Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

# Avalanche Simulation Using the Shallow-Water Equations in ExaHyPE

Ujwal Shende

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

# Avalanche Simulation Using the Shallow-Water Equations in ExaHyPE

| | |
|---|---|
| Author: | Ujwal Shende |
| Examiner: | Univ.-Prof. Dr. Michael Bader |
| 1st advisor: | M.Sc. Mario Wille |
| 2nd advisor: | Dr. Steven Gibbons |
| Submission Date: | November 9th, 2023 |

I hereby declare that this thesis is entirely the result of my own work except where other-
wise indicated. I have only used the resources given in the list of references.

November 9th, 2023 Ujwal Shende

# Acknowledgments

First of all, I would like to express my deepest gratitude to my advisor, Mario Wille, for his invaluable patience and guidance. His constant motivation, lightning-fast replies to my queries and feedback to make the thesis more organised, made this an incredible learning experience.

I would also like to thank the scientists from the Norwegian Geotechnical Insititute (NGI): Dr. Steven Gibbons, Dr. Callum Tregaskis and Dr. Finn Løvholt for providing the governing equations for the thesis and clarifying my doubts.

Furthermore, I would like to thank Univ.-Prof. Dr. Michael Bader for being the supervisor for my thesis.

Finally, I want to deeply thank my family. No combination of the twenty-six alphabets is enough to express my gratitude towards them. My CSE journey would have been impossible without their support.

*"If I have seen further, it is by standing on the shoulders of giants"*

*-Sir Isaac Newton*

# Abstract

Avalanches pose significant threats to human life, infrastructure, and the environment, making accurate prediction and mitigation strategies essential. Avalanches are modelled using the shallow-water equations and the friction law for granular materials flowing down an inclined plane. The resulting equations are known as depth-averaged avalanche equations. This work implements the depth-averaged avalanche equations in ExaHyPE 2, a high-performance computing framework for simulating hyperbolic PDEs. A preprocessing routine is also implemented to facilitate the calculation of complex derivative terms in the equations. The validation of the solver application is mainly done by simulating the flow of a circular patch of masonry sand down a rectangular plane inclined at an angle of $35°$. The finite volume method with Rusanov flux and adaptive time stepping is used for the simulations. A comparison between the flow of masonry sand and carborundum particles is done to qualitatively validate the dependence of friction on material parameters. The results of the validation are found to be in agreement with the physics of the governing equations. The availability of the static adaptive mesh refinement feature in ExaHyPE 2 is demonstrated by refining one-half of the domain while coarsening the other half. The solver application is a high-performance computing solution that provides detailed insights into avalanche dynamics.

# Contents

# 1 Introduction

Gravity-driven geophysical events sometimes involve the flow of a dry granular material. Landslides, rock avalanches and pyroclastic flows are examples of natural hazards where a granular mass flows down a slope. This thesis aims to develop an application in ExaHyPE 2 to simulate avalanches by considering them as the flow of granular material. This research avenue holds immense potential not only for understanding natural phenomena but also for a wide range of practical applications.

## 1.1 Motivation of the Thesis

The motivation behind this thesis is rooted in several fundamental aspects of science, engineering, and societal well-being.

Studying avalanches as granular flows allows us to gain deep insights into the intricate dynamics of these catastrophic events. By comprehending the behaviour of granular materials during avalanches, we can develop more accurate predictive models. Such models are invaluable for early warning systems, enabling us to mitigate the impact of avalanches on vulnerable communities.

In regions prone to avalanches, understanding granular material flow is essential for evaluating the environmental impact of these events. By simulating avalanches as granular flows, researchers can assess how ecosystems are affected, enabling us to develop strategies to preserve biodiversity and natural habitats in avalanche-prone areas.

Engineers can benefit immensely from a comprehensive understanding of granular material flow during avalanches. Simulations in this context provide valuable data for designing infrastructure such as avalanche barriers and protective structures. Optimising these designs ensures the safety of communities living in avalanche-prone regions.

Delving into the dynamics of granular materials can also drive innovations in material science. By understanding how different materials behave during avalanches, scientists can develop novel materials with properties that could revolutionise various industries, from construction to transportation. These advancements have the potential to significantly enhance the durability and safety of structures and vehicles.

Simulating avalanches as granular flows offers an exciting opportunity to advance our fundamental understanding of complex systems. Granular materials exhibit fascinating behaviours that challenge existing theories and provide avenues for groundbreaking discoveries.

## 1.2 Related Work

Researchers have explored various approaches to understand and predict the behaviour of avalanches. One significant line of research has focused on developing and refining granular flow models to simulate avalanche dynamics accurately. Early studies, such as those by Savage and Hutter [24], laid the foundation by introducing continuum-based models to describe granular flow. These models treated avalanches as a continuous flow of granular material, considering factors like shear rate, stress, and density to predict avalanche behaviour.

Researchers like Gray and Edwards [13] investigated the rheology of granular materials, providing valuable insights into the flow behaviour of granular media. Their work contributed to the formulation of constitutive equations used in granular flow models, enhancing the accuracy of avalanche simulations. They modified the basal friction laws derived by Pouliquen and Forterre [10] and incorporated it into the source term of their equations to model avalanches more accurately.

Additionally, computational techniques have been employed to simulate granular avalanches in complex terrains. Studies by Patra *et al.* [19] utilised numerical simulations based on a finite volume Gudonov solver with adaptive mesh refinement (AMR) to solve the governing model equations. They designed their simulation code to run on distributed memory supercomputers using MPI. Their software also directly connects to GIS (Geographic Information System) to dynamically obtain the topographic data as needed by the simulation.

Furthermore, Ma *et al.* [17] established a numerical model for the generation and propagation of tsunami waves by granular landslides. They simulated tsunami wave generation using the 3D non-hydrostatic wave model NHWAVE, which is capable of capturing wave dispersion efficiently using a small number of discretised vertical levels. Their model also illustrates a complex interplay between granular landslides and tsunami waves, and it reasonably predicts not only the tsunami wave generation but also the granular landslide motion from initiation to deposition.

Ashtiani *et al.* [2] used a 2D fourth-order Boussinesq-type numerical model to estimate the impact of landslide-generated waves in dam reservoirs. They validated their model against available three-dimensional experimental data. They also employed their numerical model to investigate the impact of landslide-generated waves in two real cases.

Longchamp *et al.* [16] performed experiments to understand the propagation and spreading of granular masses released at the top of a simple geometry. They measured the flow of granular material flowing down a plane inclined at $45°$ using a high-speed camera. They also simulated this granular flow using a numerical model based on the shallow-water equations and the Mohr-Coulomb friction law. They compared the results obtained through simulations and experiments and found them to be in good agreement. The magnitude of velocity measured in experiments matched precisely with what they got in simulations.

Rauter *et al.* [22] presented a novel three-dimensional granular landslide and tsunami model, which they applied to the 2014 Lake Askja landslide tsunami. Their model captures the complete event chain from the landslide dynamics to the wave generation and inundation. The model gives deep insights into the physical landslide processes and improves our understanding and prediction capabilities of frequent and catastrophic landslide tsunamis.

## 1.3 Structure of the Thesis

The following Chapter 2 presents the derivation of the depth-averaged avalanche equations, which are the governing equations used in this thesis for simulation. The chapter is divided into three sections. Section 2.1 presents a brief description of the shallow-water equations which form the basis for the depth-averaged avalanche equations. Section 2.2 explains the formulation of friction law for granular flows, which is used in the source term of the governing equations. Finally, Section 2.3 presents an overview of how the complete depth-averaged avalanche equations are derived, assuming that the granular material behaves like a viscous fluid.

Chapter 3 presents an introduction to ExaHyPE 2, which is the simulation engine used for this thesis. The chapter also presents algorithms that can be used to develop an application in ExaHyPE 2 to simulate the shallow-water equations.

Chapter 4 presents algorithms for developing the ExaHyPE 2 solver application to simulate avalanches using the depth-averaged avalanche equations. Section 4.1 presents the algorithms used for preprocessing of a patch, which is done to calculate derivatives using a finite-difference method. Section 4.2 presents the algorithms used for the development and configuration of the solver application in ExaHyPE 2.

Chapter 5 presents results in the form of surface and line plots obtained by evaluating the developed ExaHyPE 2 application. Section 5.1 presents results that validate the implementation of the algorithms in ExaHyPE 2. Section 5.1.4 compares the avalanche of two different granular materials, namely sand and carborundum. Section 5.2 presents results obtained by applying static adaptive mesh refinement (static AMR). The results are also compared with the the ones obtained in Section 5.1.2 without static AMR.

Finally, Chapter 6 presents the conclusion of this work.

