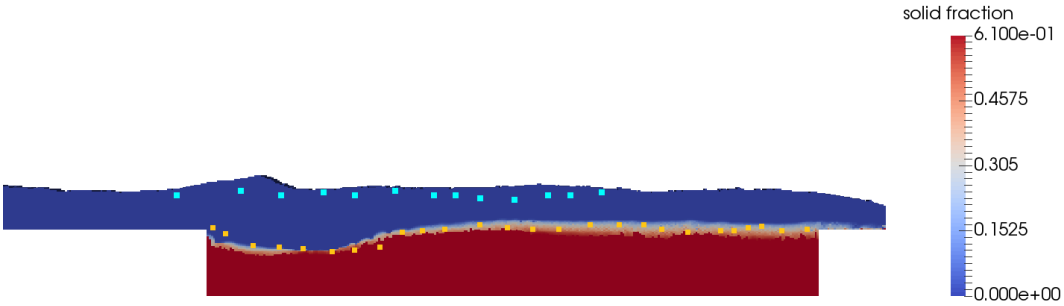


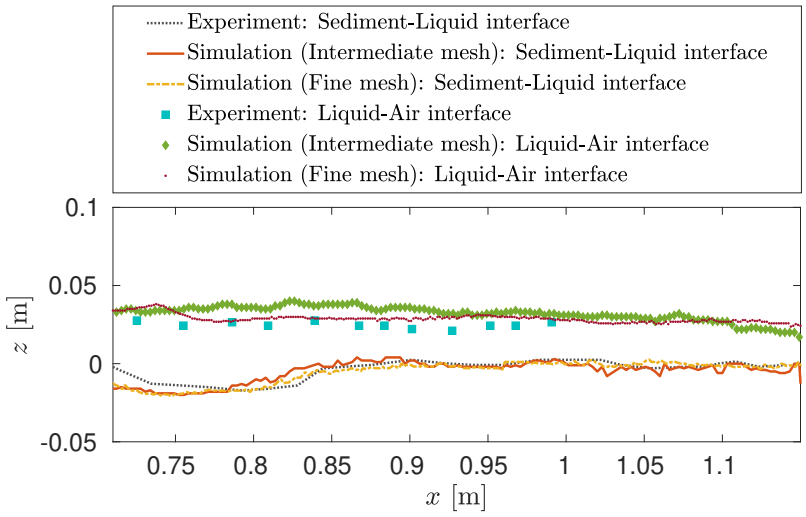
Figure 4.75 – Wall-jet scouring. Snaphots of sediment profiles at (a) $t = 0.3$ [s] (b) $t = 1.5$ [s] (c) $t = 2.7$ [s] and (d) $t = 3.5$ [s] with the Shields critical shear model, with $K_r = 10^{-3}$, $f_{s_{CO}} = 0.6099$ and $\varepsilon = 10^{-11}$. The yellow (resp. the light blue) dots represent the experimental sediment level (resp. liquid level).

Furthermore, we want to evaluate the convergence order of the numerical order for the sediment profile. First, we run the experiment, with the Shields critical shear, on the fine mesh. In Figure 4.76, we illustrate a snapshot of the numerical solution on the fine mesh at $t = 3.5$ [s]. We show the sediment and water profiles on the intermediate and the fine meshes in the same figure. Then, for each of the previously introduced meshes, we compute the error defined in Subsection 4.2.4 by (4.19) for both times: $t = 1.5$ [s] and $t = 3.5$ [s]. Respective mesh sizes are $h_1 = 0.005$ [m], $h_2 = 0.0022$ [m] and $h_3 = 0.0014$ [m] for the coarse, the intermediate and the fine meshes. We plot the computed error along with the first-order slope in Figure 4.77. It shows the proper order of convergence one. Finally, in Figure 4.78, we show the numerical sediment and liquid profiles for the finite volume method, the splitting method with the finite

differences approach, and the splitting method with the finite elements approach. The results are very similar in this setup, even with the finite elements approach.



(a)



(b)

Figure 4.76 – Wall-jet scouring. Results with Shields critical shear model on the intermediate and the fine meshes at $t = 3.5$ [s]: (a): Snapshot of the numerical solution on the fine mesh, the yellow (resp. the light blue) dots represent the experimental sediment level (resp. liquid level) - (b): Plot of sediment and water levels on the intermediate and the fine meshes.

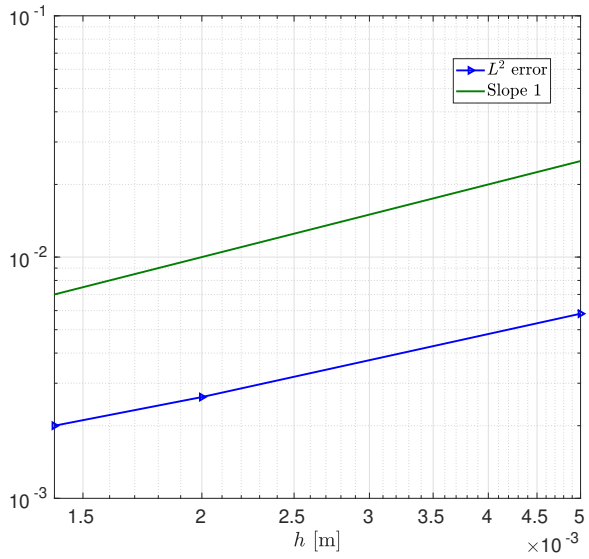


Figure 4.77 – Wall-jet scouring. L^2 error between the experiment and the numerical sediment heights for various meshes at $t = 3.5$ [s](computed as (4.19)).

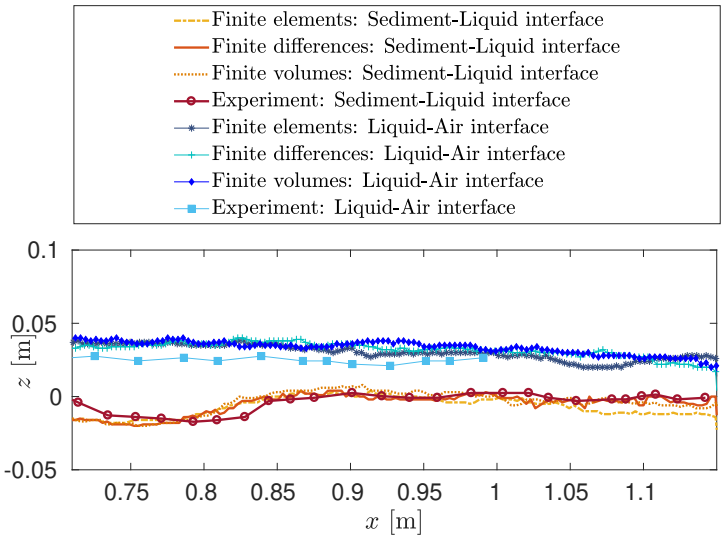


Figure 4.78 – Wall-jet scouring. Plots of sediment and liquid profiles on the intermediate mesh with different methods to solve the sedimentation equation. The results displayed are obtained on the intermediate mesh at $t = 3.5$ [s].

In Table 4.7, for each of the previously defined meshes, we present the average CPU time (in [s]) per time step Δt for this experiment for each operator: the Stokes operator, the deposition/resuspension operator, and the advection operator.

Table 4.7 – Wall-jet scouring. Average CPU time per time step in [s] for each operator, for the coarse mesh ($\Delta t = 0.05$ [s], intermediate mesh ($\Delta t = 0.025$ [s]) and fine mesh ($\Delta t = 0.0125$ [s]) all defined earlier. The results are given for the chosen parameters $\varepsilon = 10^{-11}$, $K_r = 10^{-3}$ and $\tau_{CR} = 0.05$ [N/m^2] and over a computation time of 3.5 [s].

	Stokes step	Deposition + resuspension step		Advection step	
		Deposition	Resuspension	Advection	Decomp + SLIC
Coarse	0.34	0.021	2.76	0.14	0.13
Intermediate	5.24	0.32	43.8	2.18	1.98
Fine	83.1	5.1	700.2	34.5	32.5

We remind that substeps have to be taken in order to solve the resuspension step, respecting (3.31): in this case, for each time step, 27 susbsteps are used to solve the resuspension, in average.

4.2.6 Viscous resuspension in a cylindrical rheometer

In this experiment, based on the work done in [157], the objective is to further test the resuspension of particles in a viscous fluid within a domain with curved boundaries. As shown in Figure 4.79, the experiment setup consists in having a fluid and a layer of sediment sit in a gap of 5 [mm] between two cylinders. The interior cylinder with a radius $R_i = 19$ [mm] is rotating with rotating velocity Ω [tr/min] along the vertical axis $\boldsymbol{e_z}$. The exterior cylinder of a radius $R_e = 24$ [mm] is fixed. The height of both cylinders is 10.4 [cm].

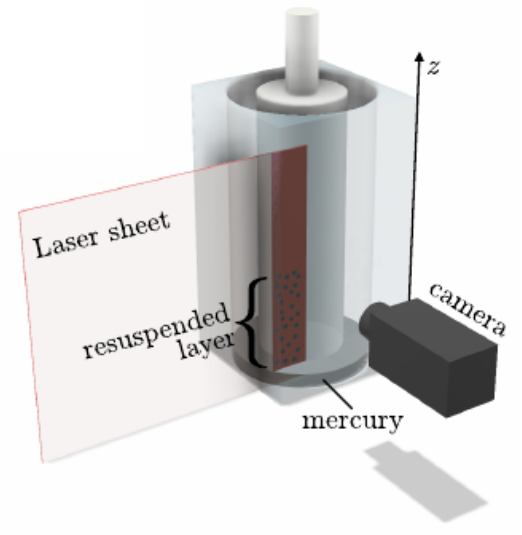


Figure 4.79 – Resuspension in a cylindrical rheometer. Setup of the real experiment: the particles are suspended in the fluid in the gap between the interior rotating cylinder and the exterior fixed cylinder. Image exctracted from [157].

The sediment particles' diameter d_* is approximately 2.84×10^{-4} [m]. The dry sediment has a density $\rho_s = 1190$ [g/m^3]. The fluid has a density $\rho_l = 1060$ [g/m^3] and a viscosity equal to 0.34 [Pa/s]. Initially, the sediments are packed at a fraction $f_s \approx 0.587$ of a height equal to 5 [mm]. As the velocity in the sediments is much lower than the velocity above the level of sediments, this creates a vertical shear that causes the resuspension of these particles. The experiments consist in varying the rotation velocity and observing the final state of the resuspended particles, i.e, different velocities results in different final fractions f_s and different heights at which the particles settle in a stationary state. The process is very slow, but experiments estimate that after about 120 [s], the state for all of the considered rotational velocitites is stationary.

Qualitative study

Before we proceed with running the actual experiment, we first conduct a qualitative study. For computational effort reasons, we consider a smaller domain such that $R_i = 10$ [mm], $R_e = 12$ [mm] and both cylinders' height is 0.005 [m]. We consider an initial sediment height equal to 0.00125 [m]. The reason behind considering a smaller domain is mainly to decrease the computational time. Figure 4.80, shows the current finite element domain and the structured grid. We consider a time step Δt such that the CFL number is smaller than 1. Moreover, we run the experiment until $T = 10$ [s]. In Figure 4.81, we show the initial condition of the qualitative experiment. Each rotational velocity Ω is inferred by imposing a Dirichlet BC for the velocity on the inside lateral surface cylinder surface. Slip boundary conditions are applied on the outside lateral surface and the top surface while no slip boundary conditions are applied on

the bottom surface. For $\Omega = 5$ [rpm], the velocity profile is displayed after a few seconds in Figure 4.82.

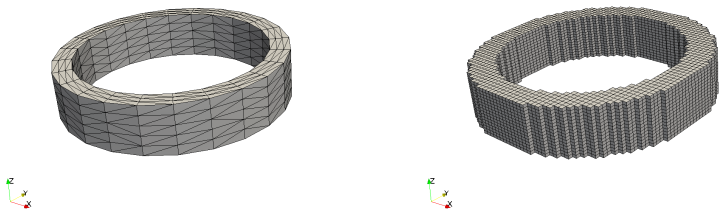


Figure 4.80 – Resuspension in a cylindrical rheometer - qualitative results. Left: Finite elements triangulation, right: structured grid cells.

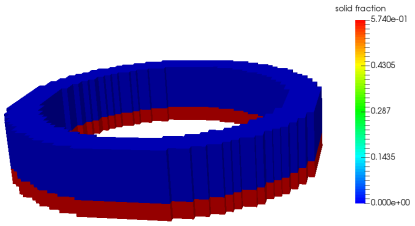


Figure 4.81 – Resuspension in a cylindrical rheometer - qualitative results. Initial condition; the sediment, displayed in red, is packed at the bottom of the domain. The domain is initially full of liquid.

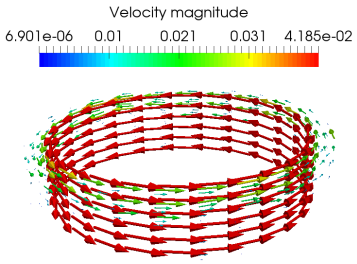


Figure 4.82 – Resuspension in a cylindrical rheometer - qualitative results. Velocity profile for $\Omega = 5$ [rpm]

Effect of the rotation velocity and the resuspension constant First, for a constant resuspension constant K_r, we vary the rotational velocity, Ω, and see how it affects the results. Second, we vary the resuspension constant K_r and check the result for a fixed rotational velocity of Ω. In order to display snapshots of the results, a surface clip is in the plane (xz) is used (as shown in Figure 4.83). In Figure 4.84, for $\Omega = 0$ [rpm], $\Omega = 1$ [rpm], $\Omega = 5$ [rpm] and $\Omega = 10$ [rpm] and $K_r = 10^{-7}$ we show the effect of the rotational velocity on the sediment resuspension. On the left, we show the average height reached by the sediment over time. On the right, we illustrate the average solid fraction f_s per vertical position z. The level of sediments rises when the velocity increases, as expected. Snapshots of the results for $\Omega = 1$ [rpm] and $\Omega = 10$ [rpm] are illustrated in Figure 4.85. On the other hand, Figure 4.86 shows, for $\Omega = 1$ [rpm] , the effect of K_r on the particles resuspension. The higher the resuspension constant K_r, the higher the vertical height reached by the sediments. For $K_r = 10^{-4}$, a snapshot is illustrated in Figure 4.87 and show that all sediment is roughly in resuspension.

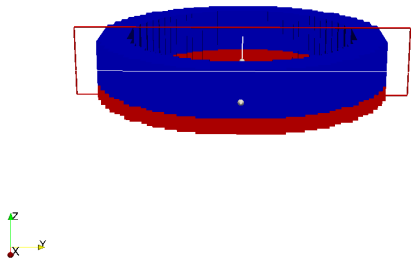


Figure 4.83 – Resuspension in a cylindrical rheometer - qualitative results. Display of the surface clip used to illustrate the result snapshots.

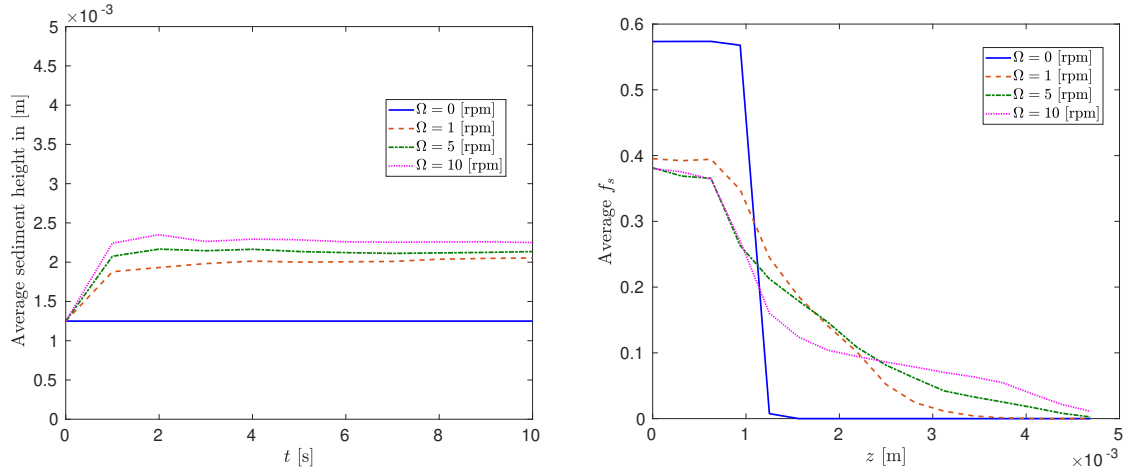


Figure 4.84 – Resuspension in a cylindrical rheometer - qualitative results: effect of the rotational velocity. Left: Average sediment height over time, right: Average solid fraction value per vertical position.

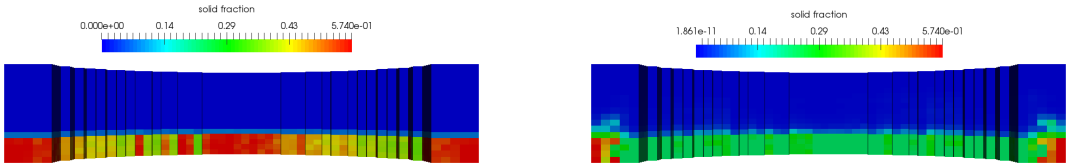


Figure 4.85 – Resuspension in a cylindrical rheometer - qualitative results: Snapshots of sediments at $t = 10$ [s], for $K_r = 10^{-7}$ and $\Omega = 1$ [rpm](left) and $\Omega = 10$ [rpm] (right).

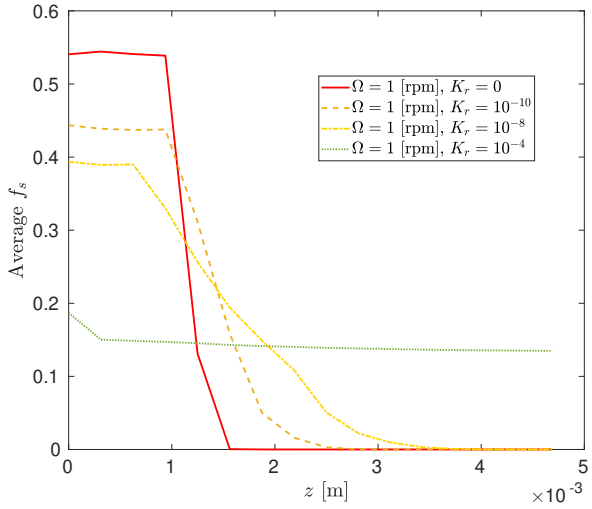


Figure 4.86 – Resuspension in a cylindrical rheometer - qualitative results: Effect of K_r on the sediment resuspension for $\Omega = 1$ [rpm]

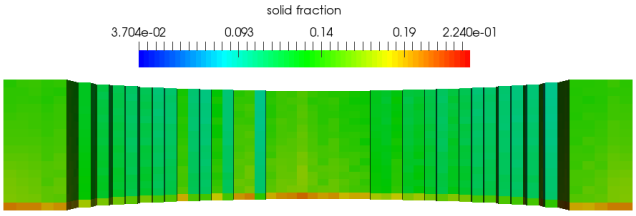


Figure 4.87 – Resuspension in a cylindrical rheometer - qualitative results: Snapshots of sediments at $t = 10$ [s], for $\Omega = 1$ [rpm] and $K_r = 10^{-4}$.

Numerical tests and comparison with experiments

After displaying qualitative results that allowed us to test the effect of the resuspension parameters as well as the velocity on the sediment behavior, we consider the experiment detailed in [157]. Due to numerical limits (especially computational CPU time), the height of the cylinders is roughly twice as high as the initial sediment height. This gives enough space for the resuspension to occur. Figure 4.88 illustrates the computational domain that we use for the benchmarking phase. The sediment displayed in red, is initially packed at the bottom. We use the same boundary conditions described in the qualititave phase. We consider two rotational velocitites $\Omega = 20$ [rpm] and $\Omega = 0.5$ [rpm] and we run the experiments up to $T = 120$ [s] where the solution is expected to have reached stationnarity [157].

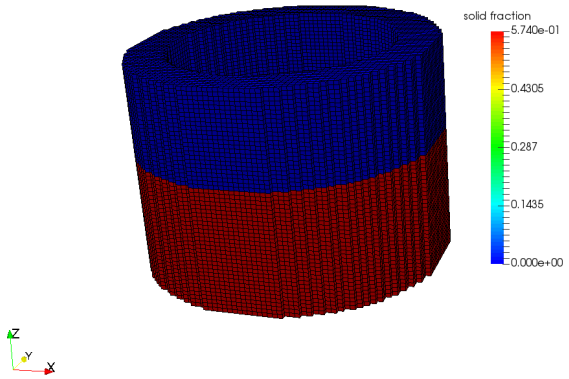


Figure 4.88 – Resuspension in a cylindrical rheometer: Numerical setup. The cylindre's height is 1 [cm], whereas the sediment (in red), is of initial height 0.5 [cm].

Sensitivity analysis with respect to the CFL number Since the boundaries are curved in this experiment, the choice of the time step is very important. More specifically, if the time step is very large, we might advect the computed quantities, including the sediment, outside the domain. As explained in the previous chapter (in Subsection 3.2.5), this results in excessive post-processing, to conserve the mass of sediments. These algorithms ensure the mass conservation, but can be time consuming, especially if the lost quantities are very big. They also cause a lack of accuracy. Our goal is to find a range of CFL numbers that minimize the loss of advected quantities, to prevent the latter. In order to do so, we run the experiment with different CFL numbers and check the sediment loss percentage over time without post-processing algorithms (decompression and redistribution algorithms). We display the obtained results in Table 4.8.

Table 4.8 – Resuspension in a cylindrical rheometer. CFL effect on sediment loss over time without decompression and redistribution algorithms.

CFL number	Average loss of sediment (loss per time step)	Loss after 100 [s]
22	30%	100%
4.7	2.62 %	40 %
1.15	0.268 %	17.8 %
0.45	0.14 %	23 %
0.1	0.034%	21.27 %

From these results, we quantify how having a very big CFL results in a massive loss. On the other hand, having a CFL number smaller than one is not necessarily better as shown in Table

4.8 (since it results in more interpolations between the two grids). A compromise between a CFL that is small enough to prevent a lot of quantity loss per time-step, yet, big enough for the quantity loss not to accumulate over time, is necessary. A CFL number that roughly equal to 1 is the best choice for our case. Having chosen the optimal CFL number, we now activate the post-processing algorithms again.

Mesh convergence We evaluate the convergence of the numerical method, when the discretization parameters (time step and mesh size) decrease. In order to do so, we run a simulation over 5 [s] on three different discretizations:

1. **Coarse mesh:** 1344 elements, 384 nodes, 27000 cells ($H \simeq 0.004$, $h \simeq 0.0014$) and $\Delta t = 0.035$
2. **Intermediate mesh:** 12960 elements, 2880 nodes, 198000 cells ($H \simeq 0.002$, $h \simeq 0.0007$) and $\Delta t = 0.0175$
3. **Fine mesh:** 123120 elements, 23560 nodes, 1584000 cells ($H \simeq 0.001$, $h \simeq 0.0035$) and $\Delta t = 0.00875$

For all the simulations, we take $\Omega = 20$ [rpm] and $K_r = 10^{-7}$. In order to simplify the illustration of the results, from now on, we choose to display the snapshots on a slice of the cylinder in the xy plane, as illustrated in Figure 4.89. The left figure shows the plane position of the slice, and the right figure shows the obtained slice. Furthermore, we will illustrate the snapshots on the left side of this slice. To simplify the description of the results later on, we will refer to this view by view of the left side. In Figure 4.90 we show snapshots (view of the left side) of the solid fraction on all three meshes. In Figure 4.91, we show the average solid fraction per vertical position z on all three meshes. Both figures show the convergence of the method.

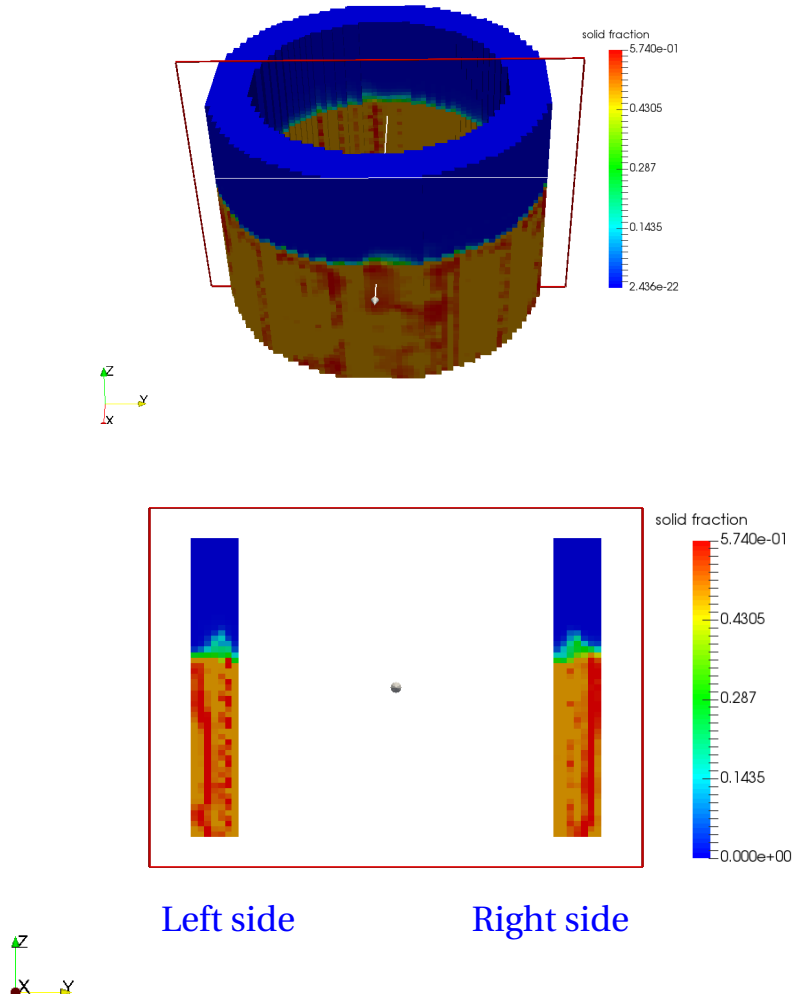


Figure 4.89 – Resuspension in a cylindrical rheometer - Display of the slicing surface used for the snapshots of the solid fraction. Top: the slicing surface position. Bottom: The obtained slice, snapshots are displayed on the left surface of this slice.

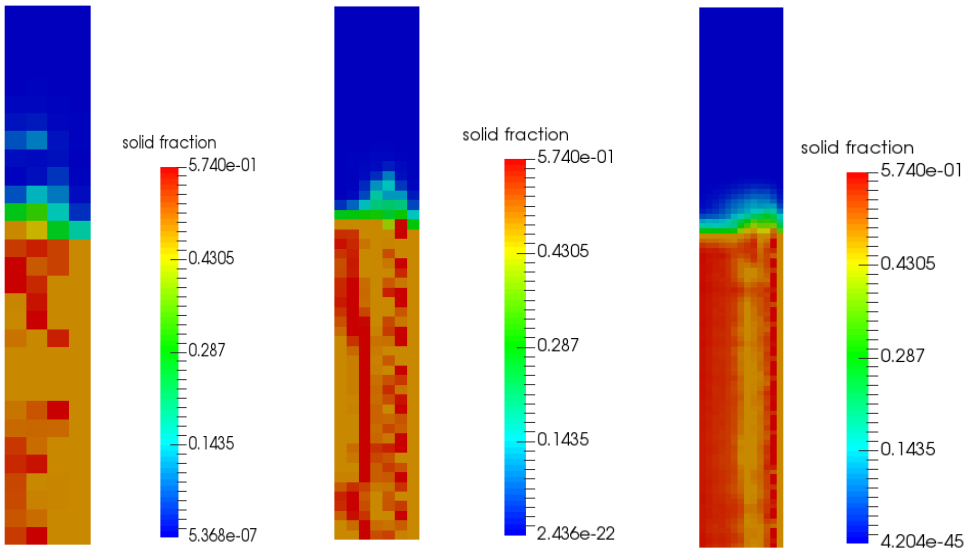


Figure 4.90 – Resuspension in a cylindrical rheometer - Snapshots (view of the left side) of the solid fraction on the coarse mesh (left), intermediate mesh (middle) and the fine mesh (right).

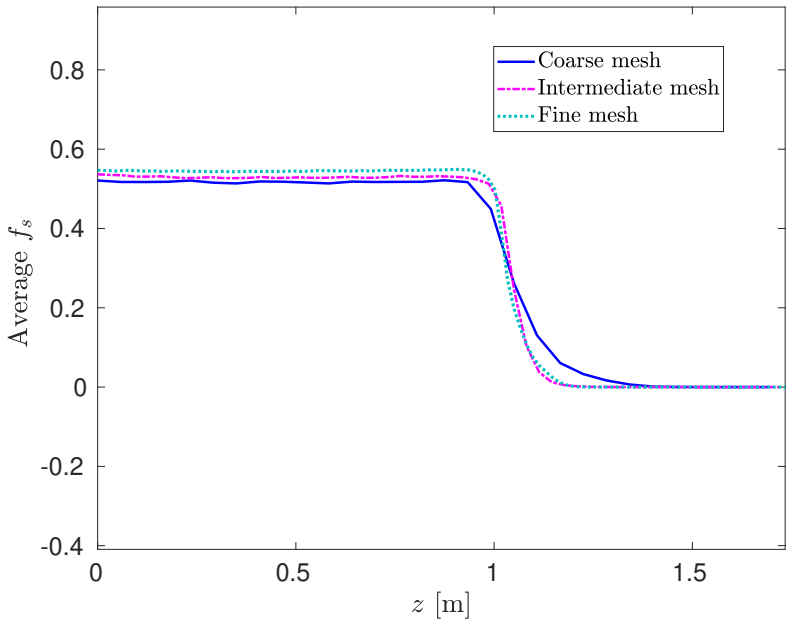


Figure 4.91 – Resuspension in a cylindrical rheometer. Average solid fraction value per vertical position for a coarse, and intermediate and a fine mesh.

Comparison Using the intermediate mesh, we run the experiment for $\Omega = 20$ [rpm] then $\Omega = 0.5$ [rpm] respectively, and compare the results with theoretical predictions from [158] and experimental results obtained in [157]. Figure 4.92 shows the obtained results for $K_r = 10^{-6}$ and $\tau_{CR} = 0.1 N/m^2$.