# 2 Theory

The mathematical model used to simulate avalanches is based on the shallow-water equations and the friction law for granular material flowing down an inclined plane. The resulting equations that govern the propagation of avalanches are known as depth-averaged avalanche equations. This chapter explains the important components of these equations and the steps behind their derivation.

## 2.1 The Shallow-Water Equations

Fluid dynamics is a fascinating field of study that explores the behaviour of fluids in various scenarios. One important aspect of fluid dynamics is the understanding of the flow below a pressure surface in a fluid. This flow is described by a set of equations known as the shallow-water equations (SWE), also referred to as the Saint-Venant equations, as their derivation was first presented by Saint-Venant in [5]. The shallow-water equations are a set of hyperbolic partial differential equations that govern the flow below a pressure surface in a fluid. These equations are derived from depth-integrating the Navier-Stokes equations with the assumption that the horizontal length scale is much greater than the vertical length scale. This assumption allows for the removal of the vertical velocity component from the equations. From tsunamis to storm surges, these equations offer valuable insights into the dynamics of shallow-water motion.

Depth-averaging of governing equations is a prevalent approach amongst researchers for modelling complex flows. Researchers have proposed various reformulation methods to increase the computational efficiency of models based on shallow-water flows. Bristeau *et al.* [4] presented an original derivation process of a non-hydrostatic shallow-water type model to approximate Euler and Navier-Stokes systems with a free surface. Escalante and Luna [9] proposed a novel first-order reformulation of the Boussinesq-type systems. Their model avoids the use of high-order derivatives, which require a large stencil size, hence reducing the computational complexity of the model. Their model also accounts for the effect of time-dependent bathymetry.

The shallow-water equations for an inviscid fluid in component form in a two-dimensional domain (horizontal coordinates) can be written as follows:

**Continuity Equation:**

$$\frac{\partial h}{\partial t} + \frac{\partial (h\bar{u})}{\partial x} + \frac{\partial (h\bar{v})}{\partial y} = 0. \tag{2.1}$$

**Momentum Equations:**

$$\frac{\partial(h\bar{u})}{\partial t} + \frac{\partial}{\partial x}\left(h\bar{u}^2 + \frac{1}{2}gh^2\right) + \frac{\partial(h\bar{u}\bar{v})}{\partial y} = -gh\frac{\partial b}{\partial x}, \tag{2.2}$$

$$\frac{\partial(h\bar{v})}{\partial t} + \frac{\partial(h\bar{u}\bar{v})}{\partial x} + \frac{\partial}{\partial y}\left(h\bar{v}^2 + \frac{1}{2}gh^2\right) = -gh\frac{\partial b}{\partial y}, \tag{2.3}$$

where:

- $h$ is the water height.
- $\bar{u}$ and $\bar{v}$ are the depth-averaged velocity components in the $x$- and $y$-direction, respectively.
- $g$ is the acceleration due to gravity.
- $b$ is the bathymetry of the bottom.

## 2.2 Friction Law for Granular Flows

Granular flows down rough inclined planes are a common phenomenon encountered in various engineering and geophysical applications. These flows involve the movement of particulate solids, such as minerals and cereals, on inclined surfaces. Understanding the behaviour and characteristics of granular flows is crucial for designing transportation systems and predicting natural events like landslides and rock avalanches. The flow of granular materials on inclined surfaces has been the subject of extensive research due to its relevance in various fields. These flows are characterised by the interplay between gravity and friction forces on the inclined plane. The behaviour of granular flows depends on factors such as the surface roughness, particle size, and inclination angle of the plane [20]. Studying granular flows down rough inclined planes provides insights into the rheological properties of particulate systems and helps develop predictive models for such flows.

To investigate the scaling laws in granular flows down rough inclined planes, Pouliquen [20] conducted a series of experiments using different systems of glass beads. Their experimental setup consisted of an inclined plane with adjustable inclination angles, as shown in Figure 2.1. The glass beads were flowed down the inclined plane, and various parameters, such as the mean velocity and thickness of the granular layer, were measured. The measurements were performed for different surface roughness conditions and inclination angles to capture a wide range of flow behaviours.

Their experimental results revealed the existence of steady uniform flows in a specific range of inclination angles and layer thicknesses. Experimental measurements have shown the existence of two critical angles. An initially static granular layer starts to flow

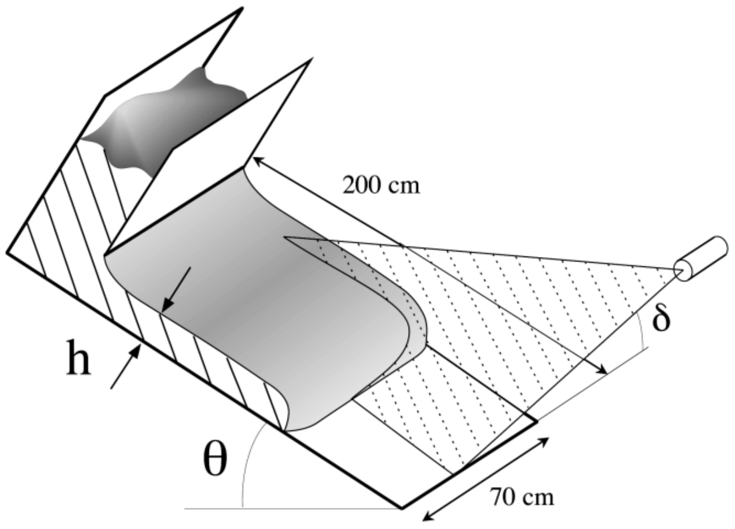


Figure 2.1: Schematic of experimental setup used by Pouliquen [20].

when the inclination reaches a critical value $\theta_{start}$. To stop the flow, the angle of inclination needs to be decreased to $\theta_{stop}$. These flows were characterised by constant mean velocities and thicknesses. The data obtained for different systems of beads were found to collapse into a single curve, as shown in Figure 2.2, when properly scaled. The scaling was based on the measurement of the minimum thickness $h_{stop}(\theta)$ necessary to observe a steady uniform flow at a given inclination angle. This empirical scaling allowed for the prediction of mean flow velocities without the need for direct velocity measurements. It is evident from Figure 2.2 that the Froude number $u/\sqrt{gh}$ varies linearly with $h/h_{stop}(\theta)$ and can be written as

$$\frac{u}{\sqrt{gh}} = \beta \frac{h}{h_{stop}(\theta)}, \tag{2.4}$$

with $\beta = 0.136$ independent of the inclination, the bead size, and the roughness of the bed. This value of $\beta$ corresponds to the best fit in Figure 2.2.

To infer the coefficient of friction $\mu$, it is assumed that at an angle $\theta$ and thickness $h_{stop}(\theta)$, the friction force is enough to balance the gravitational force down the slope. Hence, one can write

$$\mu(u, h) = \tan\theta. \tag{2.5}$$

It is important to note that the above dynamic friction coefficient $\mu(u, h)$, which is assumed to be a function of the thickness $h$ and the mean velocity $u$, is not a property of the bulk material but describes the interaction between the material and the rough surface. Pouliquen [20] also observed that when performing an experiment at a given angle $\theta$ and with a thickness $h$, the granular layer adjusts its velocity according to Equation 2.4. In other terms, fixing the inclination is equivalent to fixing the friction coefficient.

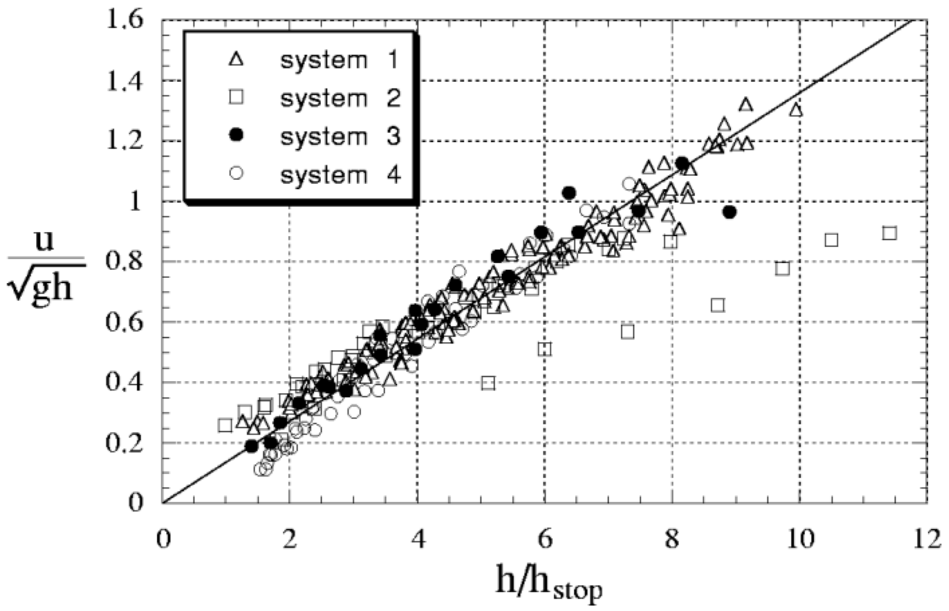


Figure 2.2: Froude number as a function of $h/h_{stop}(\theta)$ for the four systems of beads and for different inclination angles [20].

From the curves in Figure 2.3, Pouliquen [20] experimentally obtained an analytical expression describing the relationship between $h_{stop}$ and $\theta$ as

$$\tan\theta = \tan\theta_1 + (\tan\theta_2 - \tan\theta_1)\exp\left(\frac{-h_{stop}}{Ld}\right), \tag{2.6}$$

where $d$ is the particle diameter, $\theta_1$ corresponds to the angle where $h_{stop}(\theta)$ diverges, $\theta_2$ to the angle where $h_{stop}(\theta)$ vanishes and $L$ is a characteristic dimensionless thickness over which $\theta_{stop}(h)$ varies.

By substituting $h_{stop}(\theta)$ from Equation 2.4 into Equation 2.6, we obtain the following relation:

$$\mu(u,h) = \tan\theta = \tan\theta_1 + (\tan\theta_2 - \tan\theta_1)\exp\left(\frac{-\beta h}{Ld}\frac{\sqrt{gh}}{u}\right). \tag{2.7}$$

This empirical relation raises several issues. On the one hand, in the high-velocity regime, Equation 2.7 predicts that the friction coefficient tends to a limit equal to $\tan\theta_2$ when the velocity tends to infinity. The existence of an upper limit for the friction coefficient implies that no steady uniform flow can be obtained for an inclination higher than $\theta_2$. On the other hand, the low-velocity regime is not well described by Equation 2.7. According to this empirical relation, steady uniform flows can be observed for any thickness

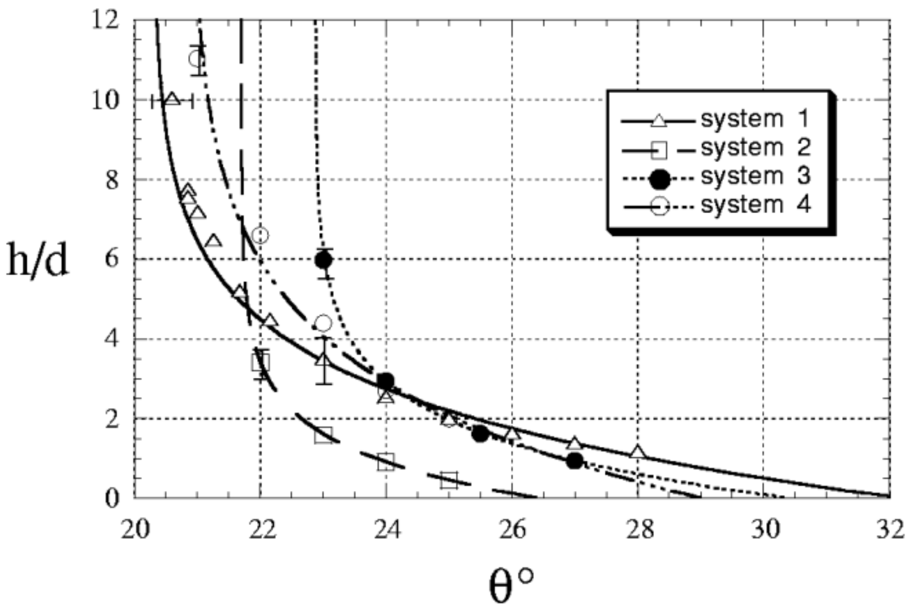


Figure 2.3: $h_{stop}$ as a function of $\theta$ [20].

as soon as the inclination is higher than $\theta_1$. This is in contradiction with the existence of the function $h_{stop}(\theta)$. However, it must be kept in mind that Equation 2.7 is derived from Equation 2.4, which is valid only for $h > h_{stop}$ or equivalently for Froude number $Fr > \beta$. Hence, Equation 2.7 is valid only when the Froude number exceeds $\beta$. For $Fr < \beta$, we have no information about the friction law. Experiments on friction between solids [14] or between a rough plate and a granular layer [18] in the low-velocity regime have shown a complex and rich dynamics which is not yet fully understood.
Hence, it can be inferred from the experiments that steady uniform flows can only develop when the inclination angle $\theta \in (\theta_1, \theta_2)$.

Pouliquen and Forterre [21] presented the following expression for determining the friction coefficient $\mu$:
If $Fr > \beta$:

$$\mu(h, Fr) = \mu_{stop}\left(h\frac{\beta}{Fr}\right), \tag{2.8}$$

if $0 < Fr \leq \beta$:

$$\mu(h, Fr) = \left(\frac{Fr}{\beta}\right)^{\kappa} (\mu_{stop}(h) - \mu_{start}(h)) + \mu_{start}(h), \tag{2.9}$$

if $Fr = 0$:

$$\mu(h, 0) = \min(\mu_{start}(h), \|\tan\theta e_x - \nabla h\|_2). \tag{2.10}$$

Equation 2.9 provides the value of $\mu$ when $Fr \in (0, \beta)$. Equation 2.9 is derived by extrapolating by a power function characterised by a power of $\kappa$. Equation 2.10 is to ensure that, when the material is static, the friction balances the other forces exactly unless the total force reaches the threshold value given by the static friction coefficient [21]. The friction coefficient is qualitatively determined by the two functions $\mu_{start}(h) = \tan(\theta_{start}(h))$ and $\mu_{stop}(h) = \tan(\theta_{stop}(h))$, which are experimentally obtained as best fits of curves similar to the ones in Figure 2.3. $\theta_1$ and $\theta_2$ are obtained from the graph between $h_{stop}/d$ and $\theta$ which is exactly the curve shown in Figure 2.3, and $\theta_3$ is the angle at which the curve between $h_{start}/d$ and $\theta$ diverges. For the glass beads used in their experiment by Pouliquen and Forterre [21], the best fits are given by the following expression:

$$\mu_{stop}(h) = \tan\theta_1 + (\tan\theta_2 - \tan\theta_1)\frac{1}{h/L+1}, \tag{2.11}$$

$$\mu_{start}(h) = \tan\theta_3 + (\tan\theta_2 - \tan\theta_1)\frac{1}{h/L+1}, \tag{2.12}$$

with $\theta_1 = 21°$, $\theta_2 = 30.5°$, $\theta_3 = 22.2°$ and $L = 0.65\ mm$.
Equation 2.10 presents the coefficient of static friction. In general, the coefficient of static friction may lie below $\mu_{start}$ while the static friction force simply balances the remaining forces to prevent motion [7]. A general two-dimensional static momentum balance implies that the friction must precisely match the gravitational force pulling the grains downslope as well as any pressure gradients that may be driving them in other directions [6]. As a result, $\mu = \|\tan\theta e_x - \nabla h\|_2$ where $e_x$ is a unit vector in the downslope direction and $\nabla h$ is the gradient of the thickness.

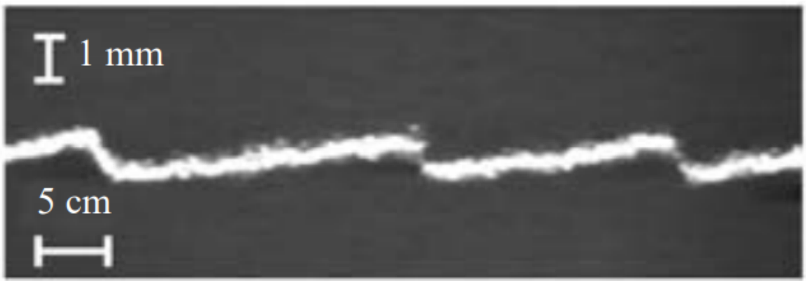


Figure 2.4: Thickness profile observed during flow of sand down an inclined plane [10].

While experimentally studying the development of long-surface-wave instability in granular material flows, Forterre and Pouliquen [10] made some interesting observations. They conducted experiments similar to the one in [20] but used sand as the granular material instead of glass beads. They made a very striking observation that sand flows such that the thickness profile of the flow looks like a wave, as seen in Figure 2.4, whereas no such waves are visible with glass beads. Hence, they realised sand and glass beads behave differently while flowing down an inclined plane. They further examined and compared the two flows experimentally, which led them to include an offset parameter $\gamma$ in Equation 2.4. They presented a more generalised version of the velocity law as seen in Equation 2.13.

$$\frac{u}{\sqrt{gh}} = -\gamma + \beta \frac{h}{h_{stop}(\theta)}, \tag{2.13}$$

with $\gamma = 0$, $\beta = 0.136$ for glass beads, and $\gamma = 0.77$, $\beta = 0.65$ for sand.