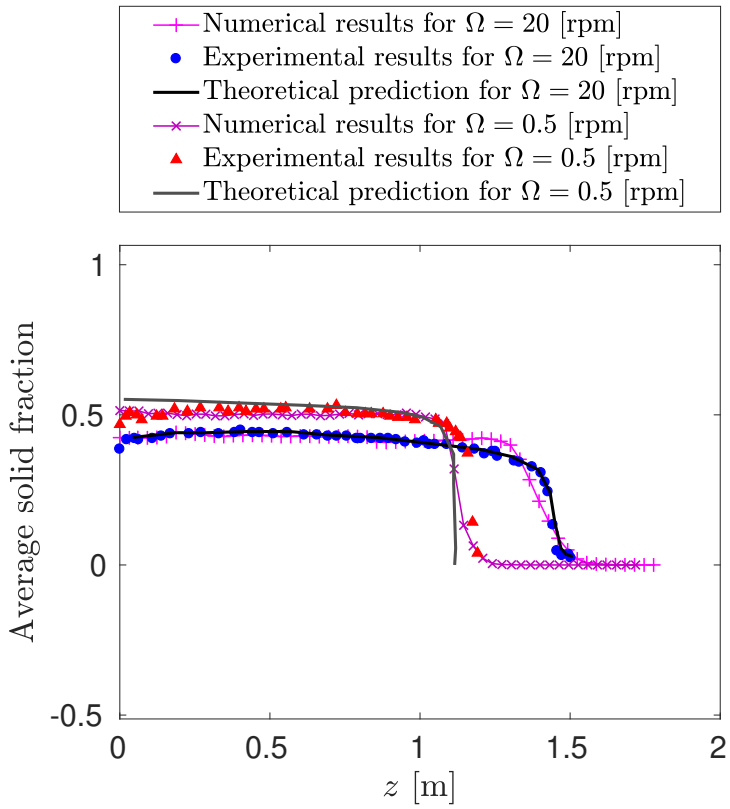


Figure 4.92 – Resuspension in a cylindrical rheometer. Real experiment: Average f_s per vertical position z at $t = 100s$ for $\Omega = 20$ [rpm] and $\Omega = 5$ [rpm] with $K_r = 10^{-6}$ and $\tau_{CR} = 0.1$ [N/m^2]. Experimental data is extracted from [157] and theoretical data is extracted from [158].

Furthermore, the choice of the resuspension parameter K_r is justified through a sensitivity analysis, as illustrated in Figure4.93. It shows the effect of the resuspension parameter K_r for the case where $\Omega = 20$ [rpm]. We can see that for $K_r = 10^{-7}$, there is not enough resuspension to suspend the sediments as desired, whereas for $K_r = 10^{-5}$, the sediments are suspended more than desired. For $K_r = 10^{-6}$ on the other hand, we obtain a better approximation.

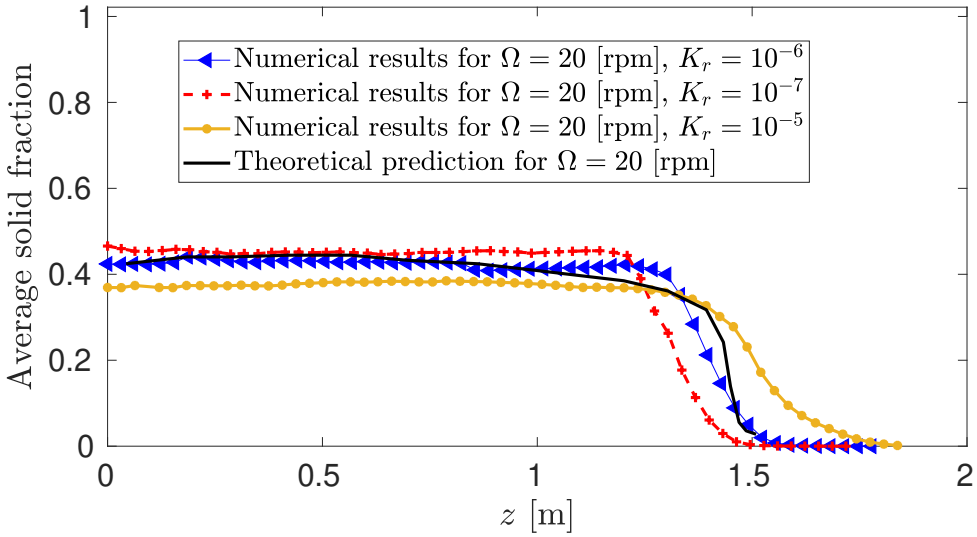


Figure 4.93 – Resuspension in a cylindrical rheometer. Real experiment: Average f_s per vertical position z at $t = 100s$ for $\Omega = 20$ rpm with $\tau_{CR} = 0.1$ [N/m^2] and different K_r values.

We illustrate the average sediment height obtained over time, for $\Omega = 20$ [rpm] in Figure 4.94. It shows that the maximal height is almost obtained in the first 30 seconds. The process becomes slow, and homogenization in the whole domain, through the resuspension effects, is done slowly. In Figure 4.95, we show snapshots of the solid fraction, at the stationary state for both rotational velocities.

In Figure 4.95, we show snapshots of the solid fraction, at the stationary state for both rotational velocities. In general, our results show good agreement with theoretical and experimental expectations, especially for $\Omega = 0.5$ [rpm]. However, in the theoretical and the experimental profiles, it is observed that the concentration is almost constant in the resuspended layer and drops to zero quite sharply, even for the highest angular velocity. However, in our case, we observe that the transition between the resuspended layer and the zero concentration is smoother. This is especially the case for the higher velocity ($\Omega = 20$ [rpm]). It is known that numerical diffusion causes this smoothening effect of sharp interfaces. Furthermore, when the rotational velocity is higher (for $\Omega = 20$ [rpm]), our results show that the sediments are slightly more packed than expected. The post-processing algorithms (the decompression and the redistribution) become more relevant as the velocity increases to ensure mass conservation. As explained in Subsection 3.2.5, in this particular case, we rely on the homogeneous redistribution of the quantities advected outside the domain (shown in Algorithm 4). Thus, the algorithm slows down the resuspension.

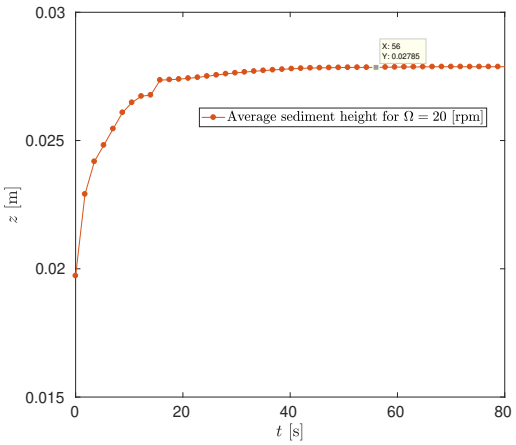


Figure 4.94 – Resuspension in a cylindrical rheometer. Real experiment: Sediment height over time for $\Omega = 20$ [rpm], with $K_r = 10^{-6}$ and $\tau_{CR} = 0.1$ [N/m^2] .

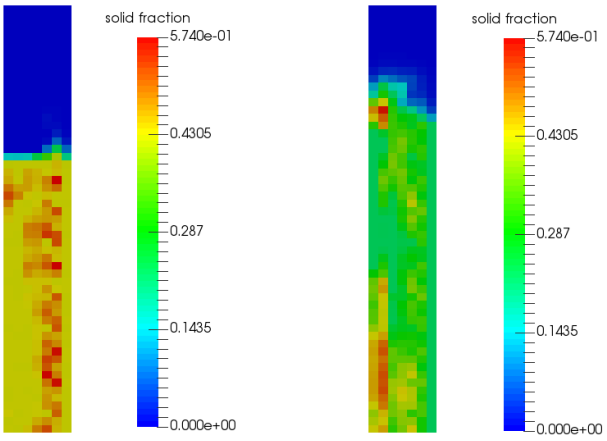


Figure 4.95 – Resuspension in a cylindrical rheometer. Real experiment: Snapshots (view of the left side) of the solid fraction for $\Omega = 5$ [rpm] (left), and for $\Omega = 20$ [rpm] (right) at $t = 100$ [s], with $K_r = 10^{-6}$ and $\tau_{CR} = 0.1$ [N/m^2].

Remark: Post-processing technique by averaging

The averaging technique is quite common in such experiments. The experimental results that we used to benchmark from [157] are also obtained by averaging the results over approximately 10000 images of decorrelated frames. A frame is the intersection between the laser sheet (shown in Figure 4.79) and the cylinder. It is, in fact, similar to what we named a view of the left side. In our case, in order to compute the average solid fraction at the vertical position

$z_k = k\,h_z$, $1 \leq k \leq N_z$, we do the following:

1. We compute the total solid fraction at position z_k, denoted by f_{s_k}: for all cell C_{ijk}, $1 \leq i \leq N_x$ and $1 \leq j \leq N_y$, $f_{s_k} := f_{s_k} + f_{s_{ijk}}$
2. We divide the obtained value by the total number of cells at position z_k. The obtained value is the average solid fraction at the current position.

In order to compute the average sediment height at time t:

1. For each column of cells, we compute the sediment height in that column (the sediment height is computed by summing the number of cells in the vertical direction that contain a significant amount of sediment).
2. We divide the sum of obtained heights by the number of columns, to obtain the average sediment height.

4.2.7 River stream simulation

The main goal of this experiment is to qualitatively show our model's capacity for displaying self-generated bedforms (dunes) in sediment bed rivers. It also allows us to show that the mass conservation when introducing sediments into the domain by an inflow surface, is guaranteed. Due to the importance of dunes in river mechanics as well as their complex physical ascetics, studying them has generated much attention recently. Even though we do not quantitatively compare our results with experimental data, we base our numerical setup on the description given in [159]. The reason behind this is that ripples and dunes form under a relatively limited range of flow conditions, and specific conditions need to be put in place. The domain's typical size, the size of the particles, the velocity (hence the shear stress and the Reynolds number) are some of the parameters that are crucial when simulating such phenomena. In [159] and references therein, more details are given. As explained in [159], there exist various situations for the bedforms. As illustrated in Figure 4.96, bedforms are typically characterized by their heights and wavelengths [160, 161]. In the lower flow regime, the natural progression is from a flat bedform to sediment movement, then to ripples. As the velocity increases, slightly larger dunes are obtained. In the upper flow regime, the dunes start to flatten out and are completely washed out when the velocity rises even further [160, 161].

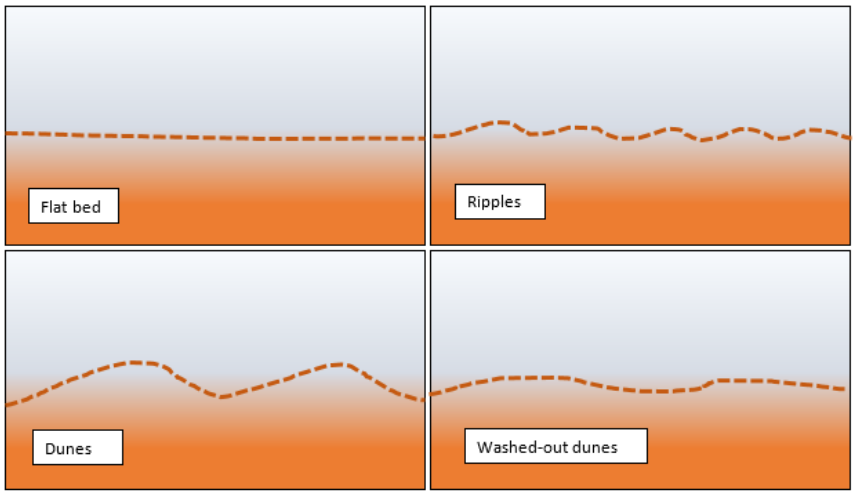


Figure 4.96 – An illustration of various bedforms.

The geometrical domain of horizontal, vertical and transversal dimension respectively denoted by L_x, L_z and L_y, is illustrated in Figure 4.97. A sediment layer of height $H_b = 0.005$ [m] is placed at the bottom of the domain. A thin upper layer of these sediments is free, and the rest are fixed (zero velocity). The sediment particles diameter is $d_* = 0.5 \times 10^{-3}$ [m]. The dry density of the sediment particles is $\rho_s = 2500$ [kg m^{-3}], while the liquid density is $\rho_l = 1000$ [kg m^{-3}] and the liquid kinetic viscosity is 1.5×10^{-6} [m^2/s]. An inflow is imposed on the left surface of the domain and free outflow conditions are allowed on the right surface. Slip boundary

conditions are allowed on the top surface and the lateral surfaces (above the fixed sediment height).

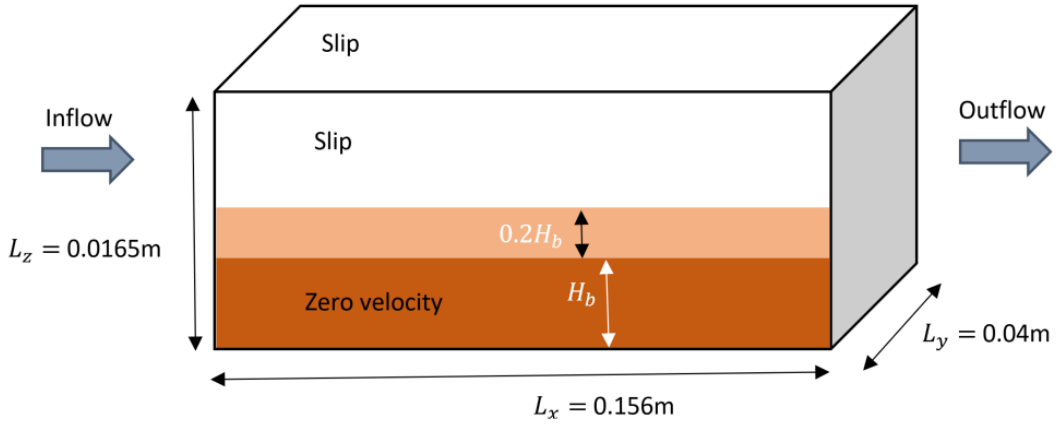


Figure 4.97 – River-stream simulation. A layout of the experimental setup: The light brown particles are moving particles; the dark brown particles are fixed.

Given the described setup, bedforms generated in experimental studies [159–161] fall into four cases, summarized in Table 4.9. Ripples are obtained for the first case. For the case 2, dune patterns are observed. For the third and the fourth cases, the dunes are flattened and washed out.

Table 4.9 – River-stream simulation. Bedform expectations based on the mean-velocity and the numerical setup.

	Case 1	Case 2	Case 3	Case 4
Description	Ripple	Large dune	Washing out	Washed out
Reynolds number $Re_b = 2U_bH_b/\nu$	6000	8000	10000	12000
Mean velocity U_b[m/s]	0.39	0.52	0.65	0.78

We want to simulate cases 1 and 2 to check if our model is capable of self-generating bedforms with physically meaningful heights and wavelengths. For the inflow velocity $\boldsymbol{u}_{in} = (u_x(z), 0, 0)\mathbb{1}_{z>H_b}$, we consider a logarithmic velocity profile:

$$u_x(z) = \frac{Q\,(1 + 2.5\ln(1+z))}{L_y\,(2.5\,(1+(L_z-H_b))\ln(1+(L_z-H_b)) - 1.5(L_z-H_b))}$$

where Q is the volumic flow rate, computed such that the mean inflow velocity is within the test case range. Such profile assures zero value at the bottom, and increasing values as z increases. The simulations are run on a sufficiently fine mesh, and the discretization parameters are the following: 15276 elements, 4149 nodes (Average size $H = 0.001$), 75000 cells ($h = 0.0002$) and

the time step is chosen such that the CFL number is approximately 5 ($\Delta t = 0.0025$ for the first case, and $\Delta t = 0.002$ for the second case). We consider a resuspension constant $K_r = 10^{-6}$ and a constant shear model with critical shear stress $\tau_{CR} = 0.1$ [N/m^2]. On this setup, we run an experiment up to $T = 24$ [s]. In Figure 4.98, we illustrate the computational domain (the finite elements mesh). In the same figure (bottom figure), we illustrate the velocity profile at time $t = 3$ [s] for the second case. We consider the penalization parameter $\varepsilon = 10^{-11}$ for all cases.

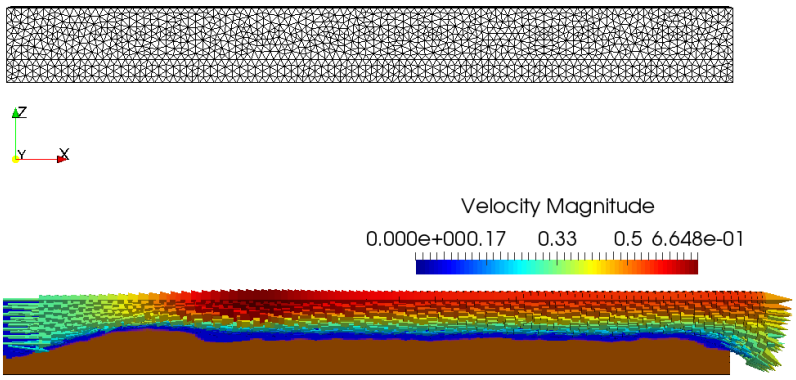


Figure 4.98 – River-stream simulation. Top: An illustration of the finite element grid. Bottom: An illustration of the velocity profile at time $t = 3$ [s].

To facilitate the presentation and compare with results in [159], the physical time t is normalized by the constant H_b/U_b and the resulting non-dimensional time is denoted as $t^* = t/(H_b/U_b)$. We show snapshots of the sediment profile for cases 1 and 2 in Figure 4.99 and Figure 4.103, respectively. In Figure 4.101, we display snapshots of the smoothened shear stress over time. It shows that the shear stress appears where the sediments are expected to be resuspended and convected to form dunes. To further investigate the stability of the bedform, the profiles of the bed surfaces obtained in each both cases during $t^* \in [600, 800]$ are plotted in Figure 4.100 for Case 1 and Figure 4.104 for Case 2. This time interval is selected because the bedforms fully develop during this time. The figures are plotted using the relative longitudinal location $X = x - U_d t$, normalized by the sediment diameter d_*, where x is the fixed downstream direction, and U_d is the bedform migration velocity, introduced in [159]. When the bedform is stable, bedform surface profiles are overlapping when plotted in the moving-frame. We can see that the self-generated bedforms in Case 1 and 2 are stable as the bed surface profiles are highly overlapping. In Figure 4.102, we show snapshots of the sediment profile for Case 1 when $K_r = 0$ (i.e. no resuspension effects). From this figure, we observe that the additional beforms do not appear. In fact, the ripples appear as a result of shear-induced motion.
Furthermore, we run an additional test for a mean velocity that is slightly smaller that that first considered in Case 1 and show the results in Figures 4.105 and 4.106. They show that, since the velocity is smaller, smaller ripples have the time to develop before being washed away. In

Figure 4.107, we present the bedform variation over time for Case 4. We note that the dune structure at a given time is highly unstable and significantly less dune-like than the bedforms in Cases 1 and 2. The shear stress of the fluid flow is large enough for the bedforms to wash out almost immediately after they appear.

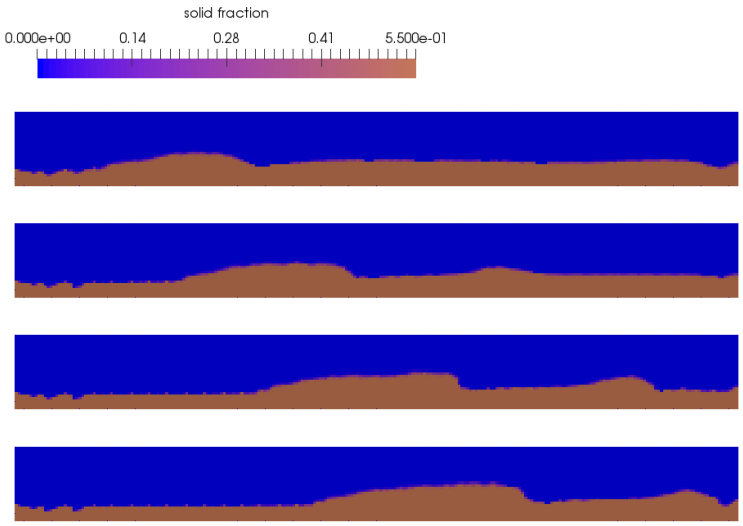


Figure 4.99 – River-stream simulation. Case 1, top to bottom: snapshots of sediment profiles for $t = 5$ [s], $t = 10$ [s], $t = 15$ [s] and $t = 20$ [s] ($K_r = 10^{-6}$).

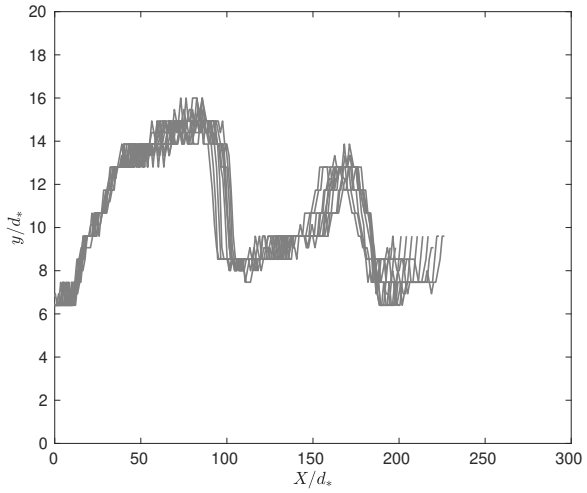


Figure 4.100 – River-stream simulation. Case 1, the surface of the bed when $t^* \in [600, 800]$. The x-axis is the relative longitudinal location X/d_*, $X = x - u_d t$. x is the horizontal direction u_d is the migration velocity of the bedform ($K_r = 10^{-6}$).

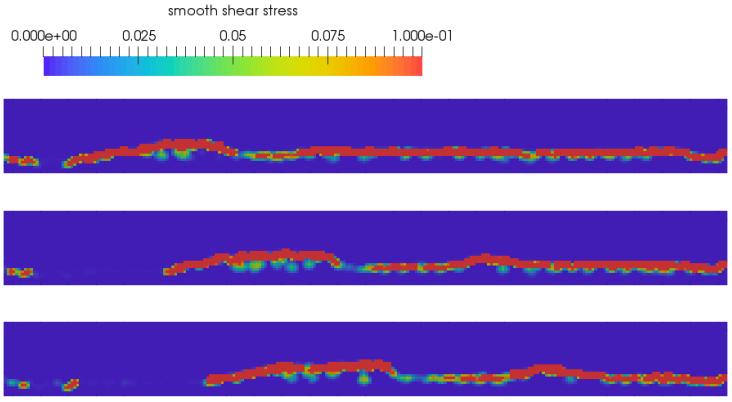


Figure 4.101 – River-stream simulation. Case 1, top to bottom: snapshots of smooth shear stress for $t = 5$ [s], $t = 10$ [s]and $t = 12.5$ [s]. Resuspension occurs when the shear stress is above $\tau_{CR} = 0.1$ [N/m^2].

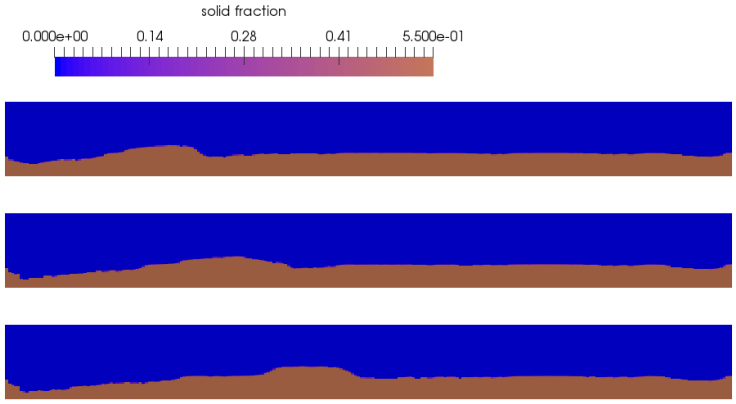


Figure 4.102 – River-stream simulation. Case 1, top to bottom: snapshots of sediment profiles for $t = 5$ [s], $t = 10$ [s]and $t = 15$ [s] ($K_r = 0$).

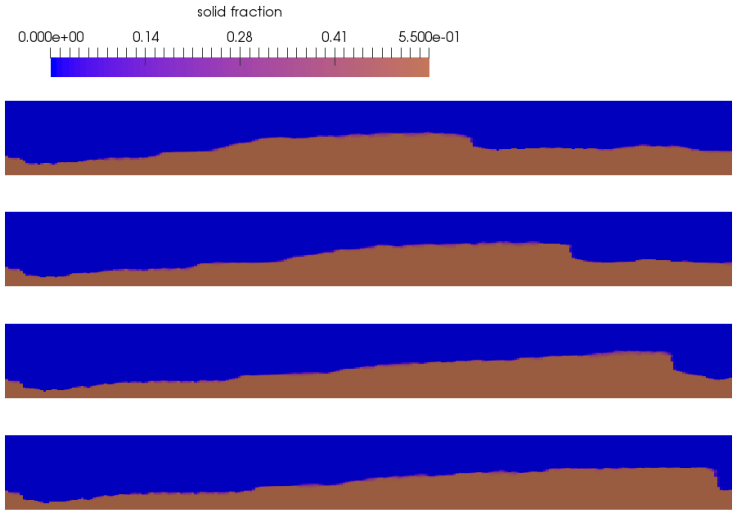


Figure 4.103 – River-stream simulation. Case 2, top to bottom: snapshots of sediment profiles for $t = 5$ [s], $t = 10$ [s], $t = 12.5$ [s] and $t = 15$ [s] ($K_r = 10^{-6}$).

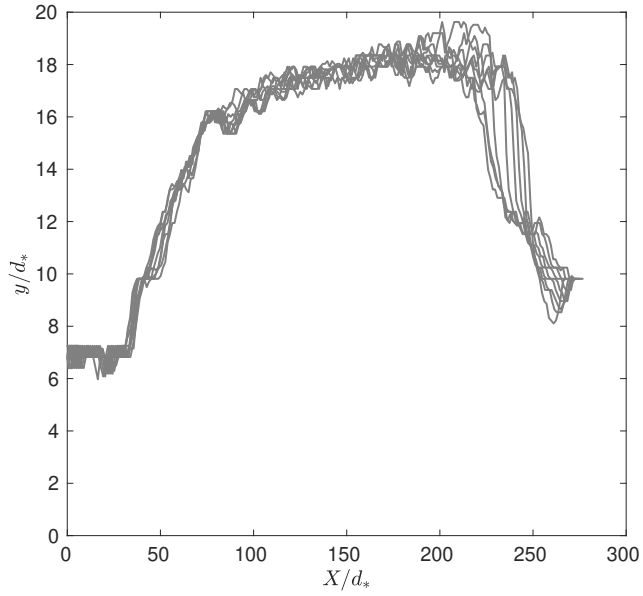


Figure 4.104 – River-stream simulation. Case 2, the surface of the bed when $t^* \in [600, 800]$. The x-axis is the relative longitudinal location X/d_*, $X = x - u_d t$. x is the horizontal direction u_d is the migration velocity of the bedform ($K_r = 10^{-6}$).

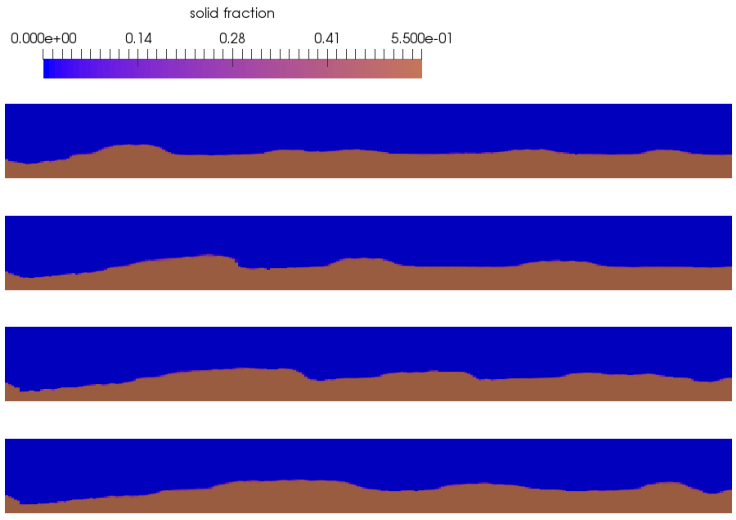


Figure 4.105 – River-stream simulation. Case 1, with smaller mean velocity, top to bottom: snapshots of sediment profiles for $t = 5$ [s], $t = 10$ [s], $t = 15$ [s] and $t = 20$ [s] ($K_r = 10^{-6}$).

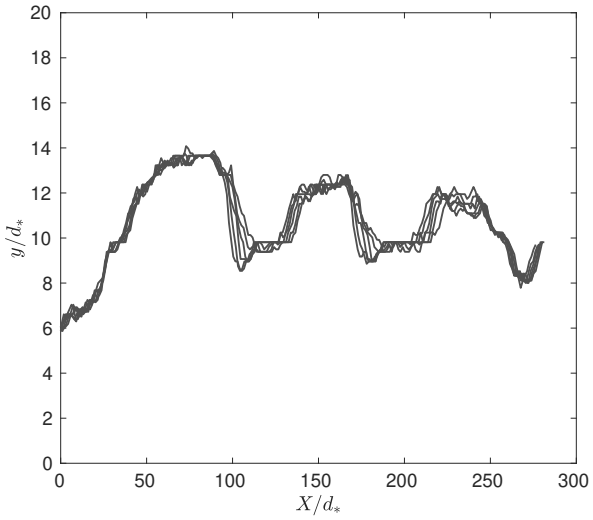


Figure 4.106 – River-stream simulation. Case 1 with smaller mean velocity, the surface of the bed when $t^* \in [600, 800]$. The x-axis is the relative longitudinal location X/d_*, $X = x - u_d t$. x is the horizontal direction u_d is the migration velocity of the bedform ($K_r = 10^{-6}$).

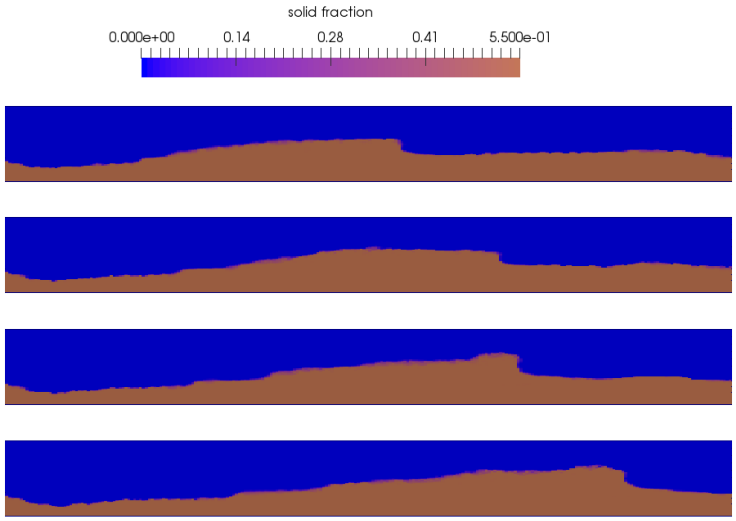


Figure 4.107 – River-stream simulation. Case 4, top to bottom: snapshots of sediment profiles for $t = 3$ [s], $t = 4.5$ [s], $t = 5$ [s] and $t = 6.5$ [s] ($K_r = 10^{-6}$).