The empirical friction law (Equation 2.4) is valid only for flows in the dynamic regime where steady uniform flows exist. According to Pouliquen and Forterre [21], a flow is in the dynamic regime if $h \geq h_{stop}(\theta)$ which is equivalent to $Fr \geq \beta$ in case of glass beads, since the Froude number offset is $\gamma = 0$. Edwards and Gray [8] found that allowing a dynamic regime for $h \geq h_{stop}(\theta)$ in the same way for glass beads, with $\beta = 0.65$ and $\gamma = 0$, leads to the formation of erosion-deposition waves that deposit grains in the wave troughs with a layer thickness $h_{deposit}$ that is significantly lower than $h_{stop}(\theta)$. From this, Edwards *et al.* [7, 6], while studying the phenomenon of erosion and hysteresis in granular flows, implied that for a steady uniform flow to leave a layer of thickness $h_{stop}$, the transition between dynamic and intermediate friction regimes must occur at a higher Froude number. Furthermore, for materials such as sand, where $\beta < \gamma$, the minimum Froude number for the transition to the intermediate regime, i.e., $Fr = \beta - \gamma$ is negative, implying that all flows can only be in the dynamic regime. To overcome these problems Edwards *et al.* [7] introduced a friction law transition point $h_*(\theta)$ such that $h_*(\theta) \in (h_{stop}(\theta), h_{start}(\theta))$, and is given by

$$h_*(\theta) = (1-a)h_{stop}(\theta) + ah_{start}, \tag{2.14}$$

where $a \in [0,1]$, and $h_{stop}(\theta)$ and $h_{start}(\theta)$ can be derived by inverting the Equations 2.11-2.12 respectively, and can be written as

$$h_{start}(\theta) = L\left(\frac{\tan\theta_2 - \tan\theta_1}{\tan\theta - \tan\theta_3} - 1\right), \tag{2.15}$$

$$h_{stop}(\theta) = L\left(\frac{\tan\theta_2 - \tan\theta_1}{\tan\theta - \tan\theta_2} - 1\right). \tag{2.16}$$

Edwards *et al.* [7] also introduced the parameter $\beta_*$ and defined it as the Froude number at $h = h_*$. Substituting Equation 2.14 in Equation 2.4, we obtain an expression for $\beta_*$ given as

$$\beta_*(\theta) = \beta\left(1 - a + a\frac{h_{start}(\theta)}{h_{stop}(\theta)}\right) - \gamma. \tag{2.17}$$

The maximum value of $\beta_*$ is at $a = 0$, i.e., when $h_*(\theta) = h_{start}$ and upon substituting $h_{start}(\theta)$ and $h_{stop}(\theta)$ from Equations 2.15-2.16 in Equation 2.17, we obtain

$$\beta_* \leq \beta_*^{max} = \beta\left(\frac{\mu_2 + \mu_3 - 2\mu_1}{\mu_2 - \mu_3}\right)^2 - \gamma. \tag{2.18}$$

In the experiments conducted by Edwards *et al.* [7] using carborundum particles as granular material flowing on a rough plane of glass beads, they found that the value of $\beta_*^{max}$ calculated using the material properties matched closely to the one chosen by them to match the deposit layer depths $h_{deposit} \approx h_{stop}$. However, for other materials and slope roughnesses, $\beta_*$ may not be equal to $\beta_*^{max}$ and should instead be inferred directly from the flow rule experiments [6]. As such, $\beta_*$ may be considered as another intrinsic rheological property. The final expression for the modified friction law given by Edwards *et al.* [7] to determine the coefficient of basal friction $\mu$ can be summarised as follows:
If $Fr > \beta_*$:

$$\mu(h, Fr) = \mu_{stop}\left(h\frac{\beta}{Fr+\gamma}\right), \tag{2.19}$$

if $0 < Fr \leq \beta_*$:

$$\mu(h, Fr) = \mu_{start}(h) + \left(\frac{Fr}{\beta_*}\right)^{\kappa}(\mu_{stop}(h) - \mu_{start}(h)), \tag{2.20}$$

if $Fr = 0$:

$$\mu(h, 0) = \min(\mu_{start}(h), \|\tan\theta e_x - K\nabla h\|_2). \tag{2.21}$$

## 2.3 Depth-Averaged Avalanche Equations

Consideration of a solid as a fully rigid body serves as a good approximation for modelling the properties of a solid. However, in practice, totally rigid bodies typically crack or shatter when subjected to high stresses. As a result, when a more realistic solid is subjected to intermittent shearing stress, the body will somewhat deform before returning to its former shape, indicating the presence of some elastic property. When describing a fluid, however, this description is altered because a fluid has no stiffness at all. Small fluid components' inability to resist any inclination by applied forces to deform them without changing shape is a fundamental property of fluids. This notion has implications for describing a granular avalanche, which can behave like a solid or fluid and switch between the two states within the same flow [1]. Avalanches are often modelled using the governing equations of fluid flow because these equations provide a simplified yet effective way to describe the flow of granular materials like snow and debris down a steep slope. The mathematical model used to describe granular flow during an avalanche for this thesis is derived using the conservation of mass (Equation 2.22) and momentum (Equation 2.23) for an incompressible fluid. The conservation equations can be written as follows:

$$\nabla\cdot\mathbf{u} = 0, \tag{2.22}$$

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t} + \nabla\cdot(\mathbf{u}\otimes\mathbf{u})\right) = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{g}, \tag{2.23}$$

where:

- t is time.
- $\nabla$ is the gradient operator.
- $u$ is the velocity vector.
- $\sigma$ is the stress tensor which is symmetric, $\sigma^T = \sigma$.
- $g$ denotes the acceleration due to gravitational body force.

The typical length scales within the plane of propagation of geophysical flows, such as the separation between the start and runout of the flow and its width, are more significant than the typical length scales in the outward normal direction, such as the depth of the avalanche. As a result, the aspect ratio is often relatively low relative to the vertical in both the cross-slope and downslope directions. This property is helpful mathematically because it allows for the use of the small spatial parameter that makes it possible to employ asymptotic analysis to make the governing equations simpler while still achieving good approximations. Averaging over the shortest dimension allows for additional simplifications and frequently reduces the problem's effective spatial dimension by one. This approach is exactly similar to what was done by Saint-Venant [5] while deriving the shallow-water equations (Equations 2.1-2.3). Equations 2.22-2.23 were integrated along the direction of depth to obtain the shallow-water equations.

The governing equations used for the numerical simulation in this thesis are based on the equations given in [3] combined with the friction law presented in Equations 2.19-2.21. These equations are also based on the shallow-water assumption. It is useful to describe the derivation of the final equations. The following is an overview of the derivation of the depth-averaged avalanche equations given in [3].

Consider a mass of granular material flowing down an inclined plane which is inclined at an angle $\theta$ to the horizontal as shown in Figure 2.5. Let $Oxyz$ be a Cartesian coordinate system with the $x$-axis aligned with the downslope direction, the $y$-axis aligned with the cross-slope direction and the $z$-axis normal to the surface of the plane. The velocity $\mathbf{u}$ has components $(u, v, w)$ in the $(x, y, z)$ directions, respectively. Assuming a rigid flat base, the free surface lies at $z = h$, where $h$ is the depth of the flow. The depth-averaged velocity, $\bar{\boldsymbol{u}} = (\bar{u}, \bar{v})$, is defined as

$$\bar{u} = \frac{1}{h} \int_0^h u \, dz \; , \tag{2.24}$$

$$\bar{v} = \frac{1}{h} \int_0^h v \, dz \; . \tag{2.25}$$

Upon depth integration of Equations 2.22-2.23, we obtain

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\bar{\boldsymbol{u}}) = 0, \tag{2.26}$$

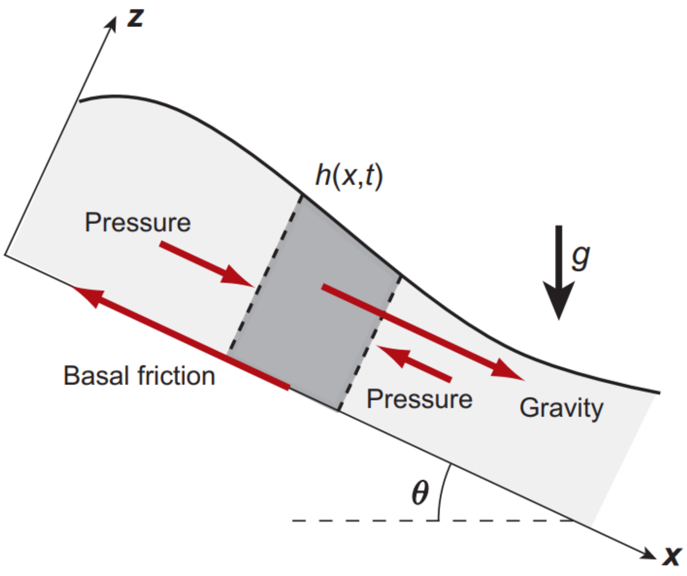


Figure 2.5: Force balance in the shallow-water description [11].

$$\frac{\partial(h\bar{\boldsymbol{u}})}{\partial t} + \nabla \cdot (\chi h\bar{\boldsymbol{u}} \otimes \bar{\boldsymbol{u}}) + \nabla\left(\frac{1}{2}gh^2\cos\theta\right) = gh\mathbf{S} + \frac{1}{\rho}\nabla\cdot(h\bar{\boldsymbol{\tau}}), \tag{2.27}$$

where $\bar{\boldsymbol{\tau}}$ is the depth-averaged deviatoric stress. The shape factor $\chi$ accounts for the difference between the square of the depth-averaged velocity and the depth-averaged square of velocities via the relation $\bar{\boldsymbol{u^2}} = \chi\bar{\boldsymbol{u}}^{\mathbf{2}}$. The value of $\chi$ depends on the vertical velocity profile, with $\chi = 5/4$ for a Bagnold profile and $\chi = 1$ if the profile is independent of $z$. We assume for our avalanche model that $\chi = 1$ like what was done by Pouliquen and Forterre [21]. The source term $\mathbf{S} = (S_x, S_y)$ occurs due to gravity and effective basal friction, where

$$S_x = \cos\theta\left(\tan\theta - \mu\frac{\bar{u}}{|\bar{\boldsymbol{u}}|}\right), \tag{2.28}$$

$$S_y = \cos\theta\left(-\mu\frac{\bar{v}}{|\bar{\boldsymbol{u}}|}\right), \tag{2.29}$$

where $\mu$ is based on the basal friction law given in Equations 2.19-2.21. The final expression on the right-hand side of the momentum equation 2.27 is the divergence of the deviatoric stress tensor. This term is usually very small and is neglected in standard inviscid shallow-water-like avalanche models. However, in more subtle problems, it plays a critical role in obtaining a correct solution. The depth-averaged deviatoric stress tensor $\bar{\boldsymbol{\tau}}$ is defined as

$$\bar{\boldsymbol{\tau}} = \frac{1}{h}\int_0^h \tau\, dz\,, \tag{2.30}$$

where $\tau$ is the deviatoric component of the (three-dimensional) Cauchy stress tensor

$$\sigma = -p\mathbf{I}_3 + \tau, \tag{2.31}$$

where $\mathbf{I}_3$ is the $3 \times 3$ identity matrix and $p$ is the hydrostatic pressure. The aim of this decomposition is to write $\tau$ in terms of the state variables already present in the formulation, i.e., $(\boldsymbol{u}, p)$, and it is through this term that constitutive information is included in the model. When the shear stress gives the deviatoric stress, the shear stress and strain rate are co-linear, i.e.,

$$\frac{\tau}{\|\tau\|} = \frac{\mathbf{D}}{\|\mathbf{D}\|}, \tag{2.32}$$

where $\mathbf{D}$ is the strain-rate tensor. The strain-rate tensor $\mathbf{D}$ is defined to using the velocity gradient $\mathbf{L} = \nabla \mathbf{u}$ as

$$\mathbf{D} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^T\right). \tag{2.33}$$

The norm on the strain-rate tensor is given by its second invariant defined as

$$\|\mathbf{D}\| = \sqrt{\frac{1}{2} tr(\mathbf{D}^2)}, \tag{2.34}$$

where $tr$ is the trace operation.
Combining the general concepts of plasticity with Coulomb's postulate for friction in granular media, we can write

$$\|\tau\| = \mu_i p, \tag{2.35}$$

where $\mu_i$ is the coefficient of internal friction. Using this formulation, $\tau$ can be written as

$$\tau = \nu(p, \|\mathbf{D}\|)\mathbf{D}, \tag{2.36}$$

where $\nu$ can be interpreted as the effective non-Newtonian granular viscosity. We can write using Equation 2.32 that

$$\tau = \mu_i(I) p \frac{\mathbf{D}}{\|\mathbf{D}\|}, \tag{2.37}$$

where $I$ is the inertial number. The Groupement de Recherche Milieux Divisés [12] gathered from numerical studies that the coefficient of internal friction $\mu_i$ is not a constant but rather a function depending on a single non-dimensional parameter $I$. $I$ is known as the inertial number and can be interpreted as the ratio between the time scale given by the shear rate and the time scale related to the confining pressure [12]. It is written in the following form:

$$I = \frac{2\|\mathbf{D}\| d}{\sqrt{p/\rho^*}}, \tag{2.38}$$

where $d$ is the particle diameter, and $\rho^*$ is the intrinsic solid density of the grains. Jop *et al.* [15] derived the function $\mu_i(I)$ from the basal friction law as

$$\mu_i(I) = \mu_1 + \frac{\mu_2 - \mu_1}{I/I_0 + 1}, \tag{2.39}$$

where the constant $I_0$ is given by

$$\frac{5\beta d}{2L\sqrt{\phi}}, \tag{2.40}$$

where $\phi$ is the solid volume fraction. The hydrostatic pressure is given as

$$p = \rho g(h - z)\cos(\theta), \tag{2.41}$$

where $\rho = \rho^*\phi$. The downslope momentum balance can also be integrated to show that the shear stress is

$$\tau_{xz} = \rho g(h - z)\sin\theta. \tag{2.42}$$

Substituting Equations 2.41 and 2.42 in Equation 2.35 implies that $\mu_i(I) = \tan\theta$. Hence for a fixed slope angle $\theta$, the inertial number $I$ is equal to a constant $I_\theta$, and is given as

$$I_\theta = I_0\left(\frac{\tan\theta - \mu_1}{\mu_2 - \tan\theta}\right). \tag{2.43}$$

The Bagnold velocity profile is given as

$$u = \frac{2I_\theta}{3d}\sqrt{\phi g\cos\theta}(h^{3/2} - (h - z)^{3/2}). \tag{2.44}$$

Integrating the Bagnold velocity in Equation 2.44 through the avalanche depth implies that the depth-averaged Bagnold velocity is

$$\bar{u} = \frac{I_\theta}{5d}\sqrt{\phi g\cos\theta}h^{3/2}. \tag{2.45}$$

The depth-averaged viscosity term in one spatial dimension was derived by Gray and Edwards in [13]. It is useful to briefly summarise their derivation only to later generalise in order to extend it to a two-dimensional domain. Equation 2.35 implies that the downslope in-plane deviatoric stress is

$$\tau_{xx} = \mu(I)p\frac{D_{xx}}{\|\mathbf{D}\|}. \tag{2.46}$$

Using Equation 2.33, the strain rate $D_{xx}$ is calculated by differentiating the Bagnold velocity profile in Equation 2.44 with respect to $x$

$$D_{xx} = \frac{\partial u}{\partial x} = \frac{I_\theta}{2d}\sqrt{\phi g\cos\theta}(h^{1/2} - (h - z)^{1/2})\frac{\partial h}{\partial x}. \tag{2.47}$$

Similarly, to leading order, the second invariant of the strain-rate tensor could also be evaluated using the Bagnold velocity profile, i.e.,

$$\|\mathbf{D}\| = \frac{1}{2}\left|\frac{\partial u}{\partial z}\right| = \frac{I_\theta}{2d}\sqrt{\phi g\cos\theta}(h - z)^{1/2}. \tag{2.48}$$

By substituting $\mu_i = \tan\theta$, and Equations 2.41, 2.47 and 2.48 in Equation 2.46 we obtain

$$\tau_{xx} = 2\rho g \sin\theta (h^{1/2}(h-z)^{1/2} - (h-z))\frac{\partial h}{\partial x}. \tag{2.49}$$

Integrating Equation 2.49 through the avalanche depth, the leading-order approximation for the depth-averaged deviatoric in plane stress $\bar{\tau}_{xx}$ is

$$h\bar{\tau}_{xx} = \frac{1}{3}\rho g \sin\theta h^2 \frac{\partial h}{\partial x}. \tag{2.50}$$