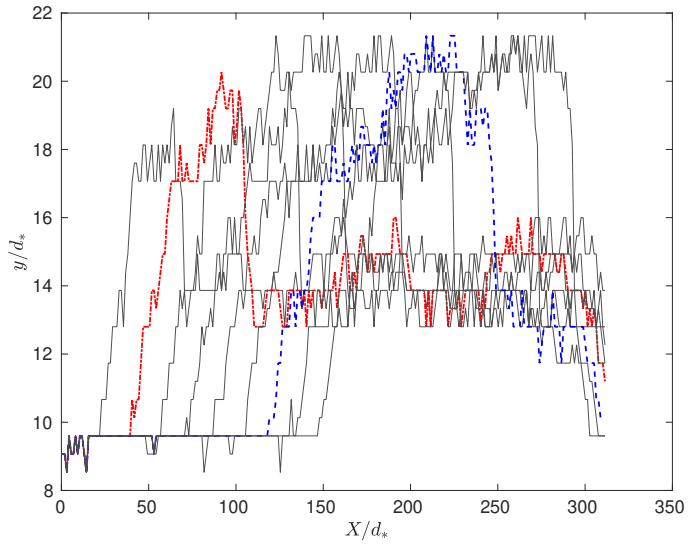


Figure 4.108 – River-stream simulation. Case 3, the surface of the bed when $t^* \in [600, 800]$. The x-axis is the relative longitudinal location X/d_*, $X = x - u_d t$. x is the horizontal direction u_d is the migration velocity of the bedform. The red dune is at $t^* = 600$ and the blue dune is for $t^* = 700$ ($K_r = 10^{-6}$).

In a second stage, we now consider the same experimental setup, but we introduce sediments along with the liquid through the inflow surface: the inflow mean velocity is 0.35 [m/s] and the sediment inflow concentration is 0.025×2500 [kg/m^3] . We run the experiment up to 90 [s], with $K_r = 0$ then with $K_r = 10^{-6}$. Free outflow conditions are still applied on the right boundary surface of the domain. We observe the sediment profile as well as the conservation of the sediment mass over time (the mass is conserved if the final mass inside the domain is equal to the total intorduced mass over 90 [s] added to the initial mass, minus the total mass that flew out of the domain 90 [s]). In Tables 4.10 and 4.11, we show the expected and obtained final masses. The expected mass is computed as:

$$\text{Final mass (expected)} = \sum_{n=0}^{N} m_{in}(t^n) + m_{t^0} - \sum_{n=0}^{N} m_{out}(t^n),$$

whereas the final mass is the sediment mass that actually exists inside the domain at the final time. The notations are introduced in the last section of Chapter 2. Both tables show that the expected and the final sediment mass are roughly equal. This confirms the mass conservation of our algorithm. In Figures 4.109 and 4.111, we illustrate the inflow and the outflow masses over 90 [s] with and without resuspension respectively. However, we can see that the final mass obtained without resuspension is slightly bigger than that obtained with resuspension, which is expected. From these figures, we observe that, until roughly 50 [s] without resuspension (and 30 [s] with resuspension) the outflow mass is minimal, and increases afterward. In Figures 4.110 and 4.112, we show snapshots of the sediment profile

over time with and without resuspension respectively.

Table 4.10 – River-stream simulation with sediment inflow without resuspension ($K_r = 0$). Sediment mass (in [kg]) conservation. The final time is $T = 90$ [s]. The notations are introduced in the last section of Chapter 2.

$m_{in}(t_n)$	m_0	Final mass (expected)	Final mass (obtained)
2.5356×10^{-7}	0.0028598	0.00509348	0.00518815

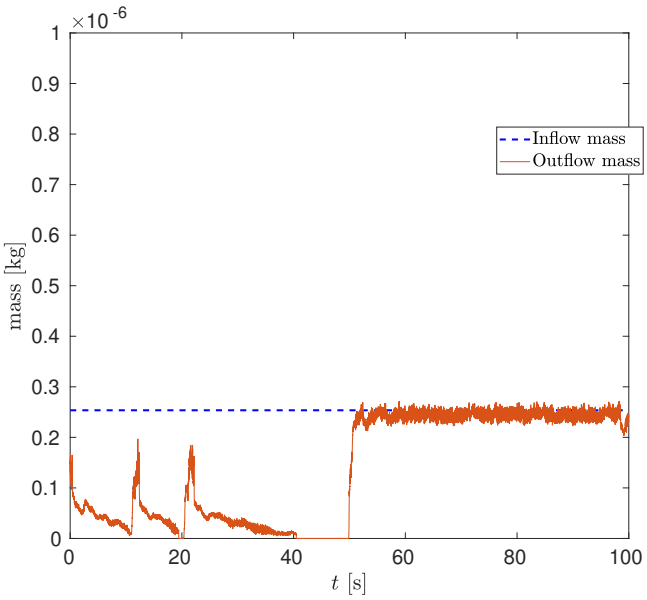


Figure 4.109 – River-stream simulation with sediment inflow without resuspension ($K_r = 0$). Sediment inflow and outflow masses over time.

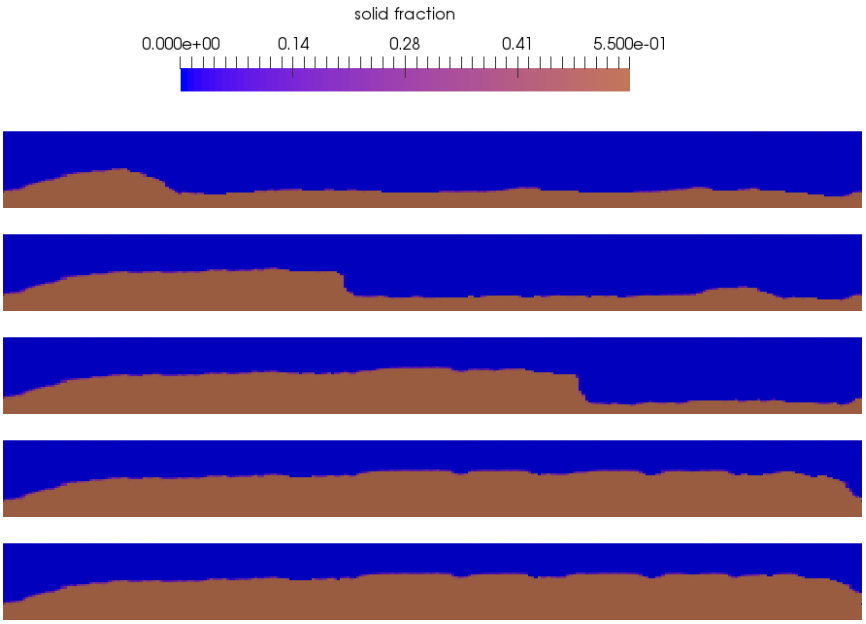


Figure 4.110 – River-stream simulation with sediment inflow without resuspension ($K_r = 0$). Top to bottom: snapshots of the sediment profile for $t = 5$ [s], $t = 15$ [s], $t = 30$ [s], $t = 50$ [s] and $t = 90$ [s].

Table 4.11 – River-stream simulation with sediment inflow with resuspension ($K_r = 10^{-6}$). Sediment mass (in [kg]) conservation. The final time is $T = 90$ [s]. The notations are introduced in the last section of Chapter 2.

$m_{in}(t_n)$	m_0	Final mass (expected)	Final mass (obtained)
2.53562×10^{-7}	0.00285985	0.00429016	0.00429011

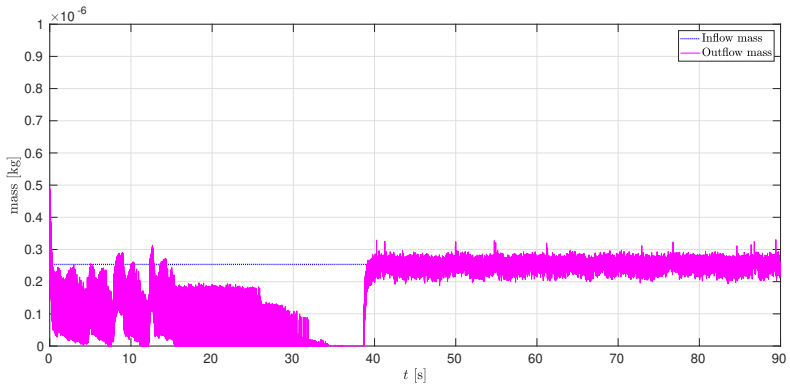


Figure 4.111 – River-stream simulation with sediment inflow with resuspension ($K_r = 10^{-6}$). Sediment inflow and outflow masses over time.

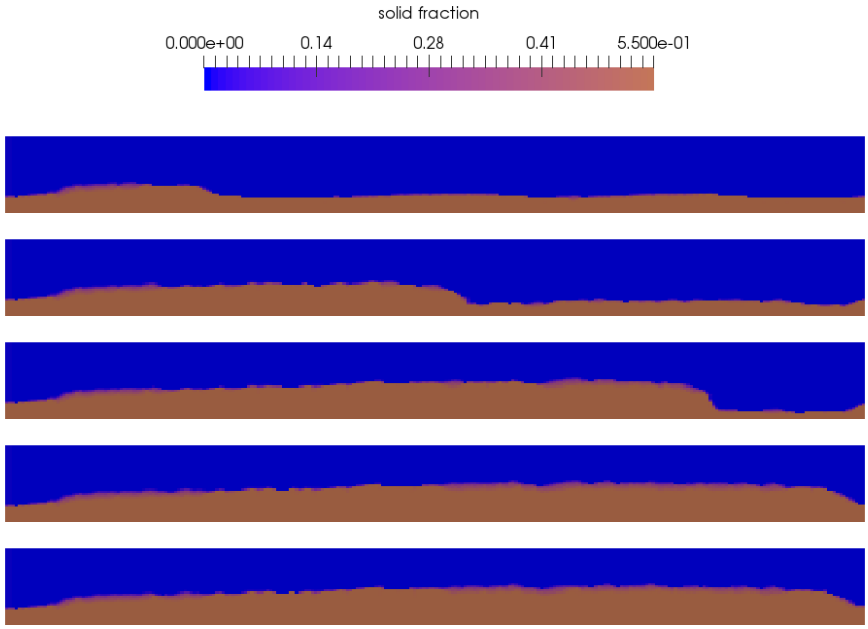


Figure 4.112 – River-stream simulation with sediment inflow. Top to bottom: snapshots of the sediment profile for $t = 5$ [s], $t = 15$ [s], $t = 30$ [s], $t = 50$ [s] and $t = 90$ [s].

4.2.8 Remarks and conclusion

We revisit the erosion by an impinging liquid jet experiment, introduced in Subsection 4.1.3 when we introduce the resuspension effects. Using the intermediate mesh, also presented in Subsection 4.1.3, we run the experiment with resuspension effects ($K_r = 10^{-4}$ and $\varepsilon = 10^{-11}$)

and compare it to the results without resuspension ($\varepsilon = 10^{-9}$), for the mean velocity $U_J = 0.28$ [m/s]. In Figure 4.113, we show a snapshot at $t = 3.5$ [s] and in Figure 4.114, we show a plot of the sediment profile, with and without resuspension. The results with resuspension slightly differ from those without resuspension effects (and lower penalization effects). However, the resuspension promotes the scouring effect, even with higher ε values. Furthermore, in Figure 4.115 we illustrate the values of the width D and depth H of the eroded hole as a function of the inlet velocity magnitude, with and without resuspension. We observe that the results with resuspension effects, and higher penalization, still fit around the regression lines and experimental results from [148, 149].

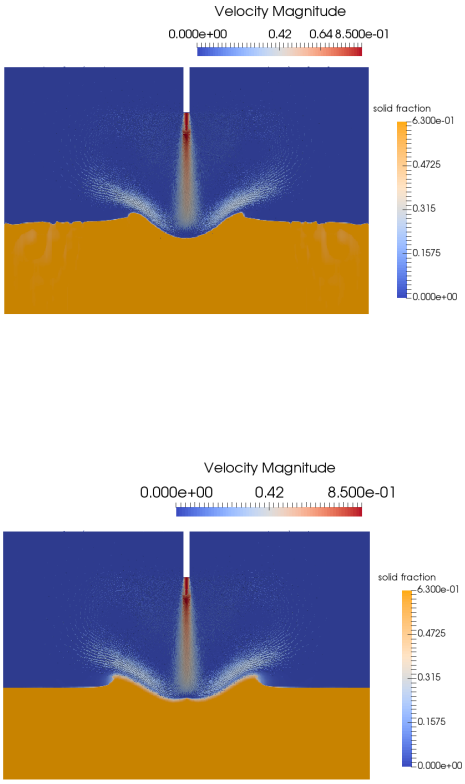


Figure 4.113 – Erosion by an impinging liquid jet. Snapshot of the sediment profile at $t = 3.5$ [s] for $U_J = 0.28$ [m/s]. Top: without resuspension ($\varepsilon = 10^{-9}$, $K_r = 0$). Bottom: with resuspension ($\varepsilon = 10^{-11}$, $K_r = 10^{-4}$, $\tau_{CR} = 0.2$ [N/m^2]).

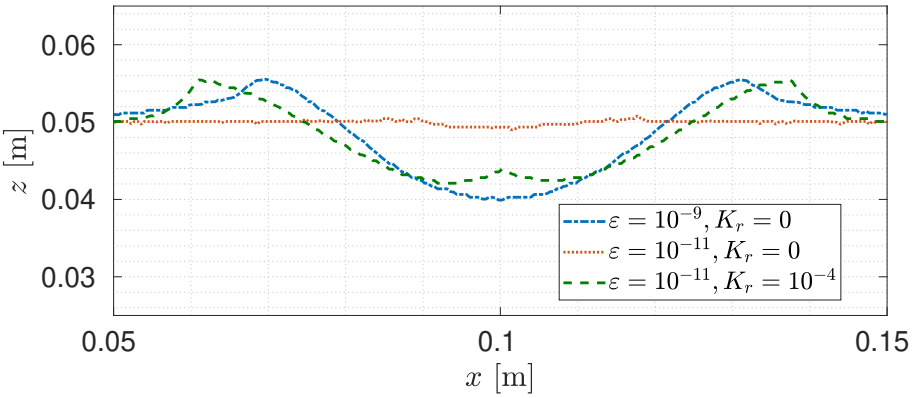


Figure 4.114 – Erosion by an impinging liquid jet. Sediment profile at $t = 3.5$ [s] with and without resuspension effects ($U_J = 0.28$ [m/s], $f_{s_{CO}} = 0.62$).

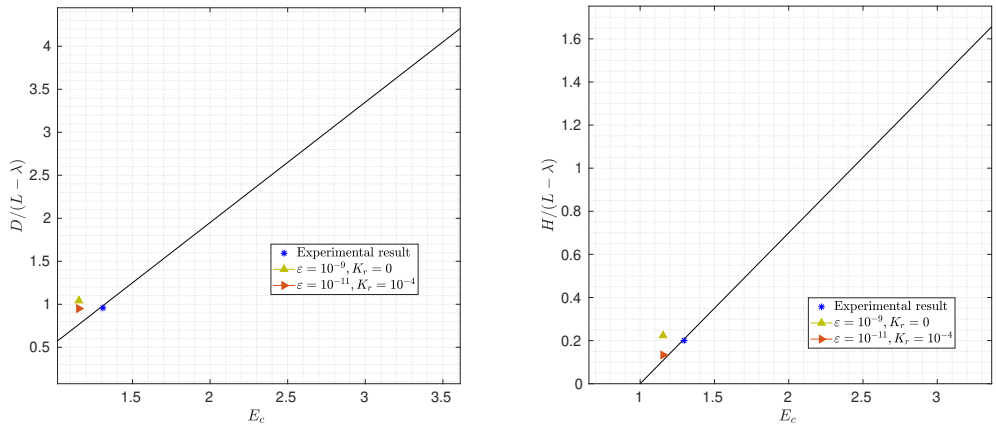


Figure 4.115 – Erosion by an impinging liquid jet. Numerical approximation of the width D and depth H of the eroded hole, as a function of the inlet velocity magnitude with and without resuspension effects; the regression lines are extracted from [149] ($\lambda = 10b$, ($U_J = 0.28$ [m/s], $f_{s_{CO}} = 0.62$).

To summarize, from all of the numerical experiments that we have studied, it appears that for regimes where the main physical phenomenon is vertical erosion, deposition effects only are enough, and the optimal parameters would be $\varepsilon = 10^{-9}$ and $f_{s_{CO}} = f_{s_{CR}} - 0.1$. On the other hand, for experiments where horizontal erosion and flushing are the main phenomena, the introduction of the resuspension effects is necessary. In this case, the optimal parameters are $\varepsilon = 10^{-11}$ (same or larger $f_{s_{CO}}$) and τ_{CR} between 0.05 and 0.4 [N/m^2]. The resuspension value

K_r is often chosen between 10^{-2} (for flushing experiments) and 10^{-4} or smaller (otherwise).

5 Error indicators for the 1D diffusion-convection equation

In Subsection 3.1 of Chapter 3, we presented the splitting scheme that we used to solve the set of equations of the presented problem. In this chapter, we consider the 1D diffusion-convection equation and aim to study an error estimate for a splitting scheme. First, we study an a posteriori error estimate for a finite element backward scheme used to discretize and solve the equation. We use the error indicator to apply an adaptive algorithm to the same scheme. Second, we write a splitting scheme that separates the diffusion and the convection parts of the equation. For the splitting scheme, we write an a posteriori error indicator and use the latter for the adaptive algorithm. We consider the following 1D diffusion-convection equation: Given $f : \mathscr{C}^0[0,1] \to L^2(0,1)$ and $u_0 \in \mathscr{C}^0[0,1]$, find $u : [0,1] \times [0,T] \to \mathbb{R}$ such that

$$\frac{\partial u}{\partial t}(x,t) - \varepsilon \frac{\partial^2 u}{\partial x^2}(x,t) + c\frac{\partial u}{\partial x}(x,t) = f, \quad 0 \le t \le T, \quad 0 < x < 1, \tag{5.1}$$

where $T, \varepsilon, c > 0$ and with $u(x,0) = u_0(x)$, $u(0,t) = 0$ and $u(1,t) = 0$. Let $0 = t_0 < t_1 < ... < t_N < t_{N+1} = T$ be a discretization of $[0,T]$ with time steps $\Delta t^n = t_{n+1} - t_n$, $n = 0,1,...,N$ and $\Delta t = \max_n \Delta t^n$. Let $0 = x_0 < x_1 < ... < x_M < x_{M+1} = 1$ be a discretization of $[0,1]$, let $h_i = x_{i+1} - x_i$ for $i = 0,...,M$, and $h = \max_i h_i$. Let V_h be the finite dimensional space $V_h = span(\varphi_1, \varphi_2, ..., \varphi_M)$, where the M affine shape functions φ_i, $i = 1,...,M$, are defined by:

$$\varphi_i(x) := \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } x \in [x_{i-1}, x_i] \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise.} \end{cases}$$

We initialize $u_h^0 := \Pi_h u_0$ (the interpolant of u_0 on V_h) and compute u_h^n the finite element approximation of the solution u at time t_n, $n = 1,...,N$.

5.1 The backward Euler scheme:

The fully discretized problem with backward Euler scheme reads: for $n = 0, 1, ..., N$, knowing $u_h^n \in V_h$, find $u_h^{n+1} \in V_h$ such that

$$\int_0^1 \frac{u_h^{n+1} - u_h^n}{\Delta t^n} v_h + \varepsilon \int_0^1 \frac{du_h^{n+1}}{dx} \frac{dv_h}{dx} + c \int_0^1 \frac{du_h^{n+1}}{dx} v_h = \int_0^1 f(t_{n+1}) v_h \quad \forall v_h \in V_h. \tag{5.2}$$

Setting $v_h = \varphi_i$, $i = 1, ..., M$, in (5.2), we obtain:

$$\frac{1}{\Delta t^n} \sum_{j=1}^{M} u_j^{n+1} \int_0^1 \varphi_j \varphi_i - \frac{1}{\Delta t} \sum_{j=1}^{M} u_j^n \int_0^1 \varphi_j \varphi_i + \varepsilon \sum_{j=1}^{M} u_j^{n+1} \int_0^1 \varphi_j' \varphi_i' + c \sum_{j=1}^{M} u_j^{n+1} \int_0^1 \varphi_j' \varphi_i = \int_0^1 f(t_{n+1}) \varphi_i, \tag{5.3}$$

Thus, the differential system writes as:

$$\frac{1}{\Delta t} D\left(\vec{u}^{n+1} - \vec{u}^n\right) + A\vec{u}^{n+1} + B\vec{u}^{n+1} = \vec{f}^{n+1}, \tag{5.4}$$

where: $u_i^{n+1} = u_i$, $D_{ij} = \int_0^1 \varphi_j \varphi_i$, $A_{ij} = \varepsilon \int_0^1 \varphi_j' \varphi_i'$, $B_{ij} = c \int_0^1 \varphi_j' \varphi_i$ and $f_i^{n+1} = \int_0^1 f(t_{n+1}) \varphi_i$. We precice that u^{n+1} is well defined as solution to (5.4) whatever c and Δt.

5.1.1 A posteriori error estimate

Let us define the approximation $u_{h\Delta t}$ of u on each subinterval $[t_n, t_{n+1}]$, $n = 0, ..., N$, by

$$u_{h\Delta t}(\mathbf{x}, t) := \frac{t - t_n}{\Delta t^n} u_h^{n+1}(\mathbf{x}) + \frac{t_{n+1} - t}{\Delta t^n} u_h^n(\mathbf{x}),$$

Thus $\dfrac{\partial u_{h\Delta t}}{\partial t} = \dfrac{u_h^{n+1} - u_h^n}{\Delta t^n}$. Setting $e = u - u_{h\Delta t}$, In order to compute an a posteriori error estimate, we start from:

$$\int_0^1 \frac{\partial e}{\partial t} e + \varepsilon \int_0^1 \left(\frac{\partial e}{\partial x}\right)^2 + c \underbrace{\int_0^1 \left(\frac{\partial e}{\partial x}\right) e}_{=0} = \int_0^1 \left(f - \frac{\partial u_{h\Delta t}}{\partial t} - c \frac{\partial u_{h\Delta t}}{\partial x}\right) e - \varepsilon \int_0^1 \frac{\partial u_{h\Delta t}}{\partial x} \left(\frac{\partial e}{\partial x}\right), \tag{5.5}$$

We can rewrite (5.2) as

$$\begin{aligned} \int_0^1 \frac{\partial u_{h\Delta t}}{\partial t} v_h \quad + \quad & \varepsilon \int_0^1 \frac{\partial u_{h\Delta t}}{\partial x} \left(\frac{\partial v_h}{\partial x}\right) + c \int_0^1 \left(\frac{\partial u_{h\Delta t}}{\partial x}\right) v_h = \int_0^1 f v_h + \int_0^1 (f(t_{n+1}) - f) v_h \\ & + \varepsilon \int_0^1 \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x} \frac{\partial v_h}{\partial x} + c \int_0^1 \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x} v_h \quad \forall v_h \in V_h. \end{aligned}$$

We get:

$$\int_0^1 \frac{\partial e}{\partial t} e + \varepsilon \int_0^1 \left(\frac{\partial e}{\partial x}\right)^2 = \int_0^1 \left(f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x}\right)(e - v_h) - \varepsilon \int_0^1 \frac{\partial u_{h\Delta t}}{\partial x}\frac{\partial (e - v_h)}{\partial x} + \int_0^1 \left(f - f(t_{n+1})\right) v_h - \varepsilon \int_0^1 \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x}\frac{\partial v_h}{\partial x} - c\int_0^1 \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x} v_h,$$

which leads to:

$$\begin{aligned}\int_0^1 \frac{\partial e}{\partial t} e + \varepsilon \int_0^1 \left(\frac{\partial e}{\partial x}\right)^2 = \sum_{i=0}^{M} \Bigg[&\int_{x_i}^{x_{i+1}} \left(f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2}\right)(e - v_h) \\ &- \left[\varepsilon \frac{\partial u_{h\Delta t}}{\partial x}(e - v_h)\right]_{x=x_i}^{x=x_{i+1}} + \int_{x_i}^{x_{i+1}} \left(f - f(t_{n+1})\right) v_h \\ &- \varepsilon \int_{x_i}^{x_{i+1}} \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x}\frac{\partial v_h}{\partial x} - c\int_{x_i}^{x_{i+1}} \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x} v_h \Bigg],\end{aligned} \tag{5.6}$$

for any $v_h \in V_h$. Assuming that u is sufficiently smooth, and choosing $v_h = R_h e$, with R_h being the Lagrange interpolant, the jump terms vanish. We then use the interpolation results:

$$\frac{1}{h_i}\|e - R_h e\| + \left\|\frac{\partial R_h e}{\partial x}\right\|_{L^2(x_i, x_{i+1})} \leq C \left\|\frac{\partial e}{\partial x}\right\|_{L^2(x_i, x_{i+1})}, \tag{5.7}$$

where C is a constant independent of h, Δt and u.
Let $R = (\frac{\partial e}{\partial t}, e) + \varepsilon \int_0^1 \left(\frac{\partial e}{\partial x}\right)^2$. Applying Cauchy-Schwarz inequality to (5.6) we get:

$$\begin{aligned}R \leq &\sum_{i=0}^{M} \left\|f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2}\right\|_{L^2(x_i, x_{i+1})} \|e - R_h e\|_{L^2(x_i, x_{i+1})} \\ &+ \sum_{i=0}^{M} \left\|(f - f(t_{n+1})) - c\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1})\right\|_{L^2(x_i, x_{i+1})} \|R_h e\|_{L^2(x_i, x_{i+1})} \\ &+ \sum_{i=0}^{M} \varepsilon \left\|\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1})\right\|_{L^2(x_i, x_{i+1})} \left\|\frac{\partial R_h e}{\partial x}\right\|_{L^2(x_i, x_{i+1})}.\end{aligned}$$

Using (5.7) and the Poincaré inequality, there exists C independent of h and u such that

$$R \leq C\left(\left(\sum_{i=0}^{M} \eta_{S,i}^2\right)^{1/2} + \left(\sum_{i=0}^{M} \eta_{T,i}^2\right)^{1/2}\right)\left\|\frac{\partial e}{\partial x}\right\|_{L^2(0,1)}^2$$

where

$$\eta_{S,i}^2 = \frac{1}{\varepsilon}\left(h_i \left\|f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2}\right\|_{L^2(x_i, x_{i+1})}\right)^2$$

and

$$\eta_{T,i}^2 = \frac{1}{\varepsilon}\left(\left\|(f - f(t_{n+1})) - c\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1})\right\|_{L^2(x_i,x_{i+1})} + \varepsilon\left\|\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1})\right\|_{L^2(x_i,x_{i+1})}\right)^2$$

Using Young's inequality we get:

$$R \leq \tilde{C}\sum_{i=0}^{M}\eta_{S,i}^2 + \frac{\varepsilon}{4}\left\|\frac{\partial e}{\partial x}\right\|_{L^2(x_i,x_{i+1})}^2 + \tilde{C}\sum_{i=0}^{M}\eta_{T,i}^2 + \frac{\varepsilon}{4}\left\|\frac{\partial e}{\partial x}\right\|_{L^2(x_i,x_{i+1})}^2$$

Finally we get:

$$\frac{1}{2}\frac{d}{dt}\|e\|_{L^2(0,1)}^2 + \frac{\varepsilon}{2}\left\|\frac{\partial e}{\partial x}\right\|_{L^2(0,1)}^2 \leq \tilde{C}\sum_{i=0}^{M}\left(\eta_{S,i}^2 + \eta_{T,i}^2\right)$$

we multiply integrate between t_n and t_{n+1} to obtain:

$$\|e(t_{n+1})\|_{L^2(0,1)}^2 + \varepsilon\int_{t_n}^{t_{n+1}}\left\|\frac{\partial e}{\partial x}\right\|_{L^2(0,1)}^2 \leq \|e(t_n)\|_{L^2(0,1)}^2 + \tilde{C}\sum_{i=0}^{M}\int_{t_n}^{t_{n+1}}\left(\eta_{S,i}^2 + \eta_{T,i}^2\right)$$

For later use we define the error as:

$$\text{Error} = \left(\|e(T)\|_{L^2(0,1)}^2 - \|e(0)\|_{L^2(0,1)}^2 + \varepsilon\int_0^T\left\|\frac{\partial e}{\partial x}\right\|_{L^2(0,1)}^2\right)^{1/2}$$

the estimator as:

$$\text{Estimator} = \left(\eta_S^2 + \eta_T^2\right)^{1/2}$$

where

$$\eta_S = \left(\sum_{n=0}^{N}\sum_{i=0}^{M}\int_{t_n}^{t_{n+1}}\eta_{S,i}^2\right)^{1/2}, \tag{5.8}$$

$$\eta_T = \left(\sum_{n=0}^{N}\sum_{i=0}^{M}\int_{t_n}^{t_{n+1}}\eta_{T,i}^2\right)^{1/2}, \tag{5.9}$$

The effectivity index is:

$$e_I = \frac{\text{Estimator}}{\text{Error}}.$$

5.1.2 Numerical results

To test the numerical scheme presented above and check the sharpness of the a posteriori error indicators, we proceed with some numerical experiments. We construct the function f such that the exact solution is:

(a) $u(x,t) = sin(\omega_1\pi x)$,

(b) $u(x,t) = sin(\omega_1\pi x)sin(\omega_2\pi t)$, where ω_1, ω_2 are positive parameters.

Unless specified differently, the final step for which we display the results is $T = 1.55$ [s].

Effectivity indices with respect to h and Δt. Before we observe the effect of h and Δt on the effectivity index, we want to check the sharpness of the time-estimator η_T separately from the space-estimator η_S. In order to do this, we consider first the case (a) with $\omega_1 = 1$ then the case (b) with $\omega_1 = 1$ and $\omega_2 = 10$. We take $\varepsilon = c = 1$. For the sharpness of η_S, we consider the case (a). In this case, the error in time is negligible. The values of η_T in Table 5.1 confirm this. The error due to time discretization is very small and the effectivity index of the space estimator is roughly a constant equal to 3.46. Similarly, for the sharpness of η_T, as we can see in Table 5.2, that the error due to the space discretization is very small. The effectivity of the time indicator is somewhere between 13 and 15. To further assess the sharpness of η_T, we consider the same case, but we take $h = O(\Delta t^2)$. The results in Table 5.3 show that the effectivity index converges to 15.
In order to obtain an effectivity index close to one, we divide η_S by C_s and η_T by C_t, where $C_s = 3.46$ and $C_t = 15$. We denote the normalized error estimator η_N:

$$\eta_N = \sqrt{\frac{(\eta_S)^2}{C_s^2} + \frac{(\eta_T)^2}{C_t^2}},$$

and the normalized effectivity index is

$$\tilde{e_I} = \frac{\eta_N}{\text{Error}}.$$

We vary ω_1 and ω_2 and check the normalized effectivity indices and the errors when we vary the discretization parameters h and Δt. We take $\varepsilon = c = 1$. The results are illustrated in Tables 5.4 to 5.8. The first observation is that the effectivity for all cases remains between 1 and 2. Furthermore, we examine the two opposite scenarios: in Table 5.5 we show the results when $\omega_1 = 1$ and $\omega_2 = 10$, whereas in Table 5.6 ,we illustrate the results when $\omega_1 = 10$ and $\omega_2 = 1$. Looking at Table 5.5, where the error mainly comes from time discretization: we observe that the true error halves only when the time step halves. We can see that η_T follows the true error, whereas η_S does not, every time the time step is divided by two. Similar observations are made, looking at Table 5.6. We can see that the true error divides by two when h divides by two. However, when we divide Δt by two, the true error barely changes. In fact, in this case, the error mainly comes from the space discretization. As we can see, η_S follows the true error, whereas η_T does not. However, when h and Δt are both divided by two, we can observe the proper order of convergence for both cases. In Tables 5.7 ands 5.8, we consider the cases where $\omega_1 = \omega_2$. The first observation is that, for the same discretization parameters, the true error is higher when $\omega_1 = \omega_2 = 10$ than it is when $\omega_1 = \omega_2 = 1$. In Figure 5.1, we show that for $\Delta t = 0.1$, the error is too big for any of the results to be significant, no matter what h is. For $\Delta t = 0.05$, we plot the same results in Figure 5.2. The second observation that we can make looking at both tables is that, the spatial estimator η_S halves when h halves, and the time estimator η_T halves when Δt halves. Furthermore, in Table 5.6, we can see that as h and Δ converge to zero, the effectivity index converges to 1, even though it takes longer than previous cases.