From Equation 2.45, it follows that the thickness gradient is

$$\frac{\partial h}{\partial x} = \frac{5d}{3I_\theta\sqrt{\phi g\cos\theta}}\frac{1}{h^{1/2}}\frac{\partial \bar{u}}{\partial x}. \tag{2.51}$$

Using Equation 2.51 in Equation 2.50, we obtain

$$h\bar{\tau}_{xx} = \rho\nu h^{3/2}\frac{\partial \bar{u}}{\partial x}, \tag{2.52}$$

where all constant parameters have been merged into $\nu$ to make it equivalent to the depth-averaged viscosity, which can be expressed as

$$\nu = \frac{2L\sqrt{g}}{9\beta}\frac{\sin\theta}{\sqrt{\cos\theta}}\left(\frac{\mu_2 - \tan\theta}{\tan\theta - \mu_1}\right). \tag{2.53}$$

Baker *et al.* [3] showed that Equation 2.52 can be generalised for two dimensions as

$$h\bar{\boldsymbol{\tau}} = \rho\nu h^{3/2}\bar{\boldsymbol{D}}, \tag{2.54}$$

where

$$\bar{\boldsymbol{D}} = \frac{1}{2}(\bar{\boldsymbol{L}} + \bar{\boldsymbol{L}}^T), \tag{2.55}$$

is the depth-integrated strain-rate tensor and $\bar{\boldsymbol{L}} = \nabla\bar{\boldsymbol{u}}$ is the two-dimensional gradient of the depth-averaged velocity.
By substituting the depth-averaged deviatoric stress tensor from Equation 2.54 in Equation 2.27, we finally obtain the depth-averaged avalanche equations in component form as

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(h\bar{u}) + \frac{\partial}{\partial y}(h\bar{v}) = 0, \tag{2.56}$$

$$\frac{\partial}{\partial t}(h\bar{u}) + \frac{\partial}{\partial x}\left(h\bar{u}^2 + \frac{1}{2}gh^2\cos\theta\right) + \frac{\partial}{\partial y}(h\bar{u}\bar{v}) = ghS_x + \frac{\partial(D_{1,x})}{\partial x} + \frac{\partial(D_{1,y})}{\partial y}, \tag{2.57}$$

$$\frac{\partial}{\partial t}(h\bar{v}) + \frac{\partial}{\partial x}(h\bar{u}\bar{v}) + \frac{\partial}{\partial y}\left(h\bar{v}^2 + \frac{1}{2}gh^2\cos\theta\right) = ghS_y + \frac{\partial(D_{2,x})}{\partial x} + \frac{\partial(D_{2,y})}{\partial y}, \tag{2.58}$$

with $S_x$ and $S_y$ given in Equations 2.28-2.29 and

$$D_{1,x} = \left(\nu h^{3/2}\frac{\partial \bar{u}}{\partial x}\right),$$

$$D_{1,y} = D_{2,x} = \left(\frac{1}{2}\nu h^{3/2}\left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right)\right),$$

$$D_{2,y} = \left(\nu h^{3/2}\frac{\partial \bar{v}}{\partial y}\right).$$

For this thesis, Equations 2.56-2.58 are used in ExaHyPE 2 for avalanche simulations.

# 3 ExaHyPE 2

ExaHyPE 2 (”An Exascale Hyperbolic PDE Engine”) is a software engine for solving systems of first-order hyperbolic partial differential equations (PDEs) [23]. Hyperbolic PDEs are typically derived from the conservation laws of physics and are useful in a wide range of application areas. Applications powered by ExaHyPE 2 can be run on a student's laptop, as well as exploit thousands of processor cores on state-of-the-art supercomputers. The engine can dynamically increase the accuracy of the simulation using adaptive mesh refinement where required. Due to the robustness and shock-capturing abilities of ExaHyPE 2's numerical methods, engine users can simulate linear and non-linear hyperbolic PDEs with very high accuracy. Users can tailor the engine to their particular PDE by specifying evolved quantities, fluxes, and source terms. A complete simulation code for a new hyperbolic PDE can often be realised within a few hours — a task that, traditionally, can take weeks, months and often years for researchers starting from scratch.

The ExaHyPE 2 engine relies on the Peano framework [26] for its dynamically adaptive Cartesian meshes. In addition, it provides an efficient mesh traversal scheme that ExaHyPE 2's algorithms plug into Peano itself. Peano decomposes the Cartesian domain into patches based on the inputs and the choice of numerical method provided by the user. These patches are then traversed in an order defined by the Peano curve. Figure 3.1 shows a schematic of a single patch with interior cells (green), halo cells (white) and corner halo cells (red). The data in corner halo cells (red) is not communicated to the neighbouring patches. Only the data in halo cells with emanating arrows (white) is communicated to neighbouring patches.

Since ExaHyPE 2 is a solver engine, domain-specific code has to be written by the user to obtain the simulation code. To write an ExaHyPE 2 application, users typically start with a Python script written using the Python API of ExaHyPE 2 and Peano. This Python script yields a native Peano application (builder mechanism), which then assembles the application. The user fills the application-specific classes with the PDE terms. The generated glue code and the initially empty templates comprise the ExaHyPE 2 application.

## 3.1 Problem Formulation

ExaHyPE 2 itself is a hyperbolic PDE engine, and its goal is to provide a generic API in which users can implement the specifics of their hyperbolic PDEs and have the engine

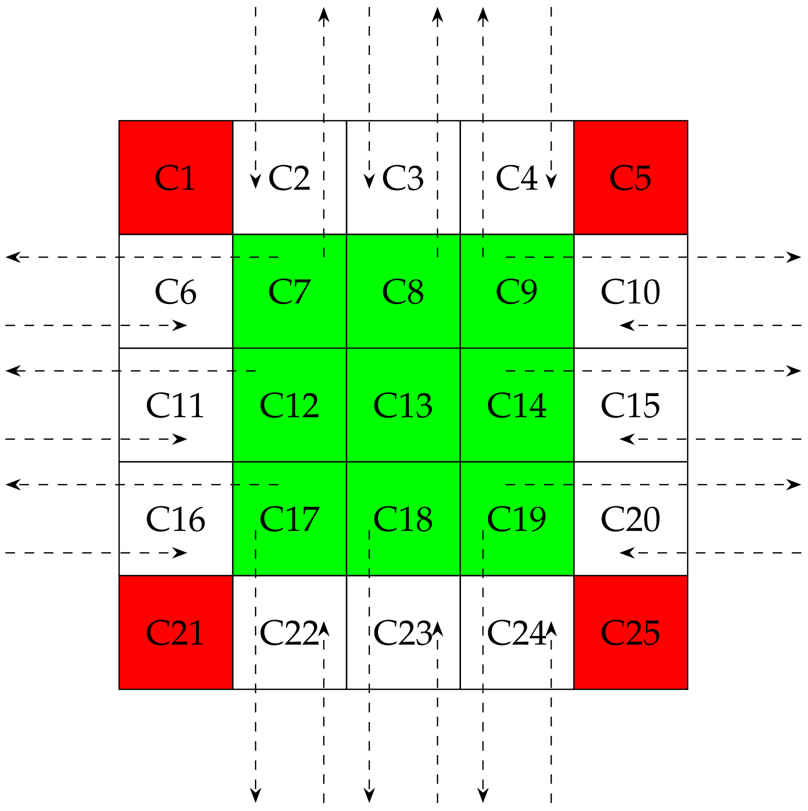


Figure 3.1: Single patch for a patch-size of 3 with internal cells (green), halo cells (white) and corner halo cells (red).

solve these. ExaHyPE 2 solves partial differential equations of the following form:

$$\frac{\partial Q}{\partial t} + \nabla \cdot F(Q, \nabla Q) + B(Q).\nabla Q = S(Q) + \sum_{i=1}^{n_{ps}} \delta_i, \tag{3.1}$$

where Q is the vector of unknowns and auxiliary variables, F is the *flux*, B is the *non-conservative product* (often shortened to NCP), and S is the *source term* of the equation.
For simplicity, the easiest way to think of each of these is as follows:

- The *flux* contains terms linked to the transmission of a quantity within the domain, such as the flow of heat from a hot part of the domain to a colder one. It concerns quantities that do not change but move from one part to another.
- The *non-conservative product* refers to the non-linear terms that appear in the hyperbolic PDEs. These terms are called non-conservative because they do not have a clear conservation interpretation, meaning they do not preserve any physical quantities in the system.
- The *source term* contains terms that are acted upon the system from outside, such as gravity or some other forces.

## 3.2 The Shallow-Water Equations in ExaHyPE 2

This section is a brief tutorial on how to implement the shallow-water equations (SWE) (Equations 2.1-2.3) in ExaHyPE 2. The algorithms required to develop the solver application are presented.
On comparing the shallow-water equations (Equations 2.1-2.3) with Equation 3.1, we can write:

$$Q = \begin{pmatrix} h \\ h\bar{u} \\ h\bar{v} \\ b \end{pmatrix}, \tag{3.2}$$

$$F = \begin{pmatrix} h\bar{u} & h\bar{v} \\ h\bar{u}^2 + \frac{1}{2}gh^2 & h\bar{u}\bar{v} \\ h\bar{u}\bar{v} & h\bar{v}^2 + \frac{1}{2}gh^2 \end{pmatrix}, \tag{3.3}$$

$$B\nabla Q = \begin{pmatrix} 0 & 0 \\ gh\frac{\partial b}{\partial x} & 0 \\ 0 & gh\frac{\partial b}{\partial y} \end{pmatrix}, \tag{3.4}$$

$$S = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.5}$$

Algorithms 1-3 are implemented for a dam-break scenario with reflective boundary conditions. This scenario is chosen as an example to demonstrate the implementation of these algorithms.

The initial conditions are set such that the water height inside a circular dam is $0.5\ m$ higher than the height outside the dam.

**Algorithm 1:** Implementation of the initial conditions function for an SWE dam-break scenario

```
Input: $Q$: Vector of unknown variables containing $h, hu, hv, b$.
       $volumeCentre$: Vector containing coordinates of the centre of the current
       volume.
Output: $Q$: Vector with initialised $h, hu, hv, b$.
Data: $dam$: Region convered by the dam in the domain.
1 Function initialConditions($Q$, $volumeCentre$):
2     if $volumeCentre \in dam$ then
3         $Q[h] \leftarrow 1.5$             // water height inside dam is 1.5 m
4     else
5         $Q[h] \leftarrow 1.0$         // water height outside the dam is 1.0 m
6     $Q[hu] \leftarrow 0.0$     // initial velocity in $x$-direction is 0.0 m/s
7     $Q[hv] \leftarrow 0.0$     // initial velocity in $y$-direction is 0.0 m/s
8     $Q[b] \leftarrow 0.0$                               // assuming flat bottom
9     return $Q$
```

Reflective boundary conditions are programmed in Algorithm 2 such that the water height at the boundary is equal to the water height in the cell adjacent to the boundary inside the domain. Velocity at the boundary is equal and opposite to the velocity in the cell adjacent to the boundary inside the domain.

The user can initially choose which part of the domain they want to refine. This feature is called static adaptive mesh refinement. For e.g., in Algorithm 3, the cells inside the dam region are refined, and volumes outside the dam are kept unrefined.

The eigenvalues of the system are the eigenvalues of the Jacobian of the *flux* function. Since the 2D shallow-water equations are a system of three PDEs, we get three eigenvalues in each of the two components (dimensions). The eigenvalues in $x$-direction are the eigenvalues of the Jacobian of the $x$-component of the *flux* in Equation 3.3, which is $(h\bar{u}, h\bar{u}^2 + \frac{1}{2}gh^2, h\bar{u}\bar{v})^T$. Similarly, the eigenvalues in $y$-direction are the eigenvalues of the Jacobian of the $y$-component of the *flux* which is $(h\bar{v}, h\bar{u}\bar{v}, h\bar{v}^2 + \frac{1}{2}gh^2)^T$. Upon calculation, the eigenvalues in $x$- and $y$-direction turn out to be $\lambda_{x_1} = \bar{u}, \lambda_{x_2} = \bar{u} - \sqrt{gh}, \lambda_{x_3} = \bar{u} + \sqrt{gh}$ and $\lambda_{y_1} = \bar{v}, \lambda_{y_2} = \bar{v} - \sqrt{gh}, \lambda_{y_3} = \bar{v} + \sqrt{gh}$, respectively. The `maxEigenvalue` function in Algorithm 4 returns the maximum eigenvalue in the normal-direction.

From Equation 3.3, we can see that the $x$-component of the *flux* is $(hu, hu^2 + \frac{1}{2}gh^2, huv)$

**Algorithm 2:** Implementation of the reflective boundary conditions function for SWE

```
Input: $Qinside$: Vector of unknown variables containing $h, hu, hv, b$ of volume
       located just inside the domain.
       $Qoutside$: Vector containing unknown variables $h, hu, hv, b$ of the volume
       at the boundary.
Output: $Qoutside$: Vector containing unknown variables with assigned boundary
        conditions.
Function boundaryConditions($Qinside, Qoutside$):
    $Qoutside[h] \leftarrow Qinside[h]$
    $Qoutside[hu] \leftarrow -Qinside[hu]$      // refective boundary condition
    $Qoutside[hv] \leftarrow -Qinside[hv]$      // refective boundary condition
    $Qoutside[b] \leftarrow Qinside[b]$
    return $Qoutside$
```

and the $y$-component is $(hv, huv, hv^2 + \frac{1}{2}gh^2)$. One of the two components is returned by Algorithm 5 based on the value of $n$.

From Equation 3.4, we can see that the $x$-component of the *non-conservative product* is $(0, gh\frac{\partial b}{\partial x}, 0)^T$ and the $y$-component is $(0, 0, gh\frac{\partial b}{\partial y})^T$. One of the two components is returned by Algorithm 6 based on the value of $n$.

Since the *source term* is $(0, 0, 0)^T$, we need not provide its implementation to the engine. The engine assumes it to be zero by default.

The details of the simulation, such as choosing the solver, volume size, time-step size and domain size are provided in the Python script used to configure the solver and to build the code. The Python script is written using the Python API of ExaHyPE 2.

**Algorithm 3:** Implementation of the refinement criterion function for SWE

```
Input: $Q$: Vector of unknown variables containing $h, hu, hv, b$.
       $volumeCentre$: Vector containing coordinates of the centre of current
       volume with $\{0 : x, 1 : y\}$.
Output: $refinementCommand \in \{Keep, Refine, Erase\}$: Refinement option.
Data: $dam$: Region convered by the dam in the domain.
Function refinementCriterion($Q, volumeCentre$):
    if $volumeCentre \in dam$ then
        return $Refine$
    else
        return $Keep$
```

**Algorithm 4:** Implementation of the function calculating max eigenvalue for SWE

```
Input: $Q$: Vector of unknown variables containing $h, hu, hv, b$.
       $n \in \{x, y\}$: Normal-direction.
Output: $maxEigenvalue$: Maximum eigenvalue in the direction pointed by $n$.
Function maxEigenvalue($Q$, $n$):
    $g \leftarrow 9.81$
    $c \leftarrow \sqrt{gQ[h]}$
    if $n$ is along $x$ then
        $u \leftarrow Q[hu]/Q[h]$
        $maxEigenvalue \leftarrow \max(|u-c|, |u+c|)$
    else if $n$ is along $y$ then
        $v \leftarrow Q[hv]/Q[h]$
        $maxEigenvalue \leftarrow \max(|v-c|, |v+c|)$
    return $maxEigenvalue$
```

**Algorithm 5:** Implementation of the *flux* function for SWE

```
Input: $Q$: Vector of unknown variables containing $h, hu, hv, b$.
       $n \in \{x, y\}$: Normal-direction.
Output: $F$: Vector containing flux in the normal-direction.
Function flux($Q$, $n$):
    $g \leftarrow 9.81$ if $n$ is along $x$ then
        $F[0] \leftarrow Q[hu]$                                                  // $hu$
        $F[1] \leftarrow Q[h]\left(\frac{Q[hu]}{Q[h]}\right)^2 + \frac{1}{2}g(Q[h])^2$   // $hu^2 + \frac{1}{2}gh^2$
        $F[2] \leftarrow Q[hu]\left(\frac{Q[hv]}{Q[h]}\right)$                    // $huv$
    else if $n$ is along $y$ then
        $F[0] \leftarrow Q[hv]$                                                  // $hv$
        $F[1] \leftarrow Q[hu]\left(\frac{Q[hv]}{Q[h]}\right)$                    // $huv$
        $F[2] \leftarrow Q[h]\left(\frac{Q[hv]}{Q[h]}\right)^2 + \frac{1}{2}g(Q[h])^2$   // $hv^2 + \frac{1}{2}gh^2$
    return $F$
```

**Algorithm 6:** Implementation of the *non-conservative product* function for SWE

```
Input: $Q$: Vector of unknown variables containing $h, hu, hv, b$.
       $n \in \{x, y\}$: Normal-direction.
       $deltaQ$: Vector containing derivatives $h_n, hu_n, hv_n, b_n$ along the
       normal-direction $n$.
Output: $BTimesDeltaQ$: Vector containing the non-conservative term in the
        normal-direction.
1  Function nonConservativeProduct($Q$, $n$, $\nabla Q$):
2      $g \leftarrow 9.81$
3      if $n$ is along $x$ then
4          $BTimesDeltaQ[0] \leftarrow 0$
5          $BTimesDeltaQ[1] \leftarrow gQ[h](deltaQ[b])$          // $ghb_x$
6          $BTimesDeltaQ[2] \leftarrow 0$
7      else if $n$ is along $y$ then
8          $BTimesDeltaQ[0] \leftarrow 0$
9          $BTimesDeltaQ[1] \leftarrow 0$
10         $BTimesDeltaQ[2] \leftarrow gQ[h](deltaQ[b])$          // $ghb_y$
11     return $BTimesDeltaQ$
```

# 4 Implementation of the Depth-Averaged Avalanche Equations in ExaHyPE 2

The depth-averaged avalanche equations (Equations 2.56-2.58) were implemented in ExaHyPE 2. This chapter presents all the algorithms required to develop a solver application that uses the finite volume method with Rusanov flux and adaptive time stepping to simulate avalanches using the depth-averaged avalanche equations (DAE).