Table 5.1 – **Case (a)** with $\omega_1 = 1$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	e_I	η_S	η_T
0.050000	0.050000	0.119450	0.413928	3.465272	0.413927	0.000632
0.025000	0.050000	0.061180	0.211942	3.464249	0.211942	0.000166
0.012500	0.050000	0.030966	0.107270	3.464118	0.107270	0.000042
0.006250	0.050000	0.015579	0.053967	3.464103	0.053967	0.000011
0.050000	0.025000	0.119452	0.413927	3.465233	0.413927	0.002846
0.025000	0.025000	0.061180	0.211942	3.464238	0.211942	0.001458
0.012500	0.025000	0.030966	0.107270	3.464115	0.107270	0.000738
0.006250	0.025000	0.015579	0.053967	3.464102	0.053967	0.000371
0.050000	0.012500	0.119452	0.413927	3.465214	0.413927	0.001491
0.025000	0.012500	0.061180	0.211942	3.464233	0.211942	0.000764
0.012500	0.012500	0.030966	0.107270	3.464114	0.107270	0.000387
0.006250	0.012500	0.015579	0.053967	3.464102	0.053967	0.000195
0.050000	0.006250	0.119453	0.413927	3.465204	0.413927	0.000768
0.025000	0.006250	0.061180	0.211942	3.464230	0.211942	0.000393
0.012500	0.006250	0.030966	0.107270	3.464113	0.107270	0.000199
0.006250	0.006250	0.015579	0.053967	3.464102	0.053967	0.000100

Table 5.2 – **Case (b)**, $\omega_1 = 1$, $\omega_2 = 10$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	e_I	η_S	η_T
0.020000	0.020000	0.246757	3.486445	14.129063	0.288260	3.474508
0.010000	0.020000	0.249795	3.477145	13.919976	0.145530	3.474098
0.005000	0.020000	0.251404	3.474751	13.821388	0.073123	3.473982
0.002500	0.020000	0.252231	3.474141	13.773656	0.036652	3.473947
0.001250	0.020000	0.252650	3.473987	13.750197	0.018349	3.473938
0.000625	0.020000	0.252861	3.473947	13.738570	0.009180	3.473935
0.020000	0.010000	0.127089	1.852024	14.572673	0.304407	1.826836
0.010000	0.010000	0.128266	1.833102	14.291427	0.153680	1.826648
0.005000	0.010000	0.128999	1.828222	14.172329	0.077218	1.826590
0.002500	0.010000	0.129403	1.826984	14.118572	0.038705	1.826574
0.001250	0.010000	0.129614	1.826672	14.093187	0.019377	1.826569
0.000625	0.010000	0.129722	1.826594	14.080873	0.009694	1.826568
0.020000	0.005000	0.065265	0.968869	14.845179	0.311453	0.917444
0.010000	0.005000	0.065060	0.930731	14.305696	0.157236	0.917353
0.005000	0.005000	0.065229	0.920721	14.115297	0.079005	0.917325
0.002500	0.005000	0.065382	0.918171	14.043116	0.039601	0.917316
0.001250	0.005000	0.065477	0.917528	14.013062	0.019825	0.917314
0.000625	0.005000	0.065528	0.917367	13.999571	0.009919	0.917313
0.020000	0.002500	0.034911	0.556868	15.951184	0.314750	0.459385
0.010000	0.002500	0.033250	0.486048	14.618082	0.158901	0.459340
0.005000	0.002500	0.032923	0.466214	14.160840	0.079841	0.459327
0.002500	0.002500	0.032896	0.461062	14.015941	0.040020	0.459322
0.001250	0.002500	0.032917	0.459758	13.967245	0.020035	0.459321
0.000625	0.002500	0.032936	0.459430	13.949057	0.010024	0.459321
0.020000	0.001250	0.021236	0.391007	18.412736	0.316341	0.229816
0.010000	0.001250	0.017739	0.279840	15.775112	0.159703	0.229794
0.005000	0.001250	0.016784	0.243395	14.501893	0.080245	0.229787
0.002500	0.001250	0.016561	0.233279	14.085741	0.040222	0.229785
0.001250	0.001250	0.016519	0.230665	13.963562	0.020136	0.229784
0.000625	0.001250	0.016516	0.230005	13.926594	0.010074	0.229784
0.020000	0.000625	0.016081	0.337307	20.975869	0.317121	0.114933
0.010000	0.000625	0.010769	0.197074	18.300918	0.160097	0.114923
0.005000	0.000625	0.008943	0.140276	15.686439	0.080443	0.114919
0.002500	0.000625	0.008432	0.121787	14.443195	0.040321	0.114918
0.001250	0.000625	0.008306	0.116677	14.047555	0.020186	0.114918
0.000625	0.000625	0.008277	0.115361	13.936720	0.010099	0.114918

Table 5.3 – **Case (b)**, $\omega_1 = 1$, $\omega_2 = 10$, $\varepsilon = c = 1$ with $h = O(\Delta t^2)$

h	Error	Estimator	e_I	η_S	η_T
0.010000	1.788253	23.761076	13.287313	0.185731	23.760350
0.005000	1.096090	15.668035	14.294472	0.076069	15.667850
0.002500	0.824558	12.514550	15.177287	0.044613	12.514471
0.001250	0.602547	9.090158	15.086226	0.024714	9.090125
0.000625	0.439415	6.749185	15.359473	0.013208	6.749172
0.000313	0.316370	4.833476	15.277937	0.006877	4.833471
0.000156	0.226767	3.451718	15.221415	0.003528	3.451716

Table 5.4 – **Case (a)**, $\omega_1 = 1$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.100000	0.100000	0.081661	0.284834	0.284806	0.004010	0.996477
0.050000	0.100000	0.042880	0.148779	0.148775	0.001099	0.991303
0.025000	0.100000	0.021973	0.076144	0.076144	0.000288	0.990112
0.012500	0.100000	0.011123	0.038534	0.038534	0.000074	0.989832
0.100000	0.050000	0.091343	0.318378	0.318370	0.002288	0.995834
0.050000	0.050000	0.047948	0.166330	0.166328	0.000627	0.991121
0.025000	0.050000	0.024567	0.085130	0.085130	0.000164	0.990064
0.012500	0.050000	0.012436	0.043082	0.043082	0.000042	0.989820
0.100000	0.025000	0.091356	0.318397	0.318395	0.001234	0.995777
0.050000	0.025000	0.047950	0.166332	0.166332	0.000338	0.991104
0.025000	0.025000	0.024567	0.085131	0.085131	0.000089	0.990059
0.012500	0.025000	0.012436	0.043082	0.043082	0.000023	0.989819
0.100000	0.012500	0.091362	0.318408	0.318407	0.000647	0.995744
0.050000	0.012500	0.047951	0.166334	0.166334	0.000177	0.991094
0.025000	0.012500	0.024567	0.085131	0.085131	0.000047	0.990056
0.012500	0.012500	0.012436	0.043082	0.043082	0.000012	0.989818

Table 5.5 – **Case (b)**, $\omega_1 = 1, \omega_2 = 10$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.050000	0.050000	0.568149	8.166865	0.562830	8.147448	0.997975
0.025000	0.050000	0.585300	8.148026	0.288027	8.142934	0.938335
0.012500	0.050000	0.594449	8.142624	0.145758	8.141319	0.915784
0.006250	0.050000	0.599170	8.141212	0.073327	8.140882	0.906486
0.050000	0.025000	0.297041	4.491089	0.680912	4.439171	1.196480
0.025000	0.025000	0.305374	4.450912	0.348393	4.437256	1.023283
0.012500	0.025000	0.310071	4.440094	0.176298	4.436592	0.967937
0.006250	0.025000	0.312555	4.437289	0.088690	4.436403	0.949814
0.050000	0.012500	0.154569	2.391624	0.731128	2.277129	1.683309
0.025000	0.012500	0.157033	2.306676	0.374055	2.276145	1.186475
0.012500	0.012500	0.158999	2.283663	0.189280	2.275806	1.014357
0.006250	0.012500	0.160171	2.277691	0.095220	2.275700	0.962653
0.050000	0.006250	0.083298	1.371364	0.753238	1.145981	2.769745
0.025000	0.006250	0.080831	1.208568	0.385352	1.145488	1.670644
0.012500	0.006250	0.080843	1.161800	0.194994	1.145320	1.173885
0.006250	0.006250	0.081189	1.149460	0.098095	1.145266	1.003155

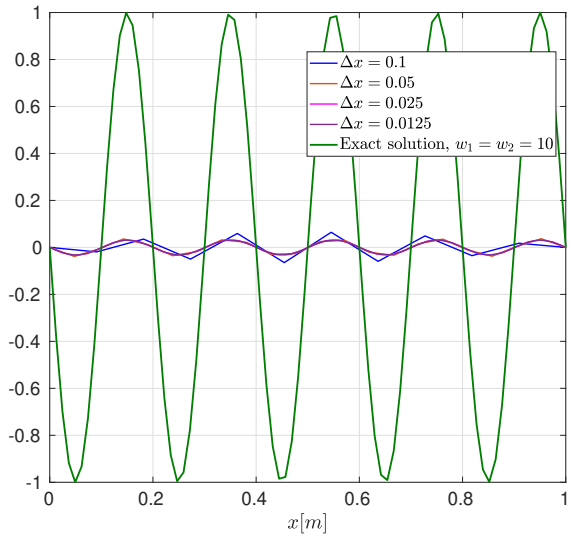


Figure 5.1 – **Case (b)**, $\omega_1 = \omega_2 = 10$ and $\varepsilon = c = 1$ - Exact and approximated solutions for $\Delta t = 0.1$, $\varepsilon = c = 1$.

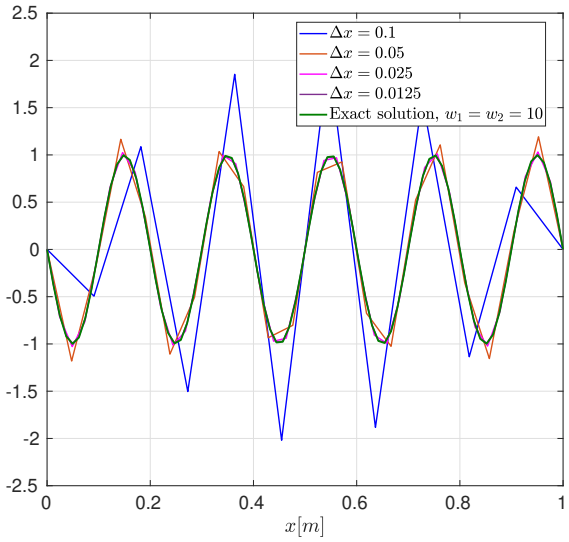


Figure 5.2 – **Case (b)**, $\omega_1 = \omega_2 = 10$ and $\varepsilon = c = 1$ - Exact and approximated solutions for $\Delta t = 0.05$, $\varepsilon = c = 1$.

Table 5.6 – **Case (b)**, $\omega_1 = 10$, $\omega_2 = 1$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	η_S	η_T	$\tilde{e_I}$
0.100000	0.050000	11.892154	4.498250	34.943735	40.503470	0.879075
0.050000	0.050000	5.375384	8.162049	18.258351	39.894531	1.099332
0.025000	0.050000	2.704819	15.054156	9.346258	39.631621	1.396967
0.012500	0.050000	1.362103	29.222831	4.730099	39.522454	2.179258
0.100000	0.025000	11.913302	3.391628	34.952880	20.270655	0.855512
0.050000	0.025000	5.384178	5.025614	18.263140	19.965882	1.011037
0.025000	0.025000	2.708996	8.094180	9.348711	19.834298	1.110427
0.012500	0.025000	1.363959	14.910757	4.731340	19.779659	1.392755
0.100000	0.012500	11.918894	3.053602	34.955164	10.137703	0.849511
0.050000	0.012500	5.386535	3.864387	18.264335	9.985276	0.987744
0.025000	0.012500	2.710151	5.029629	9.349323	9.919467	1.026459
0.012500	0.012500	1.364529	8.036129	4.731650	9.892140	1.112645
0.100000	0.006250	11.920439	2.963094	34.955737	5.069151	0.847994
0.050000	0.006250	5.387197	3.514776	18.264634	4.992932	0.981824
0.025000	0.006250	2.710481	3.904728	9.349476	4.960025	1.004367
0.012500	0.006250	1.364697	5.015856	4.731727	4.946360	1.030812

Table 5.7 – **Case (b)** for $\omega_1 = \omega_2 = 1$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.100000	0.050000	0.100818	0.643616	0.489334	0.418084	1.429774
0.050000	0.050000	0.056661	0.488642	0.256002	0.416213	1.394631
0.025000	0.050000	0.036067	0.435716	0.131078	0.415532	1.301239
0.012500	0.050000	0.028432	0.420572	0.066342	0.415306	1.184534
0.100000	0.025000	0.099922	0.533599	0.490669	0.209697	1.426099
0.050000	0.025000	0.053250	0.330868	0.256699	0.208757	1.417539
0.025000	0.025000	0.029174	0.246399	0.131435	0.208416	1.386470
0.012500	0.025000	0.018163	0.218668	0.066523	0.208303	1.305812
0.100000	0.012500	0.099969	0.502340	0.491248	0.104981	1.421955
0.050000	0.012500	0.052571	0.277439	0.257002	0.104510	1.419103
0.025000	0.012500	0.027395	0.167936	0.131590	0.104339	1.411296
0.012500	0.012500	0.014811	0.123736	0.066601	0.104283	1.381793
0.100000	0.006250	0.100104	0.494314	0.491516	0.052520	1.419529
0.050000	0.006250	0.052477	0.262403	0.257142	0.052284	1.417769
0.025000	0.006250	0.026986	0.141631	0.131661	0.052198	1.415966
0.012500	0.006250	0.013900	0.084630	0.066637	0.052170	1.407993

Table 5.8 – **Case (b)**, $\omega_1 = \omega_2 = 10$, $\varepsilon = c = 1$

h	Δt	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.100000	0.012500	11.215607	110.830745	33.640520	105.601939	1.070288
0.050000	0.012500	5.076077	105.488744	17.542406	104.019898	1.692331
0.025000	0.012500	2.561485	103.725779	8.977862	103.336515	2.873940
0.012500	0.012500	1.305783	103.153331	4.543451	103.053222	5.356618
0.100000	0.006250	11.359347	62.831589	33.646167	53.063584	0.910950
0.050000	0.006250	5.133734	55.133439	17.544632	52.267408	1.198453
0.025000	0.006250	2.582849	52.694208	8.978980	51.923573	1.675013
0.012500	0.006250	1.301855	51.979935	4.544015	51.780938	2.837058
0.100000	0.003125	11.402969	42.870651	33.647787	26.565375	0.866857
0.050000	0.003125	5.152401	31.504281	17.545247	26.166468	1.040785
0.025000	0.003125	2.591626	27.501401	8.979288	25.994220	1.204101
0.012500	0.003125	1.304904	26.318014	4.544170	25.922738	1.663414
0.100000	0.001563	11.417449	36.176701	33.648292	13.287065	0.855287
0.050000	0.001563	5.158831	21.888897	17.545428	13.087465	0.997405
0.025000	0.001563	2.594882	15.800716	8.979379	13.001284	1.054426
0.012500	0.001563	1.306407	13.738792	4.544216	12.965513	1.203508

For $\varepsilon = 1$ and $c = 10$, we first run a few tests for numerous values of Δt as illustrated in figures 5.3. We can see that the choice of Δt really affects the results. Thus, from now and further, we take $\Delta t = h/2$ (unless specified otherwise), in order to assure enough accuracy.

Effectivity index with respect to ε and c. Now, we want to study the performance of the error indicator and check the effectivity index as we vary the equation parameters ε and c. We display the corresponding error and estimator results in Table 5.9, for the case (a). We observe

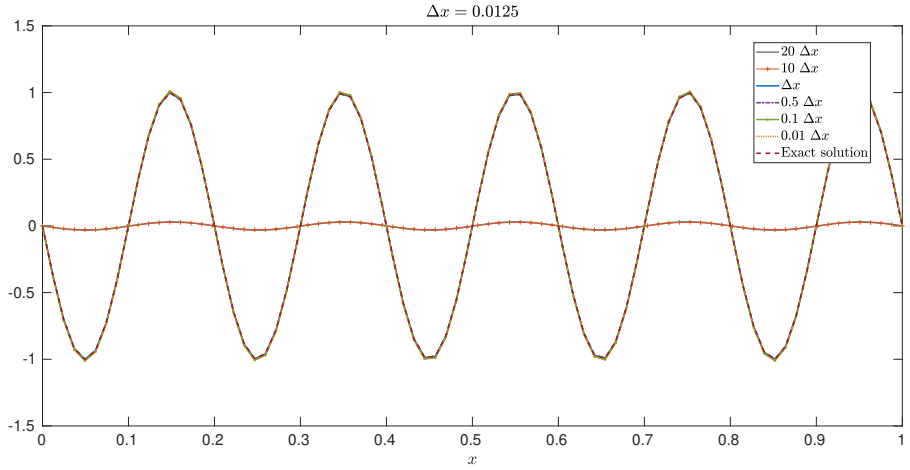


Figure 5.3 – **Case (b)** - Exact solution and approximated solutions with $\omega_1 = \omega_2 = 10$, $c = 10$, $\varepsilon = 1$ and various time steps Δt at $T = 1.55s$

that the effectivity index remains nearly constant (= 1) for several discretization parameters and different values of ε and c. In Figure 5.4, we show the exact and approximated solutions for $\varepsilon = 1$, $c = 0.1$.

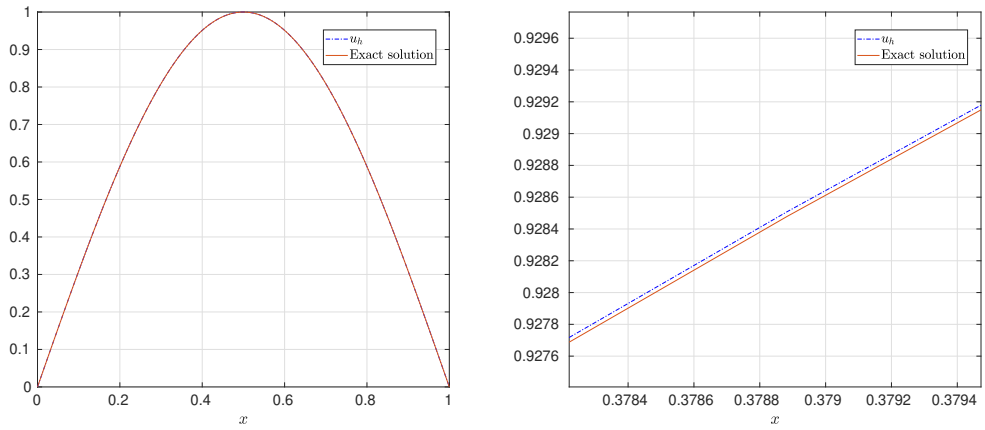


Figure 5.4 – **Case (a)** with $\omega_1 = 1$ - Exact and approximated solutions for $\varepsilon = 1$, $c = 0.1$ - Left unzoomed, right zoomed. $h = 0.025$ and $\Delta t = 0.5h$

Table 5.9 – **Case (a)** with $\omega_1 = 1$, a posteriori error estimate results for the finite elements method to solve equation (15) - $\Delta t = 0.5h$.

	h	Error	Estimator	η_S	η_T	$\tilde{e}_I$
$(\varepsilon, c) = (0.1, 0.1)$	0.050000	0.022483	0.078076	0.078076	0.000032	1.003661
	0.025000	0.011522	0.039938	0.039938	0.000004	1.001796
	0.012500	0.005833	0.020208	0.020208	0.000001	1.001337
	0.006250	0.002935	0.010166	0.010166	0.000000	1.001223
$(\varepsilon, c) = (0.1, 1.0)$	0.050000	0.022515	0.080621	0.080615	0.000988	1.034819
	0.025000	0.011527	0.040284	0.040284	0.000136	1.010050
	0.012500	0.005834	0.020254	0.020254	0.000018	1.003450
	0.006250	0.002935	0.010172	0.010172	0.000002	1.001757
$(\varepsilon, c) = (0.1, 10.0)$	0.050000	0.022571	0.209947	0.202121	0.056788	2.593575
	0.025000	0.011533	0.063782	0.063081	0.009430	1.581736
	0.012500	0.005834	0.023805	0.023764	0.001397	1.177338
	0.006250	0.002935	0.010649	0.010647	0.000194	1.048543
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.071134	0.246545	0.246545	0.000169	1.001717
	0.025000	0.036441	0.126247	0.126247	0.000023	1.001274
	0.012500	0.018446	0.063899	0.063899	0.000003	1.001201
	0.006250	0.009280	0.032147	0.032147	0.000000	1.001189
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.071139	0.246626	0.246625	0.000340	1.001972
	0.025000	0.036442	0.126258	0.126258	0.000047	1.001344
	0.012500	0.018446	0.063900	0.063900	0.000006	1.001220
	0.006250	0.009280	0.032147	0.032147	0.000001	1.001193
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.071258	0.253912	0.253819	0.006856	1.029497
	0.025000	0.036458	0.127239	0.127234	0.001091	1.008637
	0.012500	0.018448	0.064027	0.064027	0.000158	1.003092
	0.006250	0.009280	0.032164	0.032164	0.000022	1.001667
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.224991	0.779403	0.779402	0.001100	1.001201
	0.025000	0.115244	0.399196	0.399196	0.000170	1.001136
	0.012500	0.058331	0.202061	0.202061	0.000024	1.001166
	0.006250	0.029346	0.101658	0.101658	0.000003	1.001179
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.224991	0.779406	0.779405	0.001199	1.001201
	0.025000	0.115244	0.399196	0.399196	0.000185	1.001136
	0.012500	0.058331	0.202061	0.202061	0.000026	1.001166
	0.006250	0.029346	0.101658	0.101658	0.000004	1.001180
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.225008	0.779654	0.779651	0.002274	1.001443
	0.025000	0.115246	0.399230	0.399229	0.000355	1.001202
	0.012500	0.058331	0.202065	0.202065	0.000051	1.001183
	0.006250	0.029346	0.101658	0.101658	0.000007	1.001184

We move back to the case (b). The results for various values of ω_1 and ω_2 are reported in Tables 5.10, 5.11, 5.12 and 5.13. Figure 5.5 shows the exact and approximated solutions for $\omega_1 = \omega_2 = 1$ for $\varepsilon = c = 1$. We observe that:

- The effectivity index values are close to one for all considered values of ε and c
- The appropriate order of convergence is obtained

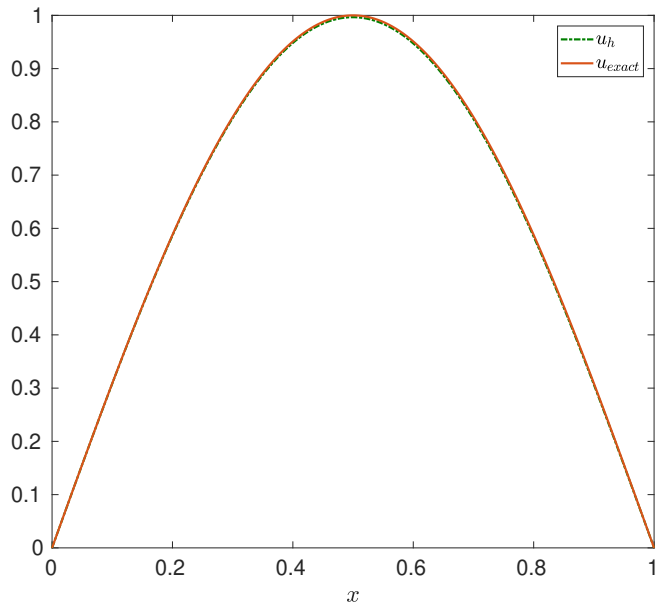


Figure 5.5 – **Case (b)**, $\omega_1 = \omega_2 = 1$ and $\varepsilon = c = 1$, exact and approximated solutions for $h = 0.00625$, $\Delta t = 0.5h$.

Table 5.10 – **Case (b)**, $\omega_1 = 1, \omega_2 = 10$, a posteriori error estimate results for the finite elements method to solve equation (15) - $\Delta t = 0.5h$.

	h	Error	Estimator	η_S	η_T	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.419310	6.830061	1.009364	6.755066	1.279648
	0.025000	0.221511	3.497553	0.551415	3.453812	1.264167
	0.012500	0.113846	1.759673	0.286818	1.736141	1.250517
	0.006250	0.057711	0.881824	0.146092	0.869639	1.242769
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.415250	6.828586	1.008446	6.753711	1.291631
	0.025000	0.220841	3.496179	0.551078	3.452475	1.267423
	0.012500	0.113970	1.759204	0.286718	1.735682	1.248786
	0.006250	0.057906	0.881679	0.146065	0.869495	1.238373
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.372832	6.887295	0.972220	6.818329	1.433334
	0.025000	0.232035	3.433148	0.533258	3.391480	1.179265
	0.012500	0.132715	1.730641	0.280870	1.707697	1.053563
	0.006250	0.071445	0.872481	0.144404	0.860448	0.992927
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.541869	7.739608	0.647744	7.712455	1.009811
	0.025000	0.252717	3.978298	0.338209	3.963896	1.114918
	0.012500	0.121215	2.005826	0.172583	1.998387	1.173593
	0.006250	0.059256	1.006494	0.087138	1.002715	1.205519
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.541310	7.739866	0.647755	7.712713	1.010886
	0.025000	0.252542	3.978345	0.338211	3.963943	1.115704
	0.012500	0.121164	2.005835	0.172583	1.998397	1.174089
	0.006250	0.059241	1.006496	0.087138	1.002717	1.205824
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.532899	7.754675	0.648890	7.727479	1.028786
	0.025000	0.249350	3.981376	0.338494	3.966961	1.130856
	0.012500	0.119965	2.006520	0.172654	1.999078	1.186245
	0.006250	0.058759	1.006651	0.087156	1.002871	1.215917

Table 5.11 – **Case (b)**, $\omega_1 = 10$, $\omega_2 = 1$, a posteriori error estimate results for the finite elements method to solve equation (15) - $\Delta t = 0.5h$.

	h	Error	Estimator	η_S	η_T	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	5.384156	27.046707	18.254876	19.957051	1.010585
	0.025000	2.710019	13.629419	9.348191	9.918285	1.026386
	0.012500	1.364674	6.844912	4.731580	4.946208	1.030798
	0.006250	0.685380	3.430843	2.380456	2.470649	1.032179
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	5.384178	27.058801	18.263140	19.965882	1.011037
	0.025000	2.710151	13.631055	9.349323	9.919467	1.026459
	0.012500	1.364697	6.845124	4.731727	4.946360	1.030812
	0.006250	0.685383	3.430870	2.380475	2.470669	1.032183
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	5.502962	28.237847	19.068831	20.826802	1.032795
	0.025000	2.727056	13.789214	9.458704	10.033710	1.032020
	0.012500	1.366957	6.865588	4.745948	4.961075	1.032200
	0.006250	0.685675	3.433473	2.382289	2.472537	1.032529
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	17.024014	85.481341	57.537531	63.217815	1.007700
	0.025000	8.569696	43.077876	29.471541	31.418651	1.023553
	0.012500	4.315515	21.634281	14.917838	15.668447	1.027976
	0.006250	2.167381	10.843548	7.505266	7.826463	1.029363
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	17.023589	85.481724	57.537793	63.218095	1.007730
	0.025000	8.569682	43.077928	29.471577	31.418689	1.023556
	0.012500	4.315515	21.634287	14.917842	15.668451	1.027976
	0.006250	2.167381	10.843549	7.505266	7.826464	1.029363

Table 5.12 – **Case (b)**, $\omega_1 = \omega_2 = 1$, a posteriori error estimate results for the finite elements method to solve equation (15) - $\Delta t = 0.5h$.

	h	Error	Estimator	η_S	η_T	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.053295	0.330752	0.256623	0.208667	1.415927
	0.025000	0.027410	0.167916	0.131575	0.104325	1.410348
	0.012500	0.013905	0.084626	0.066634	0.052168	1.407377
	0.006250	0.007004	0.042485	0.033533	0.026087	1.405822
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.053250	0.330868	0.256699	0.208757	1.417539
	0.025000	0.027395	0.167936	0.131590	0.104339	1.411296
	0.012500	0.013900	0.084630	0.066637	0.052170	1.407993
	0.006250	0.007002	0.042486	0.033534	0.026088	1.406273
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.052740	0.339451	0.261806	0.216066	1.460480
	0.025000	0.027102	0.169283	0.132446	0.105426	1.436004
	0.012500	0.013753	0.084854	0.066791	0.052336	1.426397
	0.006250	0.006929	0.042527	0.033564	0.026115	1.422276
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.165928	0.951041	0.594769	0.742110	1.078036
	0.025000	0.085042	0.479856	0.304660	0.370735	1.075416
	0.012500	0.043056	0.241085	0.154214	0.185311	1.074207
	0.006250	0.021664	0.120844	0.077586	0.092647	1.073613
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.165928	0.951044	0.594771	0.742113	1.078039
	0.025000	0.085042	0.479857	0.304661	0.370735	1.075418
	0.012500	0.043056	0.241086	0.154214	0.185311	1.074207
	0.006250	0.021664	0.120844	0.077586	0.092647	1.073614
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.165936	0.951322	0.594960	0.742318	1.078326
	0.025000	0.085042	0.479896	0.304688	0.370764	1.075505
	0.012500	0.043056	0.241091	0.154218	0.185315	1.074238
	0.006250	0.021664	0.120844	0.077587	0.092648	1.073628

Table 5.13 – **Case (b)**, $\omega_1 = \omega_2 = 10$, a posteriori error estimate results for the finite elements method to solve equation (15) - $\Delta t = 0.5h$.

	h	Error	Estimator	η_S	η_T	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	3.440788	57.333992	11.878495	56.089999	1.288362
	0.025000	1.744012	28.611470	6.079916	27.958019	1.287499
	0.012500	0.880196	14.288253	3.077021	13.952997	1.284152
	0.006250	0.442440	7.140734	1.548011	6.970921	1.281858
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	3.440996	57.359250	11.883777	56.114699	1.288855
	0.025000	1.744111	28.614906	6.080648	27.961376	1.287581
	0.012500	0.880213	14.288706	3.077118	13.953440	1.284169
	0.006250	0.442442	7.140794	1.548024	6.970980	1.281862
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	3.518194	59.813386	12.397185	58.514537	1.314815
	0.025000	1.755023	28.945746	6.151160	28.284615	1.294396
	0.012500	0.881691	14.332198	3.086463	13.995916	1.285913
	0.006250	0.442641	7.146512	1.549258	6.976563	1.282309
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	10.910631	181.788142	37.188856	177.943580	1.278838
	0.025000	5.524378	90.712271	19.046886	88.690091	1.279573
	0.012500	2.785981	45.299187	9.640890	44.261378	1.277222
	0.006250	1.399754	22.638451	4.850370	22.112743	1.275498
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	10.910377	181.788947	37.189025	177.944367	1.278873
	0.025000	5.524370	90.712379	19.046909	88.690197	1.279576
	0.012500	2.785981	45.299201	9.640893	44.261392	1.277222
	0.006250	1.399754	22.638453	4.850370	22.112744	1.275498
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	10.910423	181.869428	37.205944	178.023051	1.279443
	0.025000	5.524633	90.723212	19.049229	88.700778	1.279670
	0.012500	2.786027	45.300604	9.641196	44.262761	1.277241
	0.006250	1.399760	22.638632	4.850409	22.112919	1.275502

5.1.3 A space-time adaptive algorithm

After using various solutions to test the method and the estimators on non-adapted meshes and constant time-step configurations, we implement an adaptive algorithm in time and space. The goal of this algorithm is to control the exact error at time T. Given a pre-scripted tolerance, TOL, we want to build meshes and time steps satisfying the following goal:

$$0.75TOL \leq \left(\frac{\eta_S^2 + \eta_T^2}{T} \right)^{\frac{1}{2}} \leq 1.25TOL. \tag{5.10}$$

In order to verify this, it is enough to ensure that, for $n = 0, \ldots, N-1$,

$$\frac{0.75^2 TOL^2 \Delta t^n}{2} \leq \int_{t_n}^{t_n} \sum_{i=0}^{M} \eta_{S,i}^2 \leq \frac{1.25^2 TOL^2 \Delta t^{n+1}}{2}, \tag{5.11}$$

$$\frac{0.75^2 TOL^2 \Delta t^n}{2} \leq \int_{t_n}^{t_n} \sum_{i=0}^{M} \eta_{T,i}^2 \leq \frac{1.25^2 TOL^2 \Delta t^n}{2}. \tag{5.12}$$

for all $n = 1, \ldots, N$.

Uniform refinement:
For each time step Δt^n, we need to verify if the conditions (5.11) and (5.12) hold. If the conditions (5.11) do not hold, we have to refine or coarsen the mesh. If the conditions (5.12) do not hold, we have to make the time step either bigger or smaller. For the space refinement or coarsening, the simplest is to do so uniformly. This means, for example, if we want to refine the mesh, we can multiply the number of nodes M by two. We refer to this as the uniform refinement in space. We give more details in the Algorithm 5 below.

Algorithm 5 Uniform refinement in space in order to satisfy (5.10)

Initialization of the parameters: M, Δt^0, T, $t^0 = 0$, $n = 0$.
repeat
  $t^{n+1} = t^n + \Delta t^n$ {Increment the time }
  Compute u_h^{n+1} solution of (5.4)
  Compute η_S and η_T according to (5.8) and (5.9)
  if η_S^2 and η_T^2 verify (5.11) and (5.12) respectively **then**
    Accept the time step: $n = n+1$, $u_h^n = u_h^{n+1}$, $t^n = t^{n+1}$
  else
    if (5.11) is not satisfied **then**
      if $\int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{S,i}^2 > \frac{1.25^2 TOL^2 \Delta t^n}{2}$ **then**
        $M \leftarrow 2M$ {refine the mesh}
      else if $\int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{S,i}^2 < \frac{0.75^2 TOL^2 \Delta t^n}{2}$ **then**
        $M \leftarrow M/2$ {coarsen the mesh}
      end if
      Interpolate u_h^n on the new mesh.
    end if
    if (5.12) is not satisfied **then**
      if $\int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{T,i}^2 > \frac{1.25^2 TOL^2 \Delta t^n}{2}$ **then**
        $\Delta t^n \leftarrow \frac{\Delta t^n}{2}$ {time step too big}
      else
        $\Delta t^n \leftarrow 2\Delta t^n$ {time step too small}
      end if
    end if
  end if
until $t^n < T$

We run Algorithm 5 for the case (b), with $\omega_1 = \omega_2 = 1$ and present the results for various values of TOL in Table 5.14. M corresponds to the number of grid points at final time T.

Table 5.14 – **Case (b)**, $\omega_1 = 1$, $\omega_2 = 1$, Algorithm 5 for different TOL values.

TOL	Error	N	M
1	0.0615	11	10
0.5	0.03	18	20
0.25	0.012	36	40
0.125	0.054	72	80
0.0625	0.028	143	160
0.03125	0.017	287	320
0.015625	0.0083	540	640

We observe that:

- The error at the final time is approximatively divided by 2 when TOL is divided by 2.
- The total number of points M at final time doubles as the tolerance is divided by 2.
- The total number of time steps N is multiplied by two as the tolerance divides by 2.

These observations confirm the first-order convergence of the error and estimators.

Non-uniform refinement:
As we already mentioned, Algorithm 5 only uses uniform refinement or coarsening in space. We can though, exploit the local nature of the space estimator η_S. The goal is to satisfy (5.11). A first possibility is to require that

$$\frac{0.75^2 TOL^2 \Delta t^n}{2} \frac{1}{M} \leq \int_{t_n}^{t_n} \eta_{S,i}^2 \leq \frac{1.25^2 TOL^2 \Delta t^n}{2} \frac{1}{M} \qquad \forall i = 0, \ldots, M-1. \tag{5.13}$$

The criterion (5.13) imposes the equidistribution of the error, enforcing a comparable value of the local error estimator on each subinterval regardless of its length. Another sufficient condition for (5.11) is to impose that

$$\frac{0.75^2 TOL^2 \Delta t^n}{2} (x_{i+1} - x_i) \leq \int_{t_n}^{t_{n+1}} \eta_{S,i}^2 \leq \frac{1.25^2 TOL^2 \Delta t^n}{2} (x_{i+1} - x_i) \qquad \forall i = 0, \ldots, M-1. \tag{5.14}$$

In the condition (5.14), the re-partition of the error is weighted by $(x_{i+1} - x_i)$. We rely on the condition (5.14) for the adaptive procedure that finds a (non-uniform) partition of D, given in Algorithm 6. The idea is, for each time-step, we need to check that for each subinterval $[x_i, x_{i+1}]$, $\quad i = 0, \ldots, M-1$, of the current partition that the criterion (5.13) or (5.14) is verified. If it is too large, then we should refine the interval $[x_i, x_{i+1}]$, for instance, by adding its mid-point, while a coarsening is done if it is very small.

Algorithm 6 Non-uniform refinement in space in order to satisfy (5.10)

Initialization of the parameters: M, Δt^0, T, $t^0 = 0$, $n = 0$.
repeat
  $t^{n+1} = t^n + \Delta t^n$ {Increment the time }
  Compute u_h^{n+1} solution of (5.4)
  Compute η_S and η_T according to (5.8) and (5.9)
  if η_S^2 and η_T^2 verify (5.11) and (5.12) respectively **then**
    Accept the time step: $n = n+1$, $u_h^n = u_h^{n+1}$, $t^n = t^{n+1}$
  else
    if (5.11) is not satisfied **then**
      for $i = 0, \ldots, M-1$ **do**
        if $\eta_{S,i}^2 > (x_{i+1}^n - x_i^n)\dfrac{1.25^2 TOL^2 \Delta t^n}{2}$ **then**
          add the mid-point $\dfrac{x_i + x_{i+1}}{2}$ to the mesh
        else if $\eta_{S,i}^2 < (x_{i+1}^n - x_i^n)\dfrac{0.75^2 TOL^2 \Delta t^n}{2}$ **then**
          remove the endpoint x_{i+1} from the mesh
        end if
      end for
      Project u_h^n on the new mesh.
    end if
    if (5.12) is not satisfied **then**
      if $\displaystyle\sum_{i=0}^{M} \eta_{T,i}^2 > \frac{1.25^2 TOL^2 \Delta t^n}{2}$ **then**
        $\Delta t^{n+1} \leftarrow \frac{\Delta t^n}{2}$
      else
        $\Delta t^{n+1} \leftarrow 2\Delta t^n$
      end if
    end if
  end if
until $t^n < T$

Remark: We have also implemented Algorithm 7, similar to Algorithm 6 but with goal to satisfy

$$0.75TOL \leq \left(\frac{\eta_S^2 + \eta_T^2}{\int_0^T \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}} \right)^{\frac{1}{2}} \leq 1.25TOL.$$

Instead of (5.10). Then (5.11) and (5.12) can be replaced by:

$$\frac{0.75^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2} \leq \int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{S,i}^2 \leq \frac{1.25^2 TOL^2 \int_{t_n}^{t_{n+1}} \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2}, \quad (5.15)$$

$$\frac{0.75^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2} \leq \int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{T,i}^2 \leq \frac{1.25^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2}, \quad (5.16)$$

If we look for a non-uniform partition for which (5.15) holds, for each time step we would have to verify the following criterion,

$$\begin{aligned} \frac{0.75^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2} \frac{x_{i+1} - x_i}{|D|} &\leq \int_{t_n}^{t_{n+1}} \eta_{S,i}^2 \\ &\leq \frac{1.25^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2} \frac{x_{i+1} - x_i}{|D|} \quad \forall i = 0, \ldots, M-1. \end{aligned} \quad (5.17)$$

In Algorithm 7, we give an adaptive algorithm where we rely on the condition (5.17) to adapt the space partition.

Algorithm 7 Non-uniform refinement in space (goal 2)

Initialization of the parameters: M, Δt^0, T, $t^0 = 0$, $n = 0$.
repeat
  $t^{n+1} = t^n + \Delta t^{n+1}$ {Increment the time }
  Compute the solution u_h^{n+1} at time t
  Compute η_S and η_T according to (5.8) and (5.9)
  if η_S^2 and η_T^2 verify (5.15) and (5.16) respectively **then**
    Accept the time step: $n = n+1$, $u_h^n = u_h^{n+1}$, $t^n = t^{n+1}$
  else
    if (5.15) is not satisfied **then**
      for $i = 0, \ldots, M-1$ **do**
        if $\int_{t_n}^{t_{n+1}} \eta_{S,i}^2 > (x_{i+1}^n - x_i^n) \dfrac{1.25^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2}$ **then**
          add the mid-point $\dfrac{x_i + x_{i+1}}{2}$ to the mesh
        else if $\int_{t_n}^{t_{n+1}} \eta_{S,i}^2 < (x_{i+1}^n - x_i^n) \dfrac{0.75^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2}$ **then**
          remove the endpoint x_{i+1} from the mesh
        end if
      end for
      Project u_h^n on the new mesh.
    end if
    if (5.16) is not satisfied **then**
      if $\int_{t_n}^{t_{n+1}} \sum_{i=0}^{M} \eta_{T,i}^2 > \dfrac{1.25^2 TOL^2 \int_{t_n}^{t_{n+1}} \left\| \frac{\partial u_{h\Delta t}}{\partial x} \right\|^2_{L^2([0,1])}}{2}$ **then**
        $\Delta t^n \leftarrow \frac{\Delta t^n}{2}$
      else
        $\Delta t^n \leftarrow 2\Delta t^n$
      end if
    end if
  end if
until $t^n < T$

We now run Algorithm 6. In Table 5.15 and Table 5.16 we show the result for the case (a) with $\omega_1 = 1$ then $\omega_1 = 10$ respectively. The first observation is that the time-steps number required to reach T did not change for both cases. However, the number of grid points did change: it is larger when $\omega_1 = 10$, as we would have expected. The second observation is that, with the non-uniform space adaptation, we were able to get an error that is three times smaller than the same error obtained with 160 grid points (if we compare Table 5.1 and Table 5.15), with only 119 points. We show the results for the case (b) with $\omega_1 = \omega_2 = 1$ and $\omega_1 = \omega_2 = 2$ in Tables 5.18 and 5.19 respectively. In Figure 5.6, for $\omega_1 = \omega_2 = 2$, we show the exact and the adapted solution for $TOL = 0.2$. We can see that the space adaptation results in adding more points around the oscillations for a better accuracy. Results of the case (b) with $\omega_1 = 1$, $\omega_2 = 2$ then $\omega_1 = 1$, $\omega_2 = 10$ are illustrated in Tables 5.20 and 5.21 respectively. Finally, for for the case (b) with $\omega_1 = \omega_2 = 10$, we show the results in Table 5.22. From Tables 5.16 and 5.17, we can notice that the more space-dependent the solution is, the more grid points are required for the same TOL. The same applies to the time-dependency, as we can see in Table 5.21, (where ω_2 is the largest), the number of time-steps becomes very large as we decrease TOL.

When it comes to effectivity indices, the normalized effectivity is close to one. It is particularly the case when the space-error takes over, as we can see in Tables 5.15, 5.16, and 5.17. We can make the same observations that we made for the uniform adaptation:

- The number of points M roughly doubles when TOL is divided by two.
- When time adaptivity is important (i.e., when the solution has time dependency), the number of time steps N doubles when TOL doubles.
- The adaptive schemes allow having smaller effectivity indices since our grid parameters go to zero.

Table 5.15 – **Case (a)** with $\omega_1 = 1$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
1	14	119	0.003038	0.010713	0.010713	0.000006	1.007639
0.5	14	234	0.001656	0.006075	0.006075	0.000006	1.048172
0.25	14	472	0.000784	0.002737	0.002737	0.000004	0.997879
0.125	14	943	0.000396	0.001371	0.001371	0.000004	0.990471

Table 5.16 – **Case (a)** with $\omega_1 = 10$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
1	14	555	0.066327	0.227791	0.227791	0.000004	0.981248
0.5	14	1108	0.033034	0.113389	0.113389	0.000001	0.980694
0.25	14	2207	0.016318	0.056389	0.056389	0.000000	0.987307
0.125	14	4399	0.008280	0.028575	0.028575	0.000000	0.986061
0.0625	14	8795	0.004128	0.014244	0.014244	0.000000	0.985813

Table 5.17 – **Case (b)** with $\omega_1 = 10$ and $\omega_2 = 1$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
1	794	622	0.005689	0.026200	0.019538	0.017456	0.992070
0.5	1535	1252	0.001983	0.009249	0.006897	0.006163	1.004776
0.25	3134	2363	0.000723	0.003312	0.002494	0.002179	0.996316
0.125	6248	4610	0.000253	0.001169	0.000880	0.000770	1.002616
0.0625	12533	9293	0.000089	0.000413	0.000311	0.000272	1.004473

Table 5.18 – **Case (b)** with $\omega_1 = \omega_2 = 1$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.25	5	10	0.117901	0.144175	0.136402	0.046701	1.222848
0.125	6	16	0.078218	0.105162	0.093813	0.047520	1.344476
0.0625	9	31	0.044930	0.063788	0.055290	0.031811	1.419704
0.03125	19	57	0.022083	0.031047	0.027791	0.013842	1.405947
0.015625	28	133	0.011888	0.015286	0.012228	0.009173	1.285906
0.0078125	63	249	0.005830	0.007546	0.006349	0.004077	1.294197

Table 5.19 – **Case (b)** with $\omega_1 = \omega_2 = 2$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.25	19	33	0.126420	0.186576	0.140626	0.122617	1.475840
0.125	35	69	0.069164	0.098253	0.070035	0.068911	1.420582
0.0625	74	137	0.032870	0.605002	0.037989	0.032741	1.525750
0.03125	140	271	0.019898	0.323095	0.019141	0.017537	1.304668
0.015625	278	539	0.009565	0.160304	0.009504	0.008701	1.347073
0.0078125	562	1077	0.004450	0.078819	0.004720	0.004276	1.431043

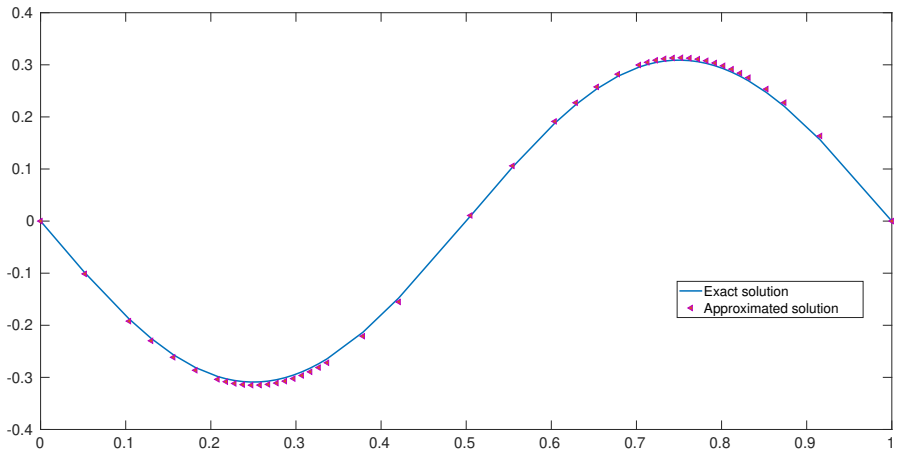


Figure 5.6 – **Case (b)** with $\omega_1 = \omega_2 = 2$. Exact solution and adapted numerical solution for $TOL = 0.2$

Table 5.20 – **Case (b)** with $\omega_1 = 1$,$\omega_2 = 2$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.25	6	10	0.201029	0.203262	0.158372	0.127413	1.011111
0.125	7	17	0.168490	0.129244	0.086022	0.096458	0.767070
0.0625	19	54	0.056265	0.051611	0.034996	0.037934	0.917289
0.03125	33	71	0.040733	0.029803	0.021536	0.020602	0.731685
0.015625	63	191	0.017179	0.015575	0.010847	0.011177	0.906657
0.0078125	141	371	0.007881	0.007606	0.005768	0.004957	0.965091

Table 5.21 – **Case (b)** with $\omega_1 = 1$, $\omega_2 = 10$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
0.25	60	50	0.192042	0.218386	0.167188	0.140501	1.137179
0.125	134	71	0.098575	0.106893	0.085930	0.063578	1.084378
0.0625	203	139	0.070611	0.769249	0.047125	0.041684	0.891019
0.03125	457	424	0.025810	0.342845	0.017555	0.018721	0.994333
0.015625	1020	860	0.012118	0.155646	0.007953	0.008499	0.960553
0.0078125	1558	1690	0.007020	0.098785	0.004504	0.005414	1.003100

Table 5.22 – **Case (b)** with $\omega_1 = 10$, $\omega_2 = 10$, Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
1	374	221	0.688965	0.893016	0.685728	0.572063	1.296171
0.5	742	438	0.310613	0.455141	0.342145	0.300149	1.465297
0.25	1483	867	0.156150	0.229754	0.171309	0.153101	1.471368
0.125	2965	1583	0.079664	0.114987	0.086595	0.075652	1.443402
0.0625	5908	3436	0.039815	0.057917	0.043272	0.038496	1.454650

In order to further test the adaptive algorithm, we consider the following brutal solution in time and space:

$$u(x,t) = \frac{100}{\pi}\left(\arctan\left(\frac{x-x_0}{\epsilon_x}\right) - \arctan\left(\frac{x-x_1}{\epsilon_x}\right)\right) \left(\arctan\left(\frac{t-t_0}{\epsilon_t}\right) - \arctan\left(\frac{t-t_1}{\epsilon_t}\right)\right) \quad (5.18)$$

We run the algorithm with $T = 0.55$ [s], $x_0 = \frac{L}{3}$, $x_1 = \frac{2L}{3}$, $t_0 = \frac{T}{3}$, $t_1 = \frac{2T}{3}$, $\epsilon_x = 0.01$ and $\epsilon_t = 0.15$. We illustrate the results in Table 5.23. In Figure 5.7, we show the exact and the adapted solutions for some TOL values.

Table 5.23 – Exact solution (5.18), Algorithm 6 for different TOL values.

TOL	N	M	Error	Estimator	η_S	η_T	$\tilde{e}_I$
10	32	96	8.297362	7.387390	5.108976	5.335906	0.890330
5	63	175	3.418831	3.280637	2.301518	2.337862	0.959579
2.5	122	334	1.555447	1.948801	1.442779	1.310044	1.252888
1.25	245	652	0.842753	0.944091	0.723408	0.606621	1.120247
0.625	339	1287	0.502159	0.501230	0.377402	0.329848	0.998151
0.3125	461	2556	0.293027	0.273138	0.193324	0.192951	0.932127

We see that the values of error are divided by two when the tolerance is. Since this is a brutal solution in space and in time, we observe that we need numerous points and time steps to reach an acceptable range of error values. Without adaptation, it seems hard to obtain the same results with a fixed number of points. Moreover, the effectivity index is roughly equal to 1 for all chosen TOL. Finally, we consider the following initial condition:

$$u(x,0) = 0.2\left(\arctan\left(\frac{x-0.15}{0.001}\right) - \arctan\left(\frac{x-0.45}{0.001}\right)\right), \quad (5.19)$$

that we display in Figure 5.8. In this test case situation, our goal is not to compare with an exact solution (Thus, there is no source term in the resolution scheme). Instead, we want to see the behavior of our solution and the way it exhibits the characteristics of the diffusion and convection terms, whilse using the adaptive algorithm. We use various ε and c and we

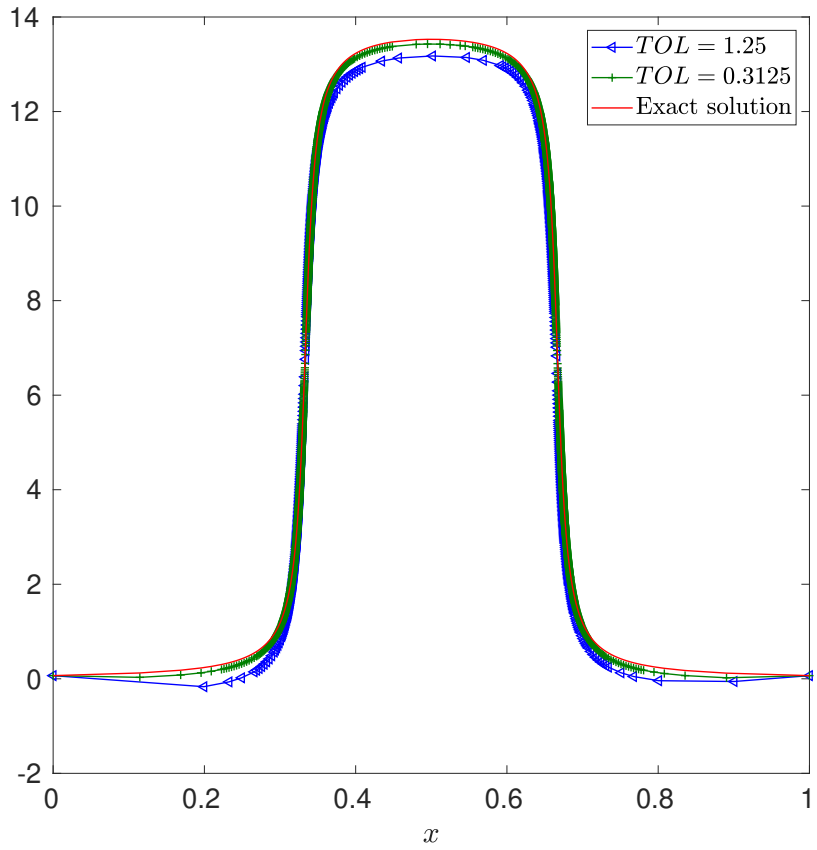


Figure 5.7 – Exact solution for the initial condition (5.18) and adapted numerical solutions for $TOL = 1.25$ and $TOL = 0.3125$

consider the final time $T = 0.15$. First, we rely on Algorithm 6 and show the results in Table 5.24.

Then, we rely on Algorithm 7 and show the results in Table 5.25, we illustrate the corresponding results. In Figures 5.13 and 5.10 we show the results for $(\varepsilon, c) = (0.1, 1)$ and $(\varepsilon, c) = (0.1, 2)$ respectively. The solution is transported to the right (more in Figure 5.10) but also diffused at the same time. For $(\varepsilon, c) = (1, 1)$, we hardly see the effect of the transport because of the diffusion as illustrate Figures 5.11 and 5.12. Furthermore, we can observe the numerical convergence of the solution as the values of TOL get smaller. From these results, we can still see a convergence with roughly the expected order.

Table 5.24 – Results for the initial condition (5.19) with Algorithm 6 and various TOL values

	TOL	N	M	η_S	η_T
$(\varepsilon, c) = (0.1, 1)$	7.812500e-03	113	42	0.095412	0.015891
	3.906250e-03	230	112	0.051061	0.008234
	1.953125e-03	454	233	0.021524	0.004710
	9.765625e-04	901	478	0.012483	0.002381
$(\varepsilon, c) = (1, 0.1)$	7.812500e-03	24	18	0.015733	0.003517
	3.906250e-03	32	84	0.008975	0.001761
	1.953125e-03	98	59	0.004571	0.000981
	9.765625e-04	191	124	0.002147	0.000541
$(\varepsilon, c) = (1, 1)$	7.812500e-03	41	45	0.024351	0.005487
	3.906250e-03	84	92	0.015012	0.002691
	1.953125e-03	166	182	0.0087023	0.000987
	9.765625e-04	340	341	0.0045123	0.000427
$(\varepsilon, c) = (0.1, 2)$	1.562500e-02	78	57	0.125733	0.021971
	7.812500e-03	152	103	0.068912	0.012393
	3.906250e-03	312	211	0.007123	0.001667
	1.953125e-03	601	405	0.017031	0.003941
	9.765625e-04	1155	689	0.008756	0.001581

Table 5.25 – Results for the initial condition (5.19) with Algorithm 7 and various TOL values

	TOL	N	M	η_S	η_T
$(\varepsilon, c) = (0.1, 1)$	7.812500e-03	259	127	0.028638	0.003547
	3.906250e-03	473	226	0.014764	0.001667
	1.953125e-03	805	380	0.008150	0.000933
	9.765625e-04	1371	676	0.004787	0.000586
$(\varepsilon, c) = (1, 0.1)$	7.812500e-03	55	27	0.007523	0.001893
	3.906250e-03	99	84	0.004427	0.000852
	1.953125e-03	178	145	0.002915	0.000388
	9.765625e-04	311	250	0.001633	0.000224
$(\varepsilon, c) = (1, 1)$	7.812500e-03	70	61	0.009704	0.002350
	3.906250e-03	127	115	0.005826	0.001026
	1.953125e-03	229	153	0.003488	0.000469
	9.765625e-04	393	324	0.002132	0.000268
$(\varepsilon, c) = (0.1, 2)$	1.562500e-02	134	84	0.070989	0.009498
	7.812500e-03	259	127	0.028638	0.003547
	3.906250e-03	473	226	0.014764	0.001667
	1.953125e-03	805	380	0.008150	0.000933
	9.765625e-04	1371	676	0.004787	0.000586

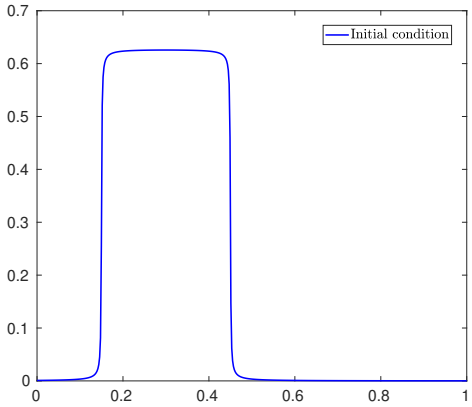


Figure 5.8 – The initial condition given by (5.19)

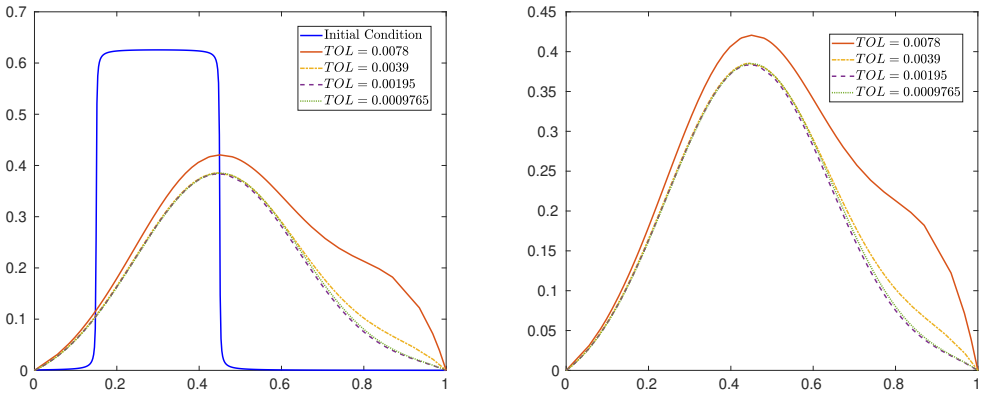


Figure 5.9 – $(\varepsilon, c) = (0.1, 1)$. Left: Initial condition and solution at $T = 0.15$. Right: Zoom on the solution at $T = 0.15$.

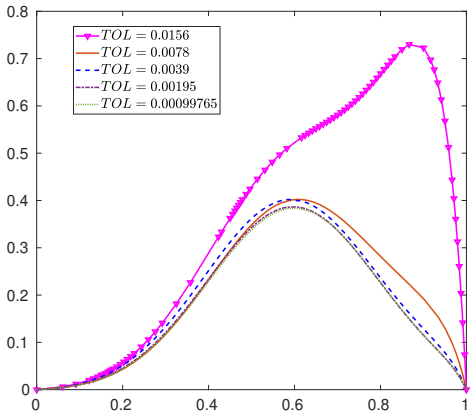


Figure 5.10 – $(\varepsilon, c) = (0.1, 2)$. Solution at $T = 0.15$

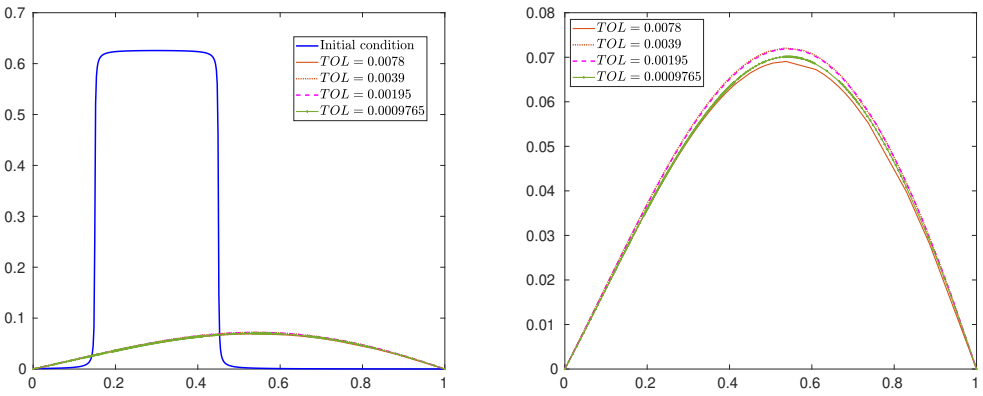


Figure 5.11 – $(\varepsilon, c) = (1, 1)$. Left: Initial condition and solution at $T = 0.15$. Right: Zoom on the solution at $T = 0.15$.

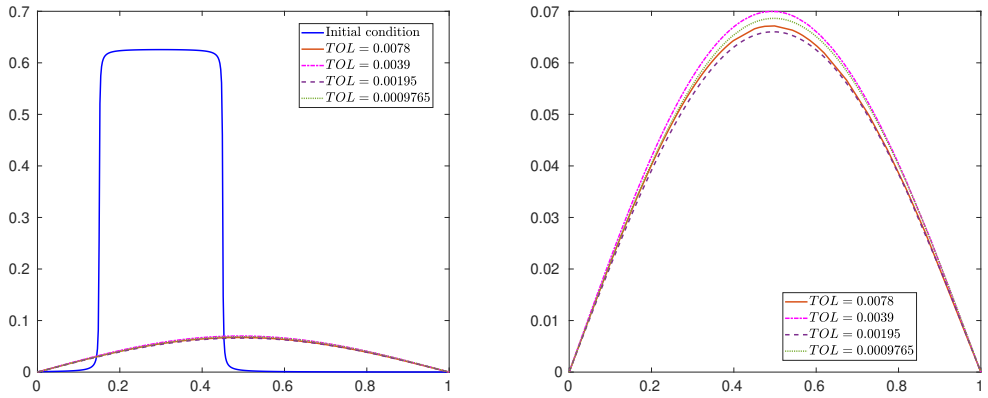


Figure 5.12 – $(\varepsilon, c) = (1, 0.1)$. Left: Initial condition and solution at $T = 0.15$. Right: Zoom on the solution at $T = 0.15$.

5.2 The time-splitting scheme:

We now want to check the results when we deal with the same equation but with a splitting scheme. This means that we solve the equation in two steps, with the finite element method. The scheme splits the diffusion and convection effects. The problem reads: knowing $u_h^n \in V_h$, find $u_h^{n+1/2} \in V_h$ such that,

$$\int_0^1 \frac{u_h^{n+1/2} - u_h^n}{\Delta t} v_h + c \int_0^1 \frac{du_h^{n+1/2}}{dx} v_h = 0 \quad \forall v_h \in V_h, \tag{5.20}$$

then find $u_h^{n+1} \in V_h$ such that

$$\int_0^1 \frac{u_h^{n+1} - u_h^{n+1/2}}{\Delta t} v_h + \varepsilon \int_0^1 \frac{du_h^{n+1}}{dx} \frac{dv_h}{dx} dx = \int_0^1 f(t_{n+1}) v_h \quad \forall v_h \in V_h. \tag{5.21}$$

We write both equations (5.20) and (5.21) in the matrix: set $v_h = \varphi_i$, $i = 1, \ldots, N$, where φ_i are the affine shape functions defined in section 3. We get:

$$\frac{1}{\Delta t} \sum_{j=0}^{M} u_j^{n+1/2} \int_0^1 \varphi_j \varphi_i - \frac{1}{\Delta t} \sum_{j=0}^{M} u_j^n \int_0^1 \varphi_j \varphi_i + c \sum_{j=0}^{M} u_j^{n+1/2} \int_0^1 \varphi_j' \varphi_i = 0, \tag{5.22}$$

$$\frac{1}{\Delta t} \sum_{j=0}^{M} u_j^{n+1} \int_0^1 \varphi_j \varphi_i - \frac{1}{\Delta t} \sum_{j=0}^{M} u_j^{n+1/2} \int_0^1 \varphi_j \varphi_i + \varepsilon \sum_{j=0}^{M} u_j^{n+1} \int_0^1 \varphi_j' \varphi_i' = \int_0^1 f(t_{n+1}) \varphi_i. \tag{5.23}$$

Finally we get the following differential systems that we need to solve:

$$\frac{1}{\Delta t} M \left(\vec{u}^{n+1/2} - \vec{u}^n \right) + B \vec{u}^{n+1/2} = \vec{0},$$

$$\frac{1}{\Delta t}M\left(\vec{u}^{n+1}-\vec{u}^{n+1/2}\right)+A\vec{u}^{n+1}=\vec{f}^{n+1}.$$

where $\vec{u}^{n+1}=(u_1,\ldots,u_M)^T$, $M_{ij}=\int_0^1\varphi_j\varphi_i$, $A_{ij}=\varepsilon\int_0^1\varphi'_j\varphi'_i$, $B_{ij}=c\int_0^1\varphi'_j\varphi_i$ and $f_i^{n+1}=\int_0^1 f(t_{n+1})\varphi_i$.

5.2.1 A posteriori error estimate

We remind that the approximation $u_{h\Delta t}$ of u on each subinterval $[t_n,t_{n+1}]$, $n=0,...,M$, is defined by:

$$u_{h\Delta t}(\mathbf{x},t):=\frac{t-t_n}{\Delta t}u_h^{n+1}(\mathbf{x})+\frac{t_{n+1}-t}{\Delta t}u_h^n(\mathbf{x}).$$

In order to compute an a posteriori error estimate, we start from ($e=u-u_{h\Delta t}$):

$$(\frac{\partial e}{\partial t},e)+\varepsilon\int_0^1\left(\frac{\partial e}{\partial x}\right)^2+c\underbrace{\int_0^1\left(\frac{\partial e}{\partial x}\right)e}_{=0}=\int_0^1\left(f-\frac{\partial u_{h\Delta t}}{\partial t}-c\frac{\partial u_{h\Delta t}}{\partial x}\right)e-\varepsilon\int_0^1\left(\frac{\partial u_{h\Delta t}}{\partial x}\right)\left(\frac{\partial e}{\partial x}\right),\quad(5.24)$$

Using the equations (5.20) and (5.21) we can write:

$$\int_0^1\frac{\partial u_{h\Delta t}}{\partial t}v_h=\int_0^1\frac{u^{n+1}-u^n}{\Delta t}v_h=\int_0^1 f(t_{n+1})v_h-\varepsilon\int_0^1\frac{du_h^{n+1}}{dx}\frac{dv_h}{dx}-c\int_0^1\frac{du_h^{n+1/2}}{dx}v_h$$

Thus,

$$\begin{aligned}\int_0^1\frac{\partial u_{h\Delta t}}{\partial t}v_h\quad+\quad&\varepsilon\int_0^1\left(\frac{\partial u_{h\Delta t}}{\partial x}\right)\left(\frac{\partial v_h}{\partial x}\right)+c\int_0^1\left(\frac{\partial u_{h\Delta t}}{\partial x}\right)v_h=\int_0^1 fv_h+\int_0^1(f(t_{n+1})-f)v_h\\&+\varepsilon\int_0^1\left(\frac{\partial(u_{h\Delta t}-u_h^{n+1})}{\partial x}\right)\frac{\partial v_h}{\partial x}+c\int_0^1\left(\frac{\partial(u_{h\Delta t}-u_h^{n+1/2})}{\partial x}\right)v_h\quad\forall v_h\in V_h.\end{aligned}$$

Using this in equation (5.24), we get:

$$\begin{aligned}(\frac{\partial e}{\partial t},e)+\varepsilon\int_0^1\left(\frac{\partial e}{\partial x}\right)^2&=\int_0^1\left(f-\frac{\partial u_{h\Delta t}}{\partial t}-c\frac{\partial u_{h\Delta t}}{\partial x}\right)(e-v_h)\\&-\varepsilon\int_0^1\frac{\partial u_{h\Delta t}}{\partial x}\frac{\partial(e-v_h)}{\partial x}+\int_0^1\left(f-f(t_{n+1})\right)v_h\\&-\varepsilon\int_0^1\frac{\partial(u_{h\Delta t}-u_h^{n+1})}{\partial x}\frac{\partial v_h}{\partial x}-c\int_0^1\frac{\partial(u_{h\Delta t}-u_h^{n+1/2})}{\partial x}v_h,\end{aligned}$$

which leads to:

$$\begin{aligned}(\frac{\partial e}{\partial t},e)+\varepsilon\int_0^1\left(\frac{\partial e}{\partial x}\right)^2&=\sum_{i=0}^M\int_{x_i}^{x_{i+1}}\left(f-\frac{\partial u_{h\Delta t}}{\partial t}-c\frac{\partial u_{h\Delta t}}{\partial x}+\varepsilon\frac{\partial^2 u_{h\Delta t}}{\partial x^2}\right)(e-v_h)\\&-\left[\varepsilon\frac{\partial u_{h\Delta t}}{\partial x}(e-v_h)\right]_{x_i}^{x_{i+1}}+\int_{x_i}^{x_{i+1}}\left(f-f(t_{n+1})\right)v_h\\&-\varepsilon\int_{x_i}^{x_{i+1}}\frac{\partial(u_{h\Delta t}-u_h^{n+1})}{\partial x}\frac{\partial v_h}{\partial x}-c\int_{x_i}^{x_{i+1}}\frac{\partial(u_{h\Delta t}-u_h^{n+1/2})}{\partial x}v_h,\end{aligned}$$

for any $v_h \in V_h$. Since $e \in H_0^1(0,1)$, $v \in C^0([0,1])$ and we can choose $v_h = R_h e$, the Lagrange interpolant. Doing so, the jump terms vanish. We site the interpolation results:

- $\|e - R_h e\| \leq h_i C_a \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})} \leq \bar{C} h_i \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}$,
- $\left\| \frac{\partial R_h e}{\partial x} \right\|_{L^2(x_i,x_{i+1})} \leq C_b \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})} \leq \bar{C} \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}$,
- $\|R_h e\|_{L^2(x_i,x_{i+1})} \leq C_c \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})} \leq \bar{C} \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}$,

where C_a, C_b and C_c are positive constants and $\bar{C} = \max(C_a, C_b, C_c)$.

Denote

$$R_s = \sum_{i=0}^{M} \int_{x_i}^{x_{i+1}} \left(f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2} \right)(e - v_h) + \int_{x_i}^{x_{i+1}} \left(f - f(t_{n+1}) \right) v_h$$
$$-\varepsilon \int_{x_i}^{x_{i+1}} \frac{\partial (u_{h\Delta t} - u_h^{n+1})}{\partial x} \frac{\partial v_h}{\partial x} - c \int_{x_i}^{x_{i+1}} \frac{\partial (u_{h\Delta t} - u_h^{n+1/2})}{\partial x} v_h.$$

Applying Cauchy-Schwarz inequality on the integrals we get:

$$R_s \leq \sum_{i=0}^{M} \left\| f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2} \right\|_{L^2(x_i,x_{i+1})} \|e - R_h e\|_{L^2(x_i,x_{i+1})}$$
$$+ \sum_{i=0}^{M} \left\| (f - f(t_{n+1})) - c\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1/2}) \right\|_{L^2(x_i,x_{i+1})} \|R_h e\|_{L^2(x_i,x_{i+1})}$$
$$+ \sum_{i=0}^{M} \varepsilon \left\| \frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1}) \right\|_{L^2(x_i,x_{i+1})} \left\| \frac{\partial R_h e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}$$

Using the inequalitites above, and applying Cauchy-Schwarz again we obtain:

$$R_s \leq \tilde{C} \left(\sum_{i=0}^{M} \left(h_i \left\| f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2} \right\|_{L^2(x_i,x_{i+1})} \right)^2 \right)^{\frac{1}{2}} \left\| \frac{\partial e}{\partial x} \right\|_{L^2([0,1])}$$
$$+ \tilde{C} \left(\sum_{i=0}^{M} \left(\left\| (f - f(t_{n+1})) - c\frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1/2}) \right\|_{L^2(x_i,x_{i+1})} + \varepsilon \left\| \frac{\partial}{\partial x}(u_{h\Delta t} - u_h^{n+1}) \right\|_{L^2(x_i,x_{i+1})} \right)^2 \right)^{\frac{1}{2}} \left\| \frac{\partial e}{\partial x} \right\|_{L^2([0,1])}$$

Using Young's inequality we get:

$$R_s \leq \tilde{C} \sum_{i=0}^{M} \eta_{S,i}^2 + \frac{\varepsilon}{4} \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}^2 + \tilde{C} \sum_{i=0}^{M} \eta_{T,i}^2 + \frac{\varepsilon}{4} \left\| \frac{\partial e}{\partial x} \right\|_{L^2(x_i,x_{i+1})}^2$$

where

$$\eta_{S,i}^2 = \frac{1}{\varepsilon} \left(h_i \left\| f - \frac{\partial u_{h\Delta t}}{\partial t} - c\frac{\partial u_{h\Delta t}}{\partial x} + \varepsilon \frac{\partial^2 u_{h\Delta t}}{\partial x^2} \right\|_{L^2(x_i,x_{i+1})} \right)^2$$

and

$$\eta_{T,i}^2 = \frac{1}{\varepsilon}\left(\left\|(f-f(t_{n+1}))-c\frac{\partial}{\partial x}(u_{h\Delta t}-u_h^{n+1/2})\right\|_{L^2(x_i,x_{i+1})}+\varepsilon\left\|\frac{\partial}{\partial x}(u_{h\Delta t}-u_h^{n+1})\right\|_{L^2(x_i,x_{i+1})}\right)^2$$

Finally we get:

$$\frac{1}{2}\frac{d}{dt}\|e\|^2_{L^2([0,1])}+\varepsilon\left\|\frac{\partial e}{\partial x}\right\|^2_{L^2([0,1])}\leq\tilde{C}\sum_{i=0}^{M}\eta_{S,i}^2+\tilde{C}\sum_{i=0}^{M}\eta_{T,i}^2+\frac{\varepsilon}{2}\left\|\frac{\partial e}{\partial x}\right\|^2_{L^2([0,1])}$$

we integrate between t_n and t_{n+1} to obtain:

$$\|e(t_{n+1})\|^2_{L^2([0,1])}+\varepsilon\int_{t_n}^{t_{n+1}}\left\|\frac{\partial e}{\partial x}\right\|^2_{L^2([0,1])}\leq\|e(t_n)\|^2_{L^2([0,1])}+\tilde{C}\sum_{i=0}^{M}\int_{t_n}^{t_{n+1}}\left(\eta_{S,i}^2+\eta_{T,i}^2\right).$$

We define the error as:

$$\text{Error}=\left(\|e(T)\|^2_{L^2([0,1])}-\|e(0)\|^2_{L^2([0,1])}+\varepsilon\int_0^T\left\|\frac{\partial e}{\partial x}\right\|^2_{L^2([0,1])}\right)^{1/2}$$

and the estimator as:

$$\text{Estimator}=\left(\sum_{n=0}^{N}\sum_{i=0}^{M}\int_{t_n}^{t_{n+1}}\left(\eta_{S,i}^2+\eta_{T,i}^2\right)\right)^{1/2}.$$

For the effectivity index and the other quantities, we use the same notations introduced earlier. We observe that the space-related estimator, η_S, remains the same obtained earlier without the splitting.

5.2.2 Numerical results

We use the following cases to test the scheme:

(a) $u(x,t)=sin(\pi x)$,

(b) $u(x,t)=sin(\omega_1\pi x)sin(\omega_2\pi t)$, where $\omega_1,\omega_2>0$.

Effectivity index with respect to h and Δt. We start with testing the estimator results for a fixed set of equation parameters ($\varepsilon=c=1$), as we vary Δt and h. First, we consider the case (a) and check the sharpness of the space indicator. To reduce the time error (and the splitting error) we consider time steps such that $\Delta t<h$. We report the results in Table 5.26. The first line of each block in Table 5.26 corresponds to cases where the space error is bigger than the time error, particularly in the last block. That is why the effectivity index converges to the value we obtained earlier without splitting for the space indicator. To confirm this, we consider the same case (a) and we take time-steps that are very small compared to h. The error in time is negligible compared to that in space and the values of η_T in Table 5.27 confirm this. We observe that the effectivity index converges to 3.5. We illustrate the exact and approximated

solutions for this case in Figure 5.13.

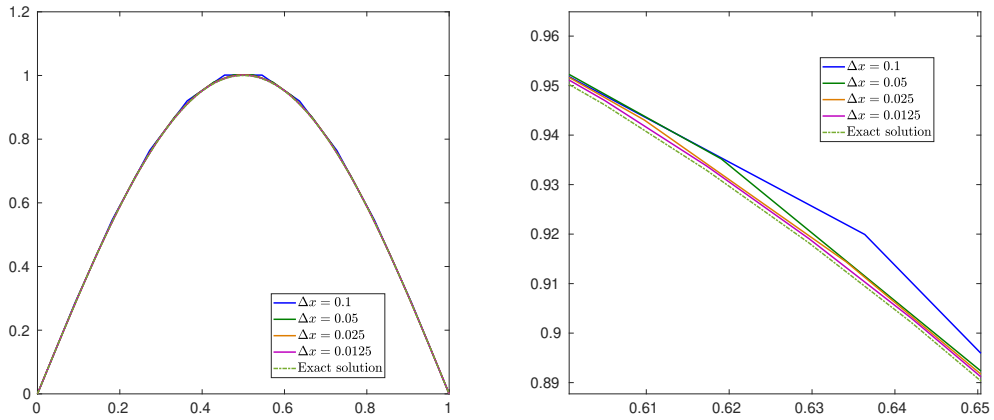


Figure 5.13 – **Case (a)**, exact and approximated solutions for the splitting scheme, unzoomed (left), zoomed (right).

Table 5.26 – **Case (a)**, $\varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	e_I
0.050000	0.006250	0.116146	0.269223	2.317972
0.025000	0.006250	0.098876	0.163374	1.652314
0.012500	0.006250	0.094617	0.121452	1.283620
0.006250	0.006250	0.095533	0.110637	1.158102
0.050000	0.003125	0.085738	0.253072	2.951707
0.025000	0.003125	0.059402	0.136721	2.301610
0.012500	0.003125	0.050462	0.081965	1.624300
0.006250	0.003125	0.048109	0.060483	1.257201
0.050000	0.001563	0.075542	0.248568	3.290463
0.025000	0.001563	0.043620	0.129108	2.959822
0.012500	0.001563	0.030055	0.068921	2.293149
0.006250	0.001563	0.025497	0.041056	1.610226
0.050000	0.000781	0.072507	0.247271	3.410313
0.025000	0.000781	0.038490	0.127057	3.301026
0.012500	0.000781	0.022005	0.065231	2.964420
0.006250	0.000781	0.015119	0.034605	2.288837

Table 5.27 – **Case (a)** with $\omega_1 = 1$ - Space effectivity index

h	Δt	Error	Estimator	η_S	η_T	e_I
0.100000	0.001000	0.062262	0.217708	0.217578	0.007512	3.496620
0.050000	0.000500	0.032610	0.113254	0.113192	0.003728	3.473023
0.025000	0.000250	0.016687	0.057859	0.057829	0.001860	3.467292
0.012500	0.000125	0.008441	0.029257	0.029242	0.000930	3.465873

For the sharpness of η_T we use the case (b) with $\omega_1 = 1$ and $\omega_2 = 10$. The results, illustrated in Table 5.28 show that the spatial error is negligible compared to the temporal one. We can see in each block (corresponding to a fixed Δt) that the error does not follow h. The effectivity index converges to 15, which now is the sharpness of the time error indicator.

Table 5.28 – **Case (b)** $\omega_1 = 1, \omega_2 = 10$, $\varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	η_S	η_T	e_I
0.025000	0.025000	0.450673	6.917756	0.516581	6.898442	15.349850
0.012500	0.025000	0.455195	6.971403	0.261404	6.966501	15.315203
0.006250	0.025000	0.457631	7.226897	0.131504	7.225701	15.791977
0.003125	0.025000	0.458893	8.159885	0.065956	8.159618	17.781685
0.025000	0.012500	0.226783	3.568565	0.551436	3.525703	15.735628
0.012500	0.012500	0.228265	3.556741	0.279035	3.545778	15.581627
0.006250	0.012500	0.229270	3.623423	0.140373	3.620703	15.804178
0.003125	0.012500	0.229839	3.898653	0.070404	3.898017	16.962512
0.025000	0.006250	0.117721	1.861765	0.566818	1.773383	15.815017
0.012500	0.006250	0.116824	1.802818	0.286816	1.779857	15.431978
0.006250	0.006250	0.116864	1.806843	0.144287	1.801073	15.461092
0.003125	0.006250	0.117011	1.880487	0.072367	1.879094	16.071059
0.025000	0.003125	0.063835	1.057578	0.573906	0.888315	16.567358
0.012500	0.003125	0.060370	0.936993	0.290402	0.890855	15.520748
0.006250	0.003125	0.059543	0.909143	0.146090	0.897329	15.268723
0.003125	0.003125	0.059378	0.921587	0.073271	0.918669	15.520558

In order to attain an effectivity index that is close to 1, we divide the space indicator η_S by 3.5 and the time indicator by 15 in order to obtain the following normalized error indicator:

$$\sqrt{\left(\frac{\eta_S}{3.5}\right)^2 + \left(\frac{\eta_T}{15}\right)^2}.$$

The corresponding effectivity is denoted by $\tilde{e_I}$. We examine the case (b) for $\varepsilon = c = 1$, and different values of ω_1 and ω_2. The results are reported in Tables 5.29, 5.30, 5.31 and 5.32. The expected order of convergence (1) is observed. If we look at the last line of each block of these Tables, we can see that when Δt and h are simultaneously divided by two, the error and the estimator are both divided by two. Furthermore, the effectivity index is always close to one as we vary h and Δt.

Table 5.29 – **Case (b)**,$\omega_1 = 1, \omega_2 = 10$, $\varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	$\tilde{e}_I$
0.025000	0.025000	0.450673	0.514868	1.142444
0.012500	0.025000	0.455195	0.503310	1.105702
0.006250	0.025000	0.457631	0.517519	1.130865
0.003125	0.025000	0.458893	0.583142	1.270758
0.025000	0.012500	0.226783	0.298029	1.314164
0.012500	0.012500	0.228265	0.265800	1.164434
0.006250	0.012500	0.229270	0.261784	1.141818
0.003125	0.012500	0.229839	0.279172	1.214641
0.025000	0.006250	0.117721	0.207081	1.759074
0.012500	0.006250	0.116824	0.151770	1.299142
0.006250	0.006250	0.116864	0.135238	1.157227
0.003125	0.006250	0.117011	0.135841	1.160926
0.025000	0.003125	0.063835	0.177591	2.782027
0.012500	0.003125	0.060370	0.105326	1.744662
0.006250	0.003125	0.059543	0.076752	1.289026
0.003125	0.003125	0.059378	0.068952	1.161224

Table 5.30 – **Case (b)**,$\omega_1 = 1, \omega_2 = 1$, $\varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	$\tilde{e}_I$
0.025000	0.012500	0.036442	0.044173	1.212130
0.050000	0.012500	0.057824	0.075912	1.312816
0.025000	0.012500	0.036442	0.044173	1.212130
0.012500	0.012500	0.027982	0.041473	1.482158
0.025000	0.006250	0.029891	0.039444	1.319611
0.050000	0.006250	0.054100	0.074690	1.380576
0.025000	0.006250	0.029891	0.039444	1.319611
0.012500	0.006250	0.018690	0.024618	1.317132
0.025000	0.003125	0.027812	0.038392	1.380424
0.050000	0.003125	0.053039	0.074416	1.403059
0.025000	0.003125	0.027812	0.038392	1.380424
0.012500	0.003125	0.015207	0.020510	1.348651
0.025000	0.001563	0.027201	0.038145	1.402368
0.050000	0.001563	0.052742	0.074360	1.409872
0.025000	0.001563	0.027201	0.038145	1.402368
0.012500	0.001563	0.014108	0.019567	1.386975

Table 5.31 – **Case (b)** $\omega_1 = 10, \omega_2 = 1, \varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	$\tilde{e}_I$
0.025000	0.012500	2.731533	2.791806	1.022066
0.050000	0.012500	5.412469	5.327120	0.984231
0.025000	0.012500	2.731533	2.791806	1.022066
0.012500	0.012500	1.394919	1.553781	1.113887
0.025000	0.006250	2.718716	2.724547	1.002145
0.050000	0.006250	5.414626	5.290911	0.977152
0.025000	0.006250	2.718716	2.724547	1.002145
0.012500	0.006250	1.373163	1.414090	1.029804
0.025000	0.003125	2.717940	2.707752	0.996252
0.050000	0.003125	5.420791	5.281842	0.974367
0.025000	0.003125	2.717940	2.707752	0.996252
0.012500	0.003125	1.367891	1.379031	1.008144
0.025000	0.001563	2.719238	2.703569	0.994238
0.050000	0.001563	5.425205	5.279578	0.973157
0.025000	0.001563	2.719238	2.703569	0.994238
0.012500	0.001563	1.367084	1.370411	1.002433

Table 5.32 – **Case (b)** $\omega_1 = 10, \omega_2 = 10$, $\varepsilon = c = 1$, Splitting scheme

h	Δt	Error	Estimator	$\tilde{e}_I$
0.050000	0.006250	5.166146	6.303154	1.220088
0.025000	0.006250	2.599412	4.537944	1.745758
0.012500	0.006250	1.321370	3.940030	2.981776
0.006250	0.006250	0.685034	3.774458	5.509887
0.050000	0.003125	5.185946	5.405633	1.042362
0.025000	0.003125	2.601284	3.193807	1.227781
0.012500	0.003125	1.311313	2.274739	1.734704
0.006250	0.003125	0.663751	1.971095	2.969632
0.050000	0.001563	5.195233	5.156628	0.992569
0.025000	0.001563	2.603965	2.756975	1.058760
0.012500	0.001563	1.309757	1.608315	1.227949
0.006250	0.001563	0.658886	1.139179	1.728947
0.050000	0.000781	5.199674	5.092499	0.979388
0.025000	0.000781	2.605693	2.636561	1.011846
0.012500	0.000781	1.309757	1.392928	1.063501
0.006250	0.000781	0.657725	0.807286	1.227392

Effectivity index with respect to ε and c. In this part, we focus on altering the equation parameters ε and c for both cases (a) and (b). In Table 5.33, we show the results for case (a) with $\omega_1 = 1$. We observe that the time error is very small (as indicates the time estimator η_T, and that the effectivity index is always close to one, as obtained previously with the non-splitting scheme. From Tables 5.34 5.35 and 5.36 and 5.37 we can see that, the effectivity index is not very sensitive to the equation parameters, but rather the error is. The effectivity indices remain close to one for all cases.

Table 5.33 – **Case (a)**, with $\omega_1 = 1$, splitting scheme, $\Delta t = \frac{h}{10c}$.

	h	Error	Estimator	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.071156	0.071255	1.001390
	0.025000	0.036445	0.036487	1.001156
	0.012500	0.018447	0.018468	1.001154
	0.006250	0.009280	0.009291	1.001169
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.071813	0.071730	0.998842
	0.025000	0.036828	0.036953	1.003399
	0.012500	0.018654	0.018923	1.014425
	0.006250	0.009388	0.009735	1.036946
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.094184	0.112511	1.194586
	0.025000	0.049698	0.069160	1.391610
	0.012500	0.025703	0.045025	1.751747
	0.006250	0.013100	0.030399	2.320518
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.225001	0.225261	1.001154
	0.025000	0.115245	0.115375	1.001123
	0.012500	0.058331	0.058399	1.001161
	0.006250	0.029346	0.029381	1.001178
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.224964	0.225276	1.001387
	0.025000	0.115242	0.115389	1.001282
	0.012500	0.058334	0.058414	1.001361
	0.006250	0.029349	0.029395	1.001573
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.227342	0.226972	0.998374
	0.025000	0.116615	0.116902	1.002459
	0.012500	0.059076	0.059852	1.013140
	0.006250	0.029735	0.030788	1.035436

Table 5.34 – **Case (b)**, $\omega_1 = 1$, $\omega_2 = 1$, splitting scheme, $\Delta t = \frac{h}{2c}$.

	h	Error	Estimator	$\tilde{e_I}$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.052615	0.074385	1.413750
	0.025000	0.026993	0.038100	1.411489
	0.012500	0.013676	0.019286	1.410240
	0.006250	0.006884	0.009703	1.409558
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.054100	0.074635	1.379569
	0.025000	0.027812	0.038348	1.378835
	0.012500	0.014108	0.019528	1.384213
	0.006250	0.007106	0.009940	1.398747
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.070037	0.098478	1.406077
	0.025000	0.036986	0.057750	1.561418
	0.012500	0.019134	0.035924	1.877497
	0.006250	0.009753	0.023461	2.405546
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.166039	0.172364	1.038097
	0.025000	0.085057	0.088272	1.037793
	0.012500	0.043056	0.044678	1.037681
	0.006250	0.021662	0.022477	1.037618
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.166163	0.172368	1.037345
	0.025000	0.085130	0.088279	1.036992
	0.012500	0.043096	0.044687	1.036928
	0.006250	0.021683	0.022487	1.037059
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.167923	0.173149	1.031123
	0.025000	0.086142	0.089148	1.034900
	0.012500	0.043640	0.045610	1.045119
	0.006250	0.021966	0.023430	1.066642

Table 5.35 – **Case (b)**, $\omega_1 = 10$, $\omega_2 = 1$, splitting scheme, $\Delta t = \frac{h}{2c}$.

	h	Error	Estimator	$\tilde{e_I}$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	5.393149	5.286860	0.980292
	0.025000	2.711785	2.706880	0.998191
	0.012500	1.365085	1.369989	1.003592
	0.006250	0.685490	0.689221	1.005443
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	5.414626	5.289364	0.976866
	0.025000	2.717940	2.707037	0.995989
	0.012500	1.367084	1.370044	1.002165
	0.006250	0.686233	0.689387	1.004595
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	5.803671	5.759122	0.992324
	0.025000	2.847113	2.879112	1.011239
	0.012500	1.418879	1.458883	1.028194
	0.006250	0.709507	0.747203	1.053129
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	17.034264	16.664018	0.978265
	0.025000	8.571156	8.534033	0.995669
	0.012500	4.315760	4.319434	1.000851
	0.006250	2.167430	2.173072	1.002603
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	17.040387	16.663333	0.977873
	0.025000	8.572829	8.533544	0.995417
	0.012500	4.316259	4.319174	1.000675
	0.006250	2.167598	2.172944	1.002467
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	17.109086	16.646602	0.972969
	0.025000	8.590939	8.523489	0.992149
	0.012500	4.321422	4.314363	0.998367
	0.006250	2.169247	2.171028	1.000821

Table 5.36 – **Case (b)**, $\omega_1 = 10$, $\omega_2 = 10$, splitting scheme, $\Delta t = \frac{h}{2c}$.

	h	Error	Estimator	$\tilde{e_I}$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	5.140180	6.149838	1.196425
	0.025000	2.593027	3.120218	1.203311
	0.012500	1.306806	1.572226	1.203106
	0.006250	0.656537	0.789292	1.202204
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	5.166146	6.158164	1.192023
	0.025000	2.601284	3.123098	1.200599
	0.012500	1.309757	1.573405	1.201296
	0.006250	0.657725	0.789924	1.200995
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	5.557237	5.542130	0.997281
	0.025000	2.725870	2.771183	1.016623
	0.012500	1.358776	1.404211	1.033438
	0.006250	0.679578	0.719092	1.058145
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	16.265090	19.378156	1.191396
	0.025000	8.204539	9.833745	1.198574
	0.012500	4.133768	4.955165	1.198704
	0.006250	2.076400	2.487587	1.198028
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	16.272180	19.378825	1.190918
	0.025000	8.206553	9.833751	1.198280
	0.012500	4.134381	4.955063	1.198502
	0.006250	2.076609	2.487512	1.197872
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	16.391297	15.987872	0.975388
	0.025000	8.230753	8.185001	0.994441
	0.012500	4.140290	4.142748	1.000594
	0.006250	2.078335	2.084602	1.003015

Table 5.37 – **Case (b)**, $\omega_1 = 1$, $\omega_2 = 10$, $\varepsilon = c$, splitting scheme, $\Delta t = \frac{h}{2c}$.

	h	Error	Estimator	$\tilde{e}_I$
$(\varepsilon, c) = (1.0, 0.1)$	0.050000	0.436544	0.536640	1.229291
	0.025000	0.222103	0.280062	1.260957
	0.012500	0.114142	0.142387	1.247458
	0.006250	0.058123	0.071731	1.234130
$(\varepsilon, c) = (1.0, 1.0)$	0.050000	0.443017	0.543742	1.227363
	0.025000	0.226783	0.283985	1.252233
	0.012500	0.116824	0.144745	1.239004
	0.006250	0.059543	0.073222	1.229731
$(\varepsilon, c) = (1.0, 10.0)$	0.050000	0.130421	0.422140	3.236744
	0.025000	0.069433	0.251937	3.628482
	0.012500	0.036084	0.157264	4.358281
	0.006250	0.018446	0.102621	5.563309
$(\varepsilon, c) = (10.0, 0.1)$	0.050000	0.560567	0.547198	0.976152
	0.025000	0.253452	0.281763	1.111701
	0.012500	0.121392	0.142259	1.171897
	0.006250	0.059624	0.071435	1.198103
$(\varepsilon, c) = (10.0, 1.0)$	0.050000	0.565762	0.548112	0.968804
	0.025000	0.257511	0.282240	1.096029
	0.012500	0.123749	0.142560	1.152009
	0.006250	0.060871	0.071631	1.176770
$(\varepsilon, c) = (10.0, 10.0)$	0.050000	0.201695	0.219918	1.090349
	0.025000	0.104922	0.119877	1.142530
	0.012500	0.053544	0.067306	1.257014
	0.006250	0.027052	0.039673	1.466574

5.2.3 A space-time adaptive algorithm

We use the normalized estimator for the adaptive algorithms. Here we use Algorithm 7 described in the previous section. First, we consider the case (b), that we recall:

$$u(x,t) = sin(\omega_1 \pi x) sin(\omega_2 \pi t), \quad \omega_1, \omega_2 > 0.$$

We vary ω_1 and ω_2, and check the convergence of the result for different TOL values. The initial time step $\Delta t = 0.0005$ and the initial space step is $h = 0.01$. Results for $(\omega_1, \omega_2) = (2,1)$ and $(\omega_1, \omega_2) = (2,2)$ with the adaptive algorithm, are plotted in Figures 5.14 and 5.15 respectively for the final time $T = 0.15$ [s]. Note for all the numerical experiments presented below, we consider the normalized error indicators: $\frac{\eta_S}{3.5}$, $\frac{\eta_T}{15}$ and the corresponding effectivity index:

$$\tilde{e}_I = \frac{\sqrt{\left(\frac{\eta_S}{3.5}\right)^2 + \left(\frac{\eta_T}{15}\right)^2}}{\text{Error}}.$$

Results for $(\omega_1, \omega_2) = (2,1)$ and $(\omega_1, \omega_2) = (1,2)$ are given in Tables 5.38, and 5.39 respectively. In Table 5.38, we observe that for a $TOL = 0.0125$, the number of grid points needed is 112, which is bigger than that needed for a smaller TOL in the case where $\omega_1 = 1$, as shows Table 5.39. This is expected since, the smaller ω_1 is, the less the solution depends less on space. When it comes the the time indicator, the same does not necessarily apply. In fact, more time steps are needed when the solution depends more in space. This is because of the splitting scheme: in order to reduce the splitting effect on the spatial distribution, more temporal precision is required. On the other hand, for $(\omega_1, \omega_2) = (2,2)$, more time and space steps are needed than both previous cases for the same TOL values. For all cases, the effectivity indices remain close to one.

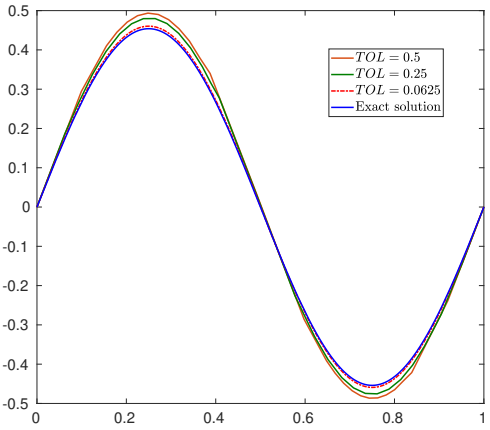


Figure 5.14 – **Case (b)**, $\omega_1 = 2$, $\omega_2 = 1$: exact and approximated solutions for different TOL values, with the splitting scheme.

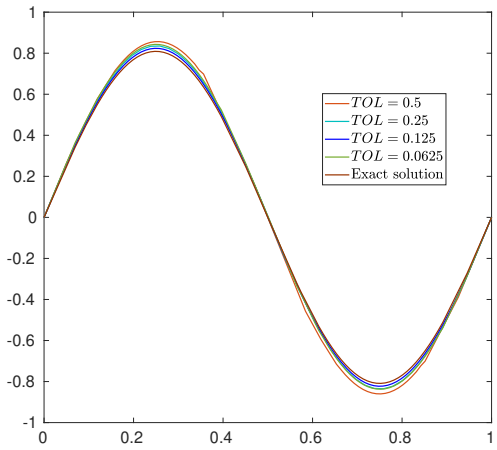


Figure 5.15 – **Case (b)**, $\omega_1 = \omega_2 = 2$: exact and approximated solutions for different TOL values, with the splitting scheme.

Table 5.38 – **Case (b)** with $\omega_1 = 2$, $\omega_2 = 1$, Algorithm 7 for different TOL values.

TOL	N	M	Error	Normalized Estimator	$\eta_S/3.5$	$\eta_T/15$	$\tilde{e}_I$
0.1	100	14	0.141423	0.484284	0.136180	0.097291	1.183419
0.05	156	26	0.053389	0.226658	0.064360	0.036734	1.388041
0.025	188	29	0.038677	0.151816	0.042493	0.032907	1.389588
0.0125	670	112	0.009742	0.041100	0.011507	0.008874	1.491592

Table 5.39 – **Case (b)** with $\omega_1 = 1$, $\omega_2 = 2$, Algorithm 7 for different TOL values.

TOL	N	M	Error	Normalized Estimator	$\eta_S/3.5$	$\eta_T/15$	$\tilde{e}_I$
0.0625	58	26	0.020236	0.130942	0.036090	0.019124	2.018358
0.03125	127	82	0.020410	0.072502	0.019589	0.012611	1.141453
0.0078	312	177	0.004284	0.022695	0.006150	0.003861	1.695187
0.0039	532	310	0.002208	0.012773	0.003473	0.002118	1.842023

Table 5.40 – **Case (b)** with $\omega_1 = \omega_2 = 2$, Algorithm 7 for different TOL values.

TOL	N	M	Error	Normalized Estimator	$\eta_S/3.5$	$\eta_T/15$	$\tilde{e}_I$
0.1	132	24	0.112393	0.501841	0.144347	0.087320	1.501011
0.05	257	49	0.092175	0.378747	0.110194	0.046241	1.296485
0.025	376	34	0.047392	0.216704	0.062899	0.029176	1.463052
0.0125	636	69	0.023532	0.108163	0.031276	0.016480	1.502282

Now, we consider the brutal solution (5.18) previously introduced in Subsection 5.1.3, that we recall here:

$$u(x,t) = \frac{100}{\pi}\left(\arctan\left(\frac{x-x_0}{\epsilon_x}\right) - \arctan\left(\frac{x-x_1}{\epsilon_x}\right)\right)\left(\arctan\left(\frac{t-t_0}{\epsilon_t}\right) - \arctan\left(\frac{t-t_1}{\epsilon_t}\right)\right)$$

We Initially start with $\Delta t = 0.0005$ and $h = 0.01$, and the final time is $T = 0.55$ [s]. In Figure 5.16, we show the convergence of the approximated solution when we apply Algorithm 7 on the splitting scheme for various TOL values. Table 5.41 illustrates the results of the algorithm. It shows that, especially for smaller TOL, the estimators and the error follow the right order. Moreover, the normalized effectivity converges to 1, as expected.

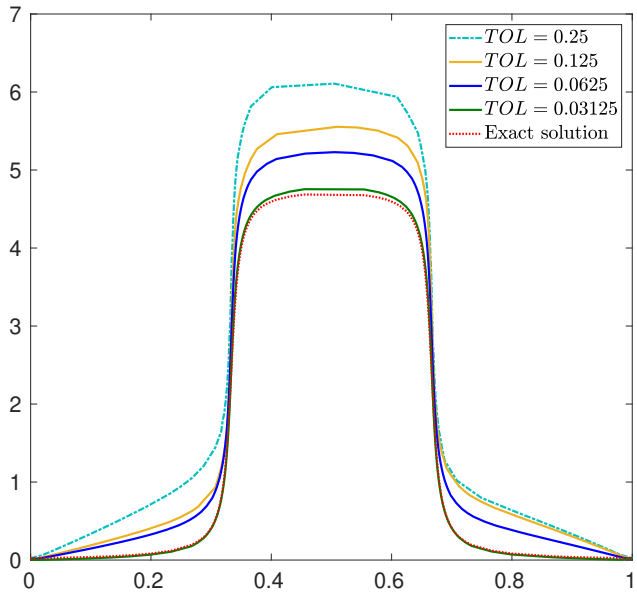


Figure 5.16 – Exact and approximated solutions for the initial condition (5.18), for different TOL values, with the splitting scheme.

Table 5.41 – Case (5.18) , Algorithm 7 for different TOL values.

TOL	N	M	Error	Normalized Estimator	$\eta_S/3.5$	$\eta_T/15$	$\tilde{e}_I$
0.25	4770	79	7.458076	7.056316	3.473665	5.693511	0.894267
0.125	4801	140	5.829379	4.839279	1.697478	4.389688	0.807369
0.0625	9600	218	3.398597	2.660731	0.793294	2.484608	0.767428
0.003125	19200	342	1.121451	1.270290	0.524751	1.103228	1.089365

Finally, the same way we did with the non-splitting scheme, we consider the initial condition (5.19) and eliminate the source term. The goal is to qualitatively check the behavior of our solution by allowing the equation to convect and diffuse our initial solution. Thus, with the

same adaptive algorithm Algorithm 7, and for $T = 0.15$ [s], we run multiple tests with different values for the equation parameters, ε and c. The first observation is that all figures show the right convergence behavior as TOL goes to zero. If we compare Figure 5.17, where $(\varepsilon, c) = (1,1)$ and Figure 5.18, where $(\varepsilon, c) = (1, 0.1)$, we observe that for the first case, the initial solution is not only diffused, but also is slightly transported to the right of the grid's horizontal axis. For the second case however, we can't easily spot the convection effect after just 0.15 [s], which is explained by the negligeable convection parametrer c of value 0.1. Similarly, we now reduce the diffusion effect by taking $\varepsilon = 0.1$. We vary c by taking it 1 then 2. In Figure 5.19, we can first observe that the initial condition is less diffused, as expected. On the other hand, for $c = 2$, we can clearly see the convection effect on Figure 5.20, in comparison to the case where $c = 1$. Comparing the results with those obtained without splitting, for the same adaptive Algorithm, we can see that the solution converges faster without splitting than when we use the splitting scheme (i.e., converges for bigger TOL values), which is expected.

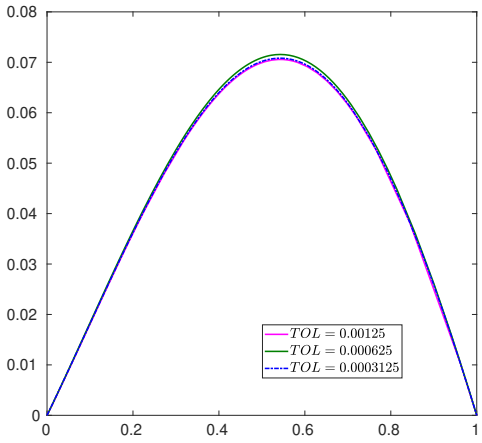


Figure 5.17 – Results for the initial condition (5.19), $(\varepsilon, c) = (1,1)$. Solution at $T = 0.15$

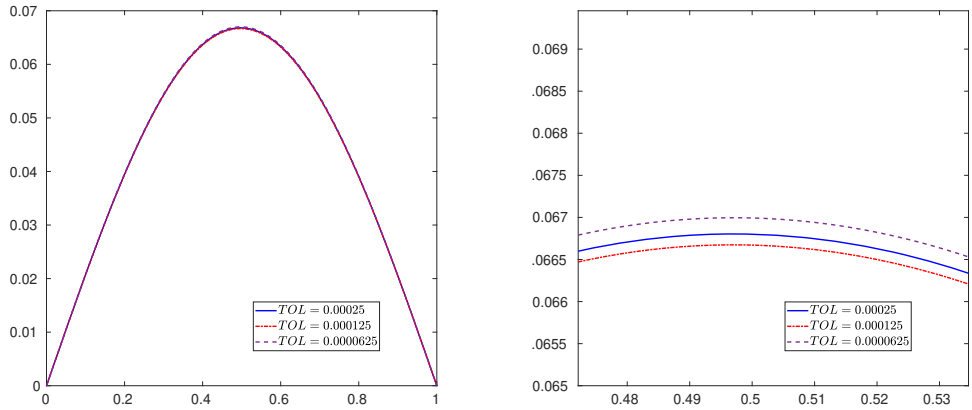


Figure 5.18 – Results for the initial condition (5.19), for $(\varepsilon, c) = (1, 0.1)$. Solution at $T = 0.15$, unzoomed (left), zoomed (right)

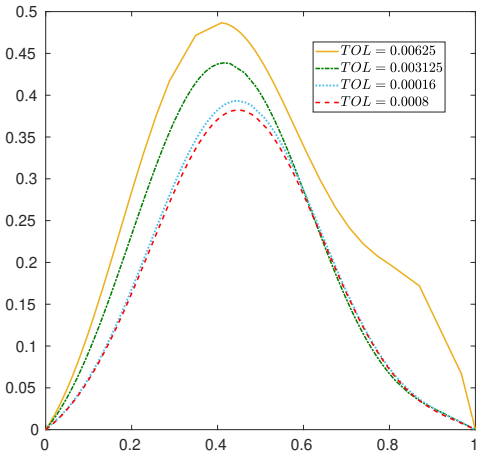


Figure 5.19 – Results for the initial condition (5.19),$(\varepsilon, c) = (0.1, 1)$. Solution at $T = 0.15$

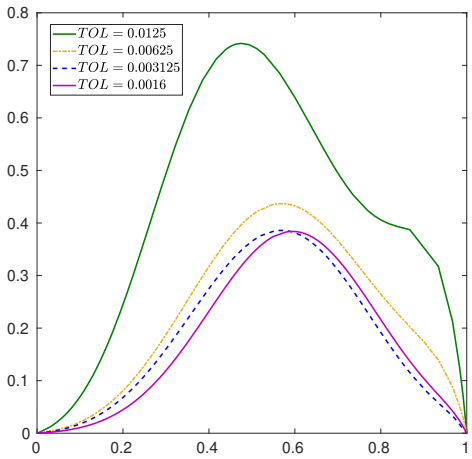


Figure 5.20 – Results for the initial condition (5.19), $(\varepsilon, c) = (0.1, 2)$. Solution at $T = 0.15$

Table 5.42 – Results for the initial condition (5.19) with Algorithm 7, various TOL values and equation parameters ε and c.

	TOL	N	M	η_S	η_T
$(\varepsilon, c) = (1, 1)$	1.562500e-02	416	19	0.068685	0.023186
	7.812500e-03	1235	55	0.115269	0.033543
	3.906250e-03	1788	81	0.046367	0.014362
	1.953125e-03	1444	110	0.006969	0.002148
	9.765625e-04	2627	201	0.003996	0.001238
	4.882812e-04	3908	198	0.001900	0.000539
$(\varepsilon, c) = (1, 0.1)$	5.000000e-03	1258	49	0.046759	0.013897
	2.500000e-03	1513	81	0.012596	0.004166
	1.250000e-03	1769	78	0.003814	0.001082
	6.250000e-04	3173	123	0.002153	0.000626
	4.625000e-04	2167	183	0.001508	0.000429
	3.422500e-04	2771	270	0.001187	0.000332
	2.312500e-04	3798	285	0.000876	0.000243
	1.572500e-04	5177	710	0.000640	0.000178
$(\varepsilon, c) = (0.1, 1)$	2.500000e-02	645	20	0.178530	0.025131
	1.250000e-02	982	29	0.060902	0.008421
	6.250000e-03	2015	73	0.037591	0.004692
	3.125000e-03	3199	106	0.020951	0.002221
	3.125000e-03	3199	106	0.020951	0.002221
	1.600000e-03	4608	170	0.010338	0.001175
	8.000000e-04	8010	303	0.005999	0.000657
$(\varepsilon, c) = (0.1, 2)$	2.500000e-02	1490	37	0.244741	0.032455
	1.250000e-02	3060	72	0.122202	0.015585
	6.250000e-03	3521	89	0.044326	0.004688
	3.125000e-03	5839	151	0.022771	0.002456
	3.125000e-03	5839	151	0.022771	0.002456
	1.600000e-03	9595	254	0.012768	0.001363

6 Conclusion

We have designed a three-dimensional numerical model for the transport, sedimentation and resuspension of mono-dispersed particles within a Newtonian flow with free surfaces.

The operator splitting scheme proposed in [117] for the simulation of free surface flows has been adapted to include sedimentation dynamics. The time-splitting strategy and the appropriate mix of finite elements, finite volumes and structured grids, has proved to be very flexible to incorporate various numerical solvers. Decompression and redistribution algorithms first used in [117] have been improved and explained. A numerical method was proposed to solve the sedimentation and resuspension problem. The method consists in splitting the equation into two parts: a deposition equation and a resuspension equation. We used a Godunov scheme [76] for the deposition operator, and a higher order finite difference scheme to solve the resuspension operator.

The method has been first tested on simplified one-dimensional problems then benchmarked with experimental results. On the first hand, we have calibrated the deposition model versus experiments; the numerical results agree with experimental measurements for pure sedimentation, erosion processes and impinging jets. On the second hand, the complete model with resuspension effects has been calibrated in situations that include shear-induced movement of the sediments such as sediment flushing and horizontal scouring.

The current model can be extended to include many sediment populations (poly-disperse model) as explained in the Appendix. Validation with experiments is still required. Future perspectives include the introduction of interactions between the sediment particles, and the simulation of real-life 3D topographies. Furthermore, a turbulence model can be introduced.

Finally, a theoretical study of one-dimensional convection-diffusion equation has been proposed. A posteriori error estimates in time and space have been developed and used for an adaptive algorithm. This could be a first step to design an error indicator for the sedimentation problem described above.

A Extension of the monodisperse sedimentation model to polydisperse model

In this appendix, we extend the proposed model for sediment dynamics into a poly-dispesed model (a model with poly-dispersed particle populations) for the miscible sediment in the liquid.

A.1 Mathematical model

Let $\Lambda \in \mathbb{R}^3$ be the bounded computational domain containing the fluid (mixture of the liquid and sediments) and the ambiant air and let $T > 0$ be the final time of the simulation. For any given time $t \in (0, T)$, let $\Omega_t \subset \Lambda$ be the domain occupied by the fluid (mixture including sediments), so that the remaining part of the domain Λ is occupied by the ambient air.
Let Q_T denote the space-time domain containing the liquid, that is $Q_T = \{(\mathbf{x}, t) : \mathbf{x} \in \Omega_t, 0 < t < T\}$. Let Γ_T be the free surface between the liquid and the air and defined by $\Gamma_T = \{(x, t) \in \partial Q_T \setminus \big(\partial\Lambda \times (0, T)\big)\}$. First let us introduce the set of **unknowns**:

- We use a volume of fluid (VOF) [91, 117] method to track the fluid volume. This is done by introducing a characteristic function of the liquid $\varphi : \Lambda \times (0, T) \to \{0, 1\}$, which implies that $Q_T = \big\{(\mathbf{x}, t) \in \Lambda \times (0, T) : \varphi(\mathbf{x}, t) = 1\big\}$.

- The velocity field $\mathbf{v} : Q_T \to \mathbb{R}^3$ and the pressure field $p : Q_T \to \mathbb{R}$ are assumed to satisfy time-dependent, incompressible Navier-Stokes equations, with variable density and viscosity coefficients, and an additional Darcy-like reaction term modeling the porous solid matrix.

- Assuming M populations of sediments (differing by size and/or density and/or shape), the presence rate of a sediment population is denoted by the solid fractions f_{s_i} for $i = 1, \ldots, M$, in the liquid domain. For the various classes of sediments, the sediment concentrations are defined in the liquid domain as $f_{s_i} : Q_T \to [0, f_{s_{CR}}]$ where $f_{s_{CR}}$ is the maximal sediment concentration.

The total amount of sediment

$$f_s = \sum_{i=1}^{M} f_{s_i}$$

is actually limited by a critical maximum value $f_{s_{CR}} < 1$ (i.e the solid fraction of the packed sediments). This critical value is essentially related to grains' size and shape.

A.1.1 The volume-of-fluid equation

In order to describe the kinematics of the free surface, $\varphi : \Lambda \times (0,T) \to \mathbb{R}$ must satisfy (in a weak sense):

$$\frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi = 0 \qquad \text{in } \Lambda \times (0,T). \tag{A.1}$$

This equation corresponds to the fact that the fluid particles move at velocity $\mathbf{v}$, more precisely, $\mathbf{v}(\mathbf{X}(t),t) = \mathbf{v}(\mathbf{X}(0),0)$, where $\mathbf{X}(t)$ is the trajectory of a fluid particle starting from $\mathbf{X}(0)$ at time $t = 0$, thus $\mathbf{X}'(t) = \mathbf{v}(\mathbf{X}(t),t)$.
The characteristic function of the liquid domain φ is given at initial time, which is equivalent to defining the initial liquid region $\Omega_0 = \{\mathbf{x} \in \Lambda : \varphi(\mathbf{x},0) = 1\}$. The initial velocity field $\mathbf{v}$ is prescribed in Ω_0 (see below), and boundary conditions are given for φ on the inlet part of $\partial\Omega_t$ (the part of the boundaries with an inflow, if any).

A.1.2 Navier-stokes equations with sediment dynamics

We assume that the liquid mixture velocity and pressure $\mathbf{v} : Q_T \to \mathbb{R}^3$ and $p : Q_T \to \mathbb{R}$ satisfy:

$$\rho_m \Big(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \Big) - 2\nabla \cdot \Big(\mu_m \mathbf{D}(\mathbf{v}) \Big) + \alpha_m \mathbf{v} + \nabla p = \rho_m \mathbf{g} \quad \text{in } Q_T, \tag{A.2}$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } Q_T. \tag{A.3}$$

Here $\mathbf{D}(\mathbf{v}) = 1/2(\nabla \mathbf{v} + \nabla \mathbf{v}^T)$ is the symmetric deformation tensor, $\mathbf{g}$ denotes the gravity field. All physical coefficients depend on the sediment concentrations f_{s_i}, $i = 1,\dots,M$. More precisely, the density ρ_m is given by a linear weighted combination of the individual densities ρ_i:

$$\rho_m = \rho_m(f_s, f_{s_1}, f_{s_2}, \dots, f_{s_M}) = \rho_l(1 - f_s) + \sum_{i=1}^{M} \rho_i f_{s_i}, \tag{A.4}$$

where ρ_l (resp. ρ_i) is the fluid density (resp. the density of sediment i). The viscosity $\mu_m = \mu_m(f_s, f_{s_1}, f_{s_2}, \dots, f_{s_M})$ is given by

$$\mu_m(f_s, f_{s_1}, f_{s_2}, \dots, f_{s_M}) = \begin{cases} \mu_l \Big(1 - \dfrac{f_s}{f_{s_{CR}}}\Big)^{-2.5 f_{s_{CR}}}, & \text{if } f_s < f_{s_{CO}}, \\ \mu_l \Big(1 - \dfrac{f_{s_{CO}}}{f_{s_{CR}}}\Big)^{-2.5 f_{s_{CR}}}, & \text{otherwise,} \end{cases} \tag{A.5}$$

where μ_l is the Newtonian dynamic viscosity of the fluid and $f_{s_{CO}}$ is a cohesion threshold parameter to be calibrated. Moreover, the velocity in the Navier-Stokes equations is penalized with a Brinkman term using Carman-Kozeny empirical law, which represents the coupling with Darcy flow in porous media. The reaction coefficient $\alpha_m = \alpha_m(f_s, f_{s_1}, f_{s_2}, \dots, f_{s_M})$ in (2.2)

is given by:

$$\alpha_m(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}) = K\frac{\mu_l f_s^2}{d_*^2(f_s)(f_{s_{CR}} - f_s)^3 + \varepsilon}, \tag{A.6}$$

where K and ε are constants to be calibrated and $d_* = d_*(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M})$ is the local mean particle diameter computed as

$$d_*(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}) = \sum_{i=1}^{M} \frac{f_{s_i}}{f_s} d_i. \tag{A.7}$$

In fact, we consider the sediment particles to be spherical and d_i to be the diameter of each population i, $i = 1, \ldots, M$.
Note that, practically, the parameter ε is small and avoids a division by zero when $f_s = f_{s_{CR}}$ in (2.6). The Navier-Stokes equations (2.2)-(2.3) are completed with initial and boundary conditions. The initial conditions for the velocity are

$$\mathbf{v}(0) = \mathbf{v}_0 \qquad \text{in} \quad \Omega_0,$$

slip or no-slip boundary conditions are imposed on the boundary of the liquid domain that is in contact with the boundary of the cavity $\partial\Lambda$. Surface tension effects on the liquid-gas interface are not taken into account, and the ambient air is supposed to have no influence on the liquid, and is treated as vacuum. The boundary conditions on the liquid-gas interface are thus given by the no-force boundary condition:

$$-p\mathbf{n}_\Gamma + 2\mu_m \mathbf{D}(\mathbf{v})\mathbf{n}_\Gamma = 0 \qquad \text{on } \Gamma_T, \tag{A.8}$$

with $\mathbf{n}_\Gamma$ the external normal vector to Γ_T.

A.1.3 The sedimentation equation

For each population $i = 1, \ldots, M$, the sediment concentration $f_{s_i} : Q_T \to \mathbb{R}$ satisfies:

$$\frac{\partial f_{s_i}}{\partial t} + \mathbf{v} \cdot \nabla f_{s_i} + \nabla \cdot \left(F_{d_i} + F_{r_i}\right) = 0, \quad i = 1, \ldots, M \tag{A.9}$$

where $F_{d_i} = F_{d_i}\left(f_{s_i}\right)$ is a deposition flux and $F_{r_i} = F_{r_i}(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M})$ is a resuspension flux.

The deposition flux

Various multiphase models for the deposition of sediments flux exist in the literature. Here, we consider a parabolic model given by:

$$F_{d_i}\left(f_{s_i}\right) = f_{s_i}\Big(1 - \frac{f_{s_i}}{f_{s_{CR}}}\Big)\Big(k v_{s_i} \frac{\mathbf{g}}{\|\mathbf{g}\|}\Big), \quad i = 1, \ldots, M \tag{A.10}$$

where k is a positive parameter independant from f_{s_i} and v_{s_i} is the maximal sediment velocity given by the Stokes' law:

$$v_{s_i} = v_{s_i}(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}) = \frac{{d_i}^2 \|\mathbf{g}\|(\rho_i - \rho_l)}{18\mu_l},$$

The resuspension flux

The resuspension of particles is taken into account through the resuspension flux given by

$$F_{r_i}(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}) = -f_{s_i}\Big(1 - \frac{f_{s_i}}{f_{s_{CR}}}\Big) A(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) \nabla f_{s_i}, \tag{A.11}$$

where based on [129], $A(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v})$ is given by:

$$A(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = K_r \Big(\frac{\tau(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) - \tau_{CR}}{\tau_{CR}} \Big)_+,$$

where K_r is the resuspension constant $[m^2/s]$, $(v)_+ = \max(v, 0)$ and $\tau(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v})$ $[N/m^2]$ is given by the tangential part of the stress tensor:

$$\tau(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = 2\mu_m(f_s) \|\mathbf{D}(\mathbf{v})\mathbf{n}(f_s) - (\mathbf{D}(\mathbf{v})\mathbf{n}(f_s) \cdot \mathbf{n}(f_s))\mathbf{n}(f_s)\|, \tag{A.12}$$

where $\mathbf{n}(f_s) = \dfrac{\nabla f_s}{\|\nabla f_s\|}$ for $\nabla f_s \neq 0$, $\tau(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = 0$ otherwise. Our model supports two configurations of the critical shear:

- **Constant critical shear**: in this case we consider a constant critical shear value $\tau_{CR} \in [0.02, 0.5]$ (see [130]).
- **Shields shear**: in this case , based on [131, 132], we define the critical shear as:

  $$\tau_{CR}(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = \theta(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) \left(\rho_m - \rho_l\right) \|\mathbf{g}\| d_*(f_s),$$

  where $\theta(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v})$ is given by:

  $$\theta(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = \begin{cases} 0.010595 \ln(Re_*) + \dfrac{0.110476}{Re_*} + 0.0027197 & \text{for} \quad Re_* \leq 500, \\ 0.068 & \text{for} \quad Re_* > 500. \end{cases}$$

  The shear Reynolds number $Re_* = Re_*(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v})$ is given by [133, 134] :

  $$Re_*(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = \frac{u^* d_*(f_s)}{\nu_l},$$

  where ν_l is the kinematic viscosity of the fluid and

  $$u^* = u^*(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v}) = \sqrt{\frac{\tau(f_s, f_{s_1}, f_{s_2}, \ldots, f_{s_M}, \mathbf{v})}{\rho_l}},$$

is the shear velocity.

Bibliography

[1] J.-L. Boillat and H. Pougatsch. State of the art of sediment management in Switzerland. In *Proceedings International Workshop and Symposium on Reservoir Sedimentation Management*, pages 35–45, 2000.

[2] L. Hakanson. The relationship between salinity, suspended particulate matter and water clarity in aquatic systems. *Ecological research*, 21:75–90, 2005.

[3] C. T Yang. *Erosion and Sedimentation Manual*. U.S. Department of the Interior, bureau of reclamation, sedimentation and river hydraulics group edition, 2006.

[4] W. E. Dietrich. Settling velocity of natural particles. *Water Resources Research*, 18(6): 1615–1626, 1982.

[5] A. A. Brown and R. G. Loucks. Influence of sediment type and depositional processes on stratal patterns in the permian basin-margin lamar limestone. In Robert G. Loucks and J. Frederick Sarg, editors, *Carbonate Sequence Stratigraphy: Recent Developments and Applications*, page 145. American Association of Petroleum Geologists, 1993.

[6] C. P. Holliday, T. C. Rasmussen, and W. P. Miller. Establishing the relationship between turbidity and total suspended sediment concentration. In *Georgia Water Resources Conference*, 2003.

[7] C. T Yang. *Sediment Transport, Theory and Practice*. McGraw-Hill, 1996.

[8] R.W Sternberg, I Berhane, and A.s Ogston. Measurement of size and settling velocity of suspended aggregates on the Northern California continental shelf. *Marine Geology*, 154:43–53, 02 1999.

[9] R. J. Gibbs. Estuarine flocs: Their size, settling velocity and density. *Journal of Geophysical Research: Oceans*, 90(C2):3249–3251, 1985.

[10] E.M. Valentine. *Sediments in the marine environment*. Foundation for water research, 2016.

[11] J. R.Gray, G. Dounglas Glysson, L. M. Turcios, and G. E. Schwarz. Comparability of suspended-sediment concentration and total suspended solids data sediment load from major rivers into puget sound and its adjacent waters. In *U.S Geological survey, Water-resources inverstigations Report*. 2000.

[12] W. Zhang. *Sediment Dynamics*. Springer Netherlands, 2013.

[13] C. Ancey, F. Bigillon, P. Frey, J. Lanier, and R. Ducret. Saltating motion of a bead in a rapid water stream. *Phys. Rev. E*, 66, Sep 2002.

[14] J. V. Baugh and A. J Manning. An assessment of a new settling velocity parameterisation for cohesive sediment transport modelling. *Continental Shelf Research*, 27, 2007.

[15] W. H. Graf. *Hydraulics of sediment transport*. Water Resources Publications, LLC, 1984.

[16] S. Dey. *Fluvial Hydrodynamics: Hydrodynamics and Sediment Transport Phenomena*. GeoPlanet: Earth and Planetary Sciences. Springer, 2014.

[17] W. Wu, W. Rodi, and T. Wenka. 3d numerical modeling of flow and sediment transport in open channels. *Journal of Hydraulic Engineering*, 126(1):4–15, 2000.

[18] H. S. Woo, P. Y. Julien, and E. V. Richardson. Suspension of large concentrations of sands. *Journal of Hydraulic Engineering*, 114(8):888–898, 1988.

[19] S. Katori, M. Mizuguchi, and A. Watanabe. A numerical model of sheet flow sediment transport. In *Coastal Engineering 1996*, pages 3818–3829, 1996.

[20] K. Nadaoka and H. Yagi. Single-phase fluid modelling of sheet-flow toward the development of "numerical mobile bed". In *Coastal Engineering 1990*, pages 2346–2359, 1990.

[21] J. Capecelatro and O. Desjardins. Eulerian–Lagrangian modeling of turbulent liquid–solid slurries in horizontal pipes. *International Journal of Multiphase Flow*, 55, 05 2013.

[22] H. Shi and X. Yu. An effective Euler–Lagrange model for suspended sediment transport by open channel flows. *International Journal of Sediment Research*, 30(4):361 – 370, 2015.

[23] J. R. Finn, M. Li, and S. V. Apte. Particle based modelling and simulation of natural sand dynamics in the wave bottom boundary layer. *Journal of Fluid Mechanics*, 796:340–385, 2016.

[24] S. Apte, K. Mahesh, and T. Lundgren. An Eulerian-Lagrangian Model to Simulate Two-Phase/Particulate Flows in Complex Geometries. In *Center for Turbulence annual research briefs 2003*, 11 2003.

[25] H. H. Hu, N.A. Patankar, and M.Y. Zhu. Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique. *Journal of Computational Physics*, 169(2):427 – 462, 2001.

[26] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics*, 271:285–309, 1994.

[27] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, and J. Périaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *Journal of Computational Physics*, 169(2): 363 – 426, 2001.

[28] F. Golay, D. Lachouette, S. Bonelli, and P. Seppecher. Interfacial erosion: a three-dimensional numerical model. *C. R. Mecanique*, 338(6):333–337, 2010.

[29] D. Lachouette, S. Bonelli, F. Golay, and P. Seppecher. Numerical modelling of soil interface erosion. *Comp. Meth. App. Mech. Engrg*, 200(1-4):383–391, 2011.

[30] J. J. Monaghan. SPH without a tensile instability. *Journal of Computational Physics*, 159 (2):290 – 311, 2000.

[31] J. J. Monaghan, A. Kos, and N. Issa. Fluid motion generated by impact. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 129(6):250–259, 2003.

[32] M. Liu and G.R. Liu. Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of Computational Methods in Engineering*, 17:25–76, 03 2010.

[33] M. A. Nabian and L. Farhadi. Multiphase mesh-free particle method for simulating granular flows and sediment transport. *Journal of Hydraulic Engineering*, 143(4):04016102, 2017.

[34] A. Shakibaeinia and Y. C. Jin. MPS mesh-free particle method for multiphase flows. *Computer Methods in Applied Mechanics and Engineering*, 229-232:13 – 26, 2012.

[35] M. Shademan and N. Nouri. A Lagrangian-Lagrangian model for two-phase bubbly flow around circular cylinder. *Journal of Computational Multiphase Flows*, 6:151–168, 04 2014.

[36] Abbas K. and H. Gotoh. Enhancement of performance and stability of MPS mesh-free particle method for multiphase flows characterized by high density ratios. *Journal of Computational Physics*, 242:211 – 233, 2013.

[37] M. Zanganeh, A. Yeganeh-Bakhtiary, and A. Khairi. Lagrangian coupling two-phase flow model to simulate current-induced scour beneath marine pipelines. *Applied Ocean Research*, 38:64 – 73, 2012.

[38] F. Bombardelli and S. Jha. Hierarchical modeling of the dilute transport of suspended sediment in open channels. *Environmental Fluid Mechanics*, 9:207–235, 04 2008.

[39] E.A. Toorman. Vertical mixing in the fully developed turbulent layer of sediment-laden open-channel flow. *Journal of Hydraulic Engineering*, 134(9):1225–1235, 2008.

[40] T.-J. Hsu, J.T. Jenkins, and P.L.-F. Liu. On two-phase sediment transport: Sheet flow of massive particles. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 460(2048):2223–2250, 2004.

[41] C. Xin, L. Yong, N. Xiaojing, L. Ming, C. Daoyi, and Xiping Y. A general two-phase turbulent flow model applied to the study of sediment transport in open channels. *International Journal of Multiphase Flow*, 37(9):1099 – 1108, 2011.

[42] C. Xin, L. Yong, N. Xiaojing, C. Daoyi, and Y. Xiping. A two-phase approach to wave-induced sediment transport under sheet flow conditions. *Coastal Engineering*, 58(11): 1072 – 1088, 2011.

[43] E. Michaelides, T. C. Clayton, and J. D. Schwarzkopf. *Multiphase Flow Handbook*. CRC Press, Boca Raton, 2 edition, 2016.

[44] W. Weiming. *Computational River Dynamics*. Taylor and Francis, London, 2008.

[45] E. Audusse, S. Boyaval, N. Goutal, M. Jodeau, and P. Ung. Numerical simulation of the dynamics of sedimentary river beds with a stochastic Exner equation. *ESAIM: Proceedings and Surveys*, 48:321–340, 03 2015.

[46] J. Oh and C. Tsai. A comparative study on Lagrangian stochastic models for sediment transport. pages 2370–2378, 05 2011.

[47] E. F. Soares, T. E. Unny, and W. C. Lennox. Conjunctive use of deterministic and stochastic models for predicting sediment storage in large reservoirs: 1. a stochastic sediment storage model. *Journal of Hydrology*, 59(1):49 – 82, 1982.

[48] C. Man and C. Tsai. Stochastic partial differential equation-based model for suspended sediment transport in surface water flows. *Journal of Engineering Mechanics-asce*, 133, 04 2007.

[49] D. A. Lyn. Turbulence models for sediment transport engineering. *ASCE Manual of Practice*, 110:763–825, 05 2008. doi: 10.1061/9780784408148.ch16.

[50] A. Ganjare and A. Patwardhan. Comparison of turbulence models for CFD simulations of sedimentation tank. In *46th national Conference on " Fluid Mechanics and Fluid Power"*, 12 2019.

[51] C. W. Tsai and S-H. Huang. Modeling suspended sediment transport under influence of turbulence ejection and sweep events. *Water Resources Research*, 55(7):5379–5393, 2019.

[52] J. Chauchat and M. Médale. A three-dimensional numerical model for incompressible two-phase flow of a granular bed submitted to a laminar shearing flow. *Comp. Meth. App. Mech. Engrg*, 199(9-12):439–449, 2010.

[53] D. H. Nguyen, F. Levy, D. P. Van Bang, S. Guillou, K. D. Nguyen, and J. Chauchat. Simulation of dredged sediment releases into homogeneous water using a two-phase model. *Adv. Water Resources*, 48:102–112, 2012.

[54] R. Ruiz and I. Lunati. Mixed finite element – discontinuous finite volume element discretization of a general class of multicontinuum models. *J. Comput. Phys.*, 322: 666–688, 2016.

[55] J. Chauchat, M. Ouriemi, P. Aussillous, and E. Guazzelli. A 3d two-phase numerical model for sediment transport. In *International Conference on Multiphase Flow*, 05 2010.

[56] M. Ishi and T. Hibiki. *Thermo-fluid Dynamics of Two-Phase Flow*. Springer, 2010.

[57] Z. Cao, L. Wei, and J. Xie. Sediment-laden flow in open channels from two-phase flow viewpoint. *Journal of Hydraulic Engineering*, 121(10):725–735, 1995.

[58] S. K. Jha and F. A. Bombardelli. Toward two-phase flow modeling of nondilute sediment transport in open channels. *Journal of Geophysical Research: Earth Surface*, 115(F3), 2010.

[59] L. Lixin, X. Yu, and F. Bombardelli. A general mixture model for sediment laden flows. *Advances in Water Resources*, 107, 06 2017.

[60] D. Zhong, G. Wang, and B. Wu. Drift velocity of suspended sediment in turbulent open channel flows. *Journal of Hydraulic Engineering*, 140:35–47, 01 2014.

[61] L. C. van Rijn. Sediment transport, part ii: Suspended load transport. *Hydraulic engineering*, 110:1613–1641, 1984.

[62] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79: 763–813, 12 2008.

[63] A. Huerta, T. Belytschko, S. Fernández-Méndez, T. Rabczuk, Xiaoying Z., and M. Arroyo. *Meshfree Methods*, pages 1–38. American Cancer Society, 2017. ISBN 9781119176817.

[64] P. Koumoutsakos. Multiscale flow simulations using particles. *Annual Review of Fluid Mechanics*, 37:–487, 01 2005.

[65] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, dec 1977.

[66] L. B. Lucy. Numerical approach to the testing of the fission hypothesis. *Astron. J.*, 82: 1013–1024, 12 1977.

[67] R. I. Ferguson and M. Church. A simple universal equation for grain settling velocity. *J. Sedimentary Research*, 74(6):933–937, 2004.

[68] D. A. Lyn. Resistance in flat-bed sediment-laden flows. *Journal of Hydraulic Engineering*, 117(1):94–114, 1991.

[69] J.J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110(2):399 – 406, 1994.

[70] D. Violeau and R. Issa. Numerical modelling of complex turbulent free-surface flows with the SPH method: An overview. *International Journal for Numerical Methods in Fluids*, 53:277 – 304, 01 2007.

[71] J. Fang, A. Parriaux, M. Rentschler, and C. Ancey. Improved SPH methods for simulating free surface flows of viscous fluids. *Applied Numerical Mathematics*, 59(2):251 – 271, 2009.

[72] A.J.C. Crespo, M. Gómez-Gesteira, and R.A. Dalrymple. 3d SPH simulation of large waves mitigation with a dike. *Journal of Hydraulic Research*, 45(5):631–642, 2007.

[73] L. Eun-Sug, D. Violeau, I. Réza, and S. Ploix. Application of weakly compressible and truly incompressible SPH to 3-d water collapse in waterworks. *Journal of Hydraulic Research*, 48:50–60, 2010.

[74] R.A. Dalrymple and B.D. Rogers. Numerical modeling of water waves with the SPH method. *Coastal Engineering*, 53(2):141 – 147, 2006.

[75] P. J. Roache. *Fundamentals of Computational Fluid Dynamics*. Hermosa Publishers, 1998.

[76] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics. Birkhaüser-Verlag, ETH Zurich, 1990.

[77] R. Eymard, T. Gallouet, and R. Herbin. *Finite Volumes Methods*. Handbook of numerical analysis. Elsevier Science B.V, 2000.

[78] R. Glowinski. *Finite Element Method For Incompressible Viscous Flow*, volume IX of *Handbook of Numerical Analysis (P.G. Ciarlet, J.L. Lions eds)*, pages 3–1176. Elsevier, Amsterdam, 2003.

[79] R. Rannacher and T. Richter. An adaptive finite element method for fluid-structure interaction problems based on a fully Eulerian formulation. In H.J. Bungartz, Miriam Mehl, and M. Schäfer, editors, *Fluid Structure Interaction II*, pages 159–191, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[80] C. S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10(2):252 – 271, 1972.

[81] H. J. Bungartz, M. Schäfer, and M. Mehl. *Fluid Structure Interaction II: Modelling, Simulation, Optimisation*. Springer, 2006.

[82] C.W Hirt and B.D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201 – 225, 1981.

[83] W. Rider and Douglas K. Reconstructing volume tracking. *Journal of Computational Physics*, 141, 03 1999.

[84] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269 – 277, 1995.

[85] J. Sethian and P. Smereka. Level set methods for fluid interfaces. *Annu. Rev. Fluid Mech*, 35:341–72, 01 2003.

[86] D. A. Di Pietro, S. Lo Forte, and N. Parolini. Mass preserving finite element implementations of the level set method. *Applied Numerical Mathematics*, 56(9):1179 – 1195, 2006. Numerical Methods for Viscosity Solutions and Applications.

[87] G.-H Cottet and E. Maitre. A level set method for fluid-structure interactions with immersed surfaces. *Mathematical Models and Methods in Applied Sciences*, 16:415–438, 04 2006.

[88] S. Gihun and H. Nahmkeon. A level set formulation for incompressible two-phase flows on nonorthogonal grids. *Numerical Heat Transfer, Part B: Fundamentals*, 48(3):303–316, 2005.

[89] F. H. Harlow and J. E. Welch. Numerical calculation of time dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8, 1965.

[90] P. Raad and R. Bidoae. The three-dimensional Eulerian–Lagrangian marker and micro cell method for the simulation of free surface flows. *Journal of Computational Physics*, 203:668–699, 03 2005.

[91] C. W. Hirt. Volume-fraction techniques: powerful tools for wind engineering. *Journal of Wind Engineering and Industrial Aerodynamics*, 46-47:327 – 338, 1993. Proceedings of the 1st International on Computational Wind Engineering.

[92] A. N. Papanicolaou, M. Elhakeem, G. Krallis, S. Prakash, and J. Edinger. Sediment transport modeling review ;current and future developments. *Journal of Hydraulic Engineering*, 134(1):1–14, 2008.

[93] *Delft3D Open Source Community*, Accessed April 20, 2020. URL https://oss.deltares.nl/web/delft3d/home.

[94] M. Pravash, I. Popescu, Sanjay G., O. Amgad, S. Kees, K. Yuichi, and D. Solomatine. Delft3d morphological modelling of sediment management in daily Peaking Run-Of-the-River hydropower (PROR) reservoirs in Nepal. In *International Commission on Large Dams*, 07 2017.

[95] A. Siqueira, M. Fiedler, and E. Yassuda. Delft3d morphological modeling downstream of sergio motta reservoir dam. In *IAEG/AEG Annual Meeting Proceedings*, pages 17–24, 08 2018.

[96] G. Gootjes. Dunes as source of sediment for delft3d-mor, 01 2000.

[97] *Sediment Transport Model*, Accessed April 20, 2020. URL https://www.flow3d.com/modeling-capabilities/sediment-transport-model/.

[98] G. Wei, M. Grünzner J. Brethour, and J. Burnham. *The Sedimentation Scour Model in FLOW-3D*. Flow Science, technical note fsi-14-tn-99 edition, 2014.

[99] W. Al-Mussawi and K. Halah. Three-dimensional numerical simulation of local scour around circular bridge pier using FLOW-3D software. *IOP Conference Series: Materials Science and Engineering*, 745, 03 2020.

[100] M. Nazari-Sharabian, A. Nazari-Sharabian, M. Karakouzian, and M. Karami. Sacrificial piles as scour countermeasures in river bridges - a numerical study using FLOW-3D. *Civil Engineering Journal*, 6, 03 2020.

[101] K. Bina, S. Aboutalebi, Seyed S. Zamanieh S., and S. A. Khoshnevis. Numerical study of sediment flow over bottom intake racks with FLOW-3D. *International Journal of Scientific Research and Management*, 7, 07 2019.

[102] *OpenFOAM: the Open Source CFD Toolbox User Guide*. OpenFOAM, the free software foundation edition, 2014.

[103] O. Kazuyuki, S. Takahiro, and N. Hajime. 3D numerical model of sediment transport considering transition from bed-load motion to suspension —application to a scour upstream of a cross-river structure—. *Journal of JSCE*, 4:173–180, 10 2016.

[104] C. Bonamy, Chauchat J., Zhen C., T. Nagel, and T-J Hsu. sedFoam, a OpenFOAM solver for sediment transport. In *12th OpenFoam Workshop*, Exeter, United Kingdom, July 2017.

[105] T. Nagel, J. Chauchat, C. Bonamy, L. Xiaofeng, Z. Cheng, and T.J. Hsu. Three-dimensional scour simulations with a two-phase flow model. *Advances in Water Resources*, page 103544, 02 2020.

[106] Z. Cheng, T. J. Hsu, and J. Calantoni. Sedfoam: A multi-dimensional Eulerian two-phase model for sediment transport and its application to momentary bed failure. *Coastal Engineering*, 119:32–50, 01 2017.

[107] *FLUENT 6.2 User's Guide*. Fluent Inc., 2005.

[108] V. Vuik. Numerical modeling of sediment transport over hydraulic structures . Master's thesis, TU Delft, 2010.

[109] P. Welahettige and K. Vaagsaether. Comparison of OpenFOAM and ANSYS FLUENT. In *EUROSIM Congress on Modelling and Simulation*, pages 1005–1012, 2016.

[110] A. Bonito, A. Caboussat, M. Picasso, and J. Rappaz. A numerical method for fluid flows with complex free surfaces. In Roland Glowinski and Pekka Neittaanmäki, editors, *Partial Differential Equations*, volume 16 of *Computational Methods in Applied Sciences*, pages 187–208. Springer Netherlands, 2008.

[111] A. Caboussat. A numerical method for the simulation of free surface flows with surface tension. *Computers and Fluids*, 35(10):1205–1216, 2006.

[112] A. Caboussat, P. Clausen, and J. Rappaz. Numerical simulation of two-phase flow with interface tracking by adaptive Eulerian grid subdivision. *Math. Comput. Modelling*, 55: 490–504, 2012.

[113] A. Caboussat, M. Picasso, and J. Rappaz. Numerical simulation of free surface incompressible liquid flows surrounded by compressible gas. *J. Comput. Phys.*, 203(2):626–649, 2005.

[114] P. C. Carman. Fluid flow through granular beds. *Chemical Engineering Research and Design*, 75:S32 – S48, 1997.

[115] S. Whitaker. Flow in porous media i: A theoretical derivation of Darcy's law. *Transport in Porous Media*, 1:3–25, 03 1986.

[116] W. Noh and P. Woodward. SLIC (Simple Line Interface Calculation. In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[117] V. Maronnier, M. Picasso, and J. Rappaz. Numerical simulation of three dimensional free surface flows. *Int. J. Num. Meth. Fluids*, 42(7):697–716, 2003.

[118] V. H. Laurmaa. *An octree-based semi-lagrangian free surface flow solver*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2016.

[119] R. Glowinski. Numerical methods for fluids. In *Handbook of Numerical Analysis*, volume 9. Elsevier, 2003.

[120] A. Caboussat, S. Boyaval, and A. Masserey. On the modeling and simulation of non-hydrostatic dam break flows. *Computing and Visualization in Science*, 14(8):401–417, 2013.

[121] S. Boyaval, A. Caboussat, A. Mrad, M. Picasso, and G. Steiner. A semi-lagrangian splitting method for the numerical simulation of sediment transport with free surface flows. *Computers & Fluids*, 172:384–396, 2018.

[122] N. James, A. Caboussat, S. Boyaval, and M. Picasso. Numerical simulation of 3D free surface flows, with multiple incompressible immiscible phases. Applications to impulse waves. *International Journal for Numerical Methods in Fluids*, 76(12):1004–1024, 2014.

[123] I. Mamoru and Z. Novak. Drag coefficient and relative velocity in bubbly, droplet or particulate flows. *AIChE Journal*, 25(5):843–855, 1979.

[124] F. Concha and M. C. Bustos. Settling velocities of particulate systems, 6. Kynch sedimentation processes : batch settling. *Int. J. Miner. Process.*, 32:193–212, 1991.

[125] G. J. Kynch. A theory of sedimentation. *Transactions of the Faraday Society*, 48:166–176, 1952.

[126] Helge. H., N. Henrik. R., and Aslak. T. Maximum principles for a class of conservation laws. *SIAM Journal on Applied Mathematics*, 55(3):651–661, 1995.

[127] D. P. Van Bang, E. Lefrançois, G. Ovarlez, and F. Bertrand. MRI experimental and 1D FE-FCT numerical investigation of the sedimentation and consolidation. In *Seventh international conference on hydroinformatics*, volume 1, pages 497–504, 2006.

[128] M. Ungarish. On the modeling and investigation of polydispersed rotating suspensions. *Int. J. Multiphase Flow*, 21:262–284, 1995.

[129] G. Blom and R. Hans Aalderink. Calibration of three resuspension/sedimentation models. *Water Sci Technol*, 37(3):41–49, 1998.

[130] J. E. Filostrat. *Estimation of Sediment resuspension and deposition in coastal waters*. PhD thesis, University of New Orleans, 2014.

[131] M.F Ahmad, P. Dong, M.Mamat, W.B Wan Nik, and M. H. Mohd. The critical shear stresses for sand and mud mixture. *Applied mathematical sciences*, 5:53–71, 2011.

[132] L. C. van Rijn. *Principles of sediment transport in rivers, estuaries and coastal seas*. Lectures in Mathematics. Aqua Publications, Universitet Utrech, delft hydraulics, 1993.

[133] M. P. Lamb, W. E. Dietrich, and J. G. Venditti. Is the critical shields stress for incipient sediment motion dependent on channel-bed slope? *Journal of geophysical research*, 113, 2008.

[134] G. T. Torok, J. Joza, and S. Baranya. A shear Reynolds number-based classification method of the non-uniform best load transport. *Water*, 11:73, 2019.

[135] A. Bonito, M. Picasso, and M. Laso. Numerical simulation of 3D viscoelastic flows with free surfaces. *J. Comput. Phys.*, 215(2):691–716, 2006.

[136] V. Maronnier, M. Picasso, and J. Rappaz. Numerical simulation of free surface flows. *J. Comput. Phys.*, 155:439–455, 1999.

[137] O. Pironneau, J. Liou, and T. Tezduyar. *Characteristic-Galerkin and Galerkin/Least-Squares Space-Time Formulations for the Advection-Diffusion Equa- tion with Time-Dependent Domain*, volume 100 of *Computer Methods in Applied Mechanics and Engineering*, pages 117–141. 1992.

[138] A. Quarteroni and L. Formaggia. *Mathematical Modelling and Numerical Simulation of the Cardiovascular System*, volume 12 of *Handbook of Numerical Analysis (P.G. Ciarlet, J.L. Lions eds)*, chapter Modelling of Living Systems, pages 3–127. Elsevier, 2004.

[139] H. Wang, H.K. Dahle, R.E. Ewing, M.S. Espedal, R.C. Sharpley, and S. Man. An ELLAM scheme for advection-diffusion equations in two dimensions. *SIAM Journal on Scientific Computing*, 20:2160–2194, 1999.

[140] V. Laurmaa, M. Picasso, and G. Steiner. An octree-based adaptive semi-Lagrangian VOF approach for simulating the displacement of free surfaces. *Computers and Fluids*, 131: 190–204, 2016.

[141] D.N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the stokes equations. *Calcolo*, 21(4):337–344, 1984.

[142] R. Franke. Scattered data interpolation: Test of some methods. *Math Comput*, 38: 181–200, 01 1982.

[143] S. Osher. Riemann solvers, the entropy condition, and difference approximations. *SIAM Journal on Numerical Analysis*, 21(2):217–235, 1984.

[144] R. J. LeVeque. Finite difference methods for ordinary and partial differential equations : steady-state and time-dependent problems.

[145] J. Kusuma, A. Ribal, and A. Mahie. On ftcs approach for box model of three-dimension advection-diffusion equation. *International Journal of Differential Equations*, 2018:1–9, 11 2018.

[146] M. Shashkov and B. Wendroff. The repair paradigm and application to conservation laws. *J. Comput. Phys.*, 198(1):265–277, 2004.

[147] K. D. Nguyen, S. Guillou, J. Chauchat, and N. Barbry. A two-phase numerical model for suspended-sediment transport in estuaries. *Adv. Water Resources*, 32:1187–1196, 2009.

[148] S. Badr, G. Gauthier, and P. Gondret. Erosion threshold of a liquid immersed granular bed by an impinging plane liquid jet. *Physics of Fluids*, 26:023302, 2014.

[149] D. Nguyen, M. U. Zapata, G. Gauthier, P. Gondret, and D. P. Van Bang. A two phase numerical model for the water injection dredging (WID) technology: An unified formulation for continuum mechanic. In *11th International Conference on Hydroinformatics, CUNY Academic works*, pages 1–6, 2014.

[150] G. Epely Chauvin, G. De Cesare, and S. Schwindt. Modelling of plunge pool scour evolution in non-cohesive sediments. *Engineering Applications of Computational Fluid Mechanics*, 8(4):477–487, 2014.

[151] S. Pagliara, W. H. Hager, and H.-E. Minor. Hydraulics of plane plunge pool scour. *J. Hydr. Engrg*, 132(5):450–461, 2006.

[152] S. Pagliara, W. H. Hager, and J. Unger. Temporal evolution of plunge pool scour. *J. Hydr. Engrg*, 134:1630–1638, 2008.

[153] S. Manenti, S. Sibilla, M. Gallati, G. Agate, and R. Guiandalini. SPH simulation of sediment flushing induced by a rapid water flow. *J. Hydraulic Engineering*, 138:272–284, 2012.

[154] J.-L. Boillat, G. De Cesare, A. Schleiss, and C. Oehy. Successful sediment flushing conditions in alpine reservoirs. In *Proceedings International Workshop and Symposium on Reservoir Sedimentation Management*, pages 47–59, 2000.

[155] W. Hackbusch. *Multi-grid methods and applications*. Springer Verlag, 1985.

[156] M. Khanpour, A.R. Zarrati, M. Kolahdoozan, A. Shakibaeinia, and M. Amirshahi. Mesh-free SPH modeling of sediment scouring and flushing. *Computers & Fluids*, 129, 02 2016.

[157] E. D'Ambrosio, F. Blanc, and E. Lemaire. Viscous resuspension of non-Brownian particles: determination of the concentration profiles and particle normal stresses. *arXiv e-prints*, art. arXiv:1907.01793, July 2019.

[158] A. Acrivos, Roberto Mauri, and X. Fan. Shear-induced resuspension in a Couette device. *International Journal of Multiphase Flow*, 19:797–802, 10 1993.

[159] S. Rui, X. Heng, and S. Kyle. Particle dynamics in self-generated dunes over a range of hydraulic and sediment transport conditions using LES–DEM. *arXiv e-prints*, art. arXiv:1610.03397, October 2016.

[160] A. G. Kidanemariam and M. Uhlmann. Interface-resolved direct numerical simulation of the erosion of a sediment bed sheared by laminar channel flow. *International Journal of Multiphase Flow*, 67:174 – 188, 2014.

[161] S. Rui. Sediment micromechanics in sheet flows induced by asymmetric waves: A CFD-DEM study. *Computers & Geosciences*, 96, 04 2016.

Arwa Mrad

Chemin du Suchet 1
1024 Ecublens, Switzerland
+41 786836390
mrad.arwa@gmail.com
Linkedin

Education

2016–present **Ph.D. in Applied Mathematics**, *Ecole Polytechnique Fédérale de Lausanne*, Lausanne, Switzerland.

2012–2015 **Engineering Degree in Modeling for Industry & Services**, *National Engineering School of Tunis, ENIT (**EUR-ACE** labelled)* , Tunis, Tunisia.

2015 **Master Project (6 months)**, *Ecole Polytechnique Fédérale de Lausanne*, MATHICSE Group, Lausanne, Switzerland.

2014 **Semester Project (3 months)**, *National Engineering School of Tunis*, Singal processing unit, Tunis, Tunisia.

2010–2012 **National Engineering Exam Preparation**, *National Preparatory Institute for Engineering Studies*, Tunis, Tunisia.

2006–2010 **New Era High School**, Hammamet, Tunisia.

Work Experience

2016–present **C++ Development**, *Ecole Polytechnique Fédérale de Lausanne*, Ycoor Systems (cfsFlow++ project), Lausanne, Switzerland.

2016–present **Research & Teaching Assistantship**, *Ecole Polytechnique Fédérale de Lausanne*, MATHICSE Group, Lausanne, Switzerland.

2015–2016 **Software Engineering -Embedded C**, *SAGEMCOM* , Tunis, Tunisia.

2014 **Summer Internship (2 months)** , *Karunya University*, Computer Science & Technology department, Coimbatore, India.

2013 **Summer Internship (1 month)**, *SUPER CABLES Company*, Grombalia, Tunisia.

Academic Honors and Awards

2016–2019 Swiss Confederation Excellence Fellowship

Technical competences & computer skills

Programming C, C++, Python, Java

Data analysis MATLAB, Maple, Octave, Paraview, Excel, G3data, Freefem++

Mathematics & algorithms Numerical analysis, optimization techniques, statistics & data analysis, machine learning techniques, sorting, hashing and graphs algorithms, text algorithms, and scientific algorithms

Editing Latex, Powerpoint, Beamer

Other (IT) Linux, GIT, SVN, OpenMP, BLAS/LAPACK, PETSc, GMSH

213

Publications

Aug. 2018 **A semi-Lagrangian splitting method for the numerical simulation of sediment transport with free surface flows**, S. Boyaval, A. Caboussat, *A. Mrad*, M. Picasso, G. Steiner, *"Computers & Fluids", Volume 172, 30 August 2018, Pages 384-396*

Languages

Arabic (native)**, French** (near native)**, English** (fluent)**, Spanish** (basic)

Extracurricular activities

- Fond of mountaineering (climbing and hiking), skiing, and cycling. I love singing and nature photography.
- Strong interest in the rising field of machine learning, completed a few courses in the E-learning platform Coursera in **machine learning**, **deep learning** and **TensorFlow**.

Personal information

Age: 28 (03.02.1992); Marital status: single; Nationality: Tunisian. International driving licence holder.