The implementation of hyperbolic PDEs in ExaHyPE 2 requires them to be arranged as per Equation 3.1. On comparing Equations 2.56-2.58 with Equation 3.1, we can write

$$Q = \begin{pmatrix} h \\ h\bar{u} \\ h\bar{v} \end{pmatrix}, \tag{4.1}$$

$$F = \begin{pmatrix} h\bar{u} & h\bar{v} \\ h\bar{u}^2 + \frac{1}{2}g\cos\theta h^2 & h\bar{u}\bar{v} \\ h\bar{u}\bar{v} & h\bar{v}^2 + \frac{1}{2}g\cos\theta h^2 \end{pmatrix}, \tag{4.2}$$

$$B\nabla Q = \begin{pmatrix} 0 & 0 \\ -\frac{\partial(D_{1,x})}{\partial x} & -\frac{\partial(D_{1,y})}{\partial y} \\ -\frac{\partial(D_{2,x})}{\partial x} & -\frac{\partial(D_{2,y})}{\partial y} \end{pmatrix}, \tag{4.3}$$

where

$$\begin{aligned}
\frac{\partial(D_{1,x})}{\partial x} &= \nu\sqrt{h}\left(\frac{3}{2}\frac{\partial h}{\partial x}\frac{\partial \bar{u}}{\partial x} + h\frac{\partial^2 \bar{u}}{\partial x^2}\right), \\
\frac{\partial(D_{1,y})}{\partial y} &= \frac{1}{2}\nu\sqrt{h}\left(\frac{3}{2}\frac{\partial h}{\partial y}\left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right) + h\left(\frac{\partial^2 \bar{u}}{\partial y^2} + \frac{\partial^2 \bar{v}}{\partial x \partial y}\right)\right), \\
\frac{\partial(D_{2,x})}{\partial x} &= \frac{1}{2}\nu\sqrt{h}\left(\frac{3}{2}\frac{\partial h}{\partial x}\left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right) + h\left(\frac{\partial^2 \bar{u}}{\partial x \partial y} + \frac{\partial^2 \bar{v}}{\partial x^2}\right)\right), \\
\frac{\partial(D_{2,y})}{\partial y} &= \nu\sqrt{h}\left(\frac{3}{2}\frac{\partial h}{\partial y}\frac{\partial \bar{v}}{\partial y} + h\frac{\partial^2 \bar{v}}{\partial y^2}\right),
\end{aligned}$$

and

$$S = \begin{pmatrix} 0 \\ gh\cos\theta - \mu(h, Fr)\frac{\bar{u}}{\sqrt{\bar{u}^2+\bar{v}^2}}hg\cos\theta \\ -\mu(h, Fr)\frac{\bar{v}}{\sqrt{\bar{u}^2+\bar{v}^2}}hg\cos\theta \end{pmatrix}, \tag{4.4}$$

where $\mu(h, Fr)$ is given in Equations 2.19-2.21.

## 4.1 Preprocessing of a Patch

The implementation of the *non-conservative product* $B\nabla Q$ in ExaHyPE 2 involves the following two challenges:

1. The $x$-component involves partial derivatives in $y$-direction, and, the $y$-component involves partial derivatives in $x$-direction.
2. Existence of double derivatives.

These challenges occur because the *deltaQ* parameter in the `nonConservativeProduct` function in Algorithm 6 contains only first-order derivatives in the normal-direction $n$. To overcome these challenges, we treat the derivative terms as auxiliary variables and store them in the vector of unknowns $Q$. Thus, the vector of unknowns $Q$ programmed in ExaHyPE 2 contains values of 16 variables as listed in Table 4.1. The auxiliary variables containing partial derivatives of $h, \bar{u}, \bar{v}$ are calculated using the finite-difference method in the `preprocess_reconstruced_patch` routine provided by ExaHyPE 2. This routine is applied to all the patches before the start of every time step. Hence, the derivatives get recalculated and updated before the start of every time step. However, there is another problem here. As discussed in Chapter 3 and also seen in Figure 4.1, the data in the corner halo cells ($C_1, C_5, C_{21}, C_{25}$) is not transferred and updated between patches. From Equations 4.5-4.8, we can see that calculation of mixed derivative for $V \in \{h, h\bar{u}, \bar{v}\}$ i.e., $\frac{\partial^2 V}{\partial x \partial y}$ at the corner internal cells ($C_7, C_9, C_{17}, C_{19}$) using the central-difference scheme requires the data at the respective corner halo cells ($C_1, C_5, C_{21}, C_{25}$):

$$\frac{\partial^2 V_{C_7}}{\partial x \partial y} \approx \frac{V_{C_3} - V_{C_1} - V_{C_{13}} + V_{C_{11}}}{4\Delta x \Delta y}, \tag{4.5}$$

$$\frac{\partial^2 V_{C_9}}{\partial x \partial y} \approx \frac{V_{C_5} - V_{C_3} - V_{C_{15}} + V_{C_{13}}}{4\Delta x \Delta y}, \tag{4.6}$$

$$\frac{\partial^2 V_{C_{17}}}{\partial x \partial y} \approx \frac{V_{C_{13}} - V_{C_{11}} - V_{C_{23}} + V_{C_{21}}}{4\Delta x \Delta y}, \tag{4.7}$$

$$\frac{\partial^2 V_{C_{19}}}{\partial x \partial y} \approx \frac{V_{C_{15}} - V_{C_{13}} - V_{C_{25}} + V_{C_{23}}}{4\Delta x \Delta y}. \tag{4.8}$$

To resolve the problem caused by missing data in the corner halo cells, we extrapolate the values of all unknown variables according to Equations 4.9-4.12.

$$V_{C_1} = \frac{V_{C_2} + V_{C_6}}{2}, \tag{4.9}$$

$$V_{C_5} = \frac{V_{C_4} + V_{C_{10}}}{2}, \tag{4.10}$$

$$V_{C_{21}} = \frac{V_{C_{16}} + V_{C_{22}}}{2}, \tag{4.11}$$

| Index | Variable | Variable Description | Variable Type |
|---|---|---|---|
| 0 | $h$ | Height of granular material. | Unknown |
| 1 | $hu$ | Height$\times$Depth-averaged velocity along $x$-direction. | Unknown |
| 2 | $hv$ | Height$\times$Depth-averaged velocity along $y$-direction. | Unknown |
| 3 | $b$ | Bathymetry of the bottom. | Auxiliary |
| 4 | $h_x$ | $\frac{\partial h}{\partial x}$ | Auxiliary |
| 5 | $h_y$ | $\frac{\partial h}{\partial y}$ | Auxiliary |
| 6 | $u_x$ | $\frac{\partial \bar{u}}{\partial x}$ | Auxiliary |
| 7 | $u_y$ | $\frac{\partial \bar{u}}{\partial y}$ | Auxiliary |
| 8 | $u_{xx}$ | $\frac{\partial^2 \bar{u}}{\partial x^2}$ | Auxiliary |
| 9 | $u_{yy}$ | $\frac{\partial^2 \bar{u}}{\partial y^2}$ | Auxiliary |
| 10 | $u_{xy}$ | $\frac{\partial^2 \bar{u}}{\partial x \partial y}$ | Auxiliary |
| 11 | $v_x$ | $\frac{\partial \bar{v}}{\partial x}$ | Auxiliary |
| 12 | $v_y$ | $\frac{\partial \bar{v}}{\partial y}$ | Auxiliary |
| 13 | $v_{xx}$ | $\frac{\partial^2 \bar{v}}{\partial x^2}$ | Auxiliary |
| 14 | $v_{yy}$ | $\frac{\partial^2 \bar{v}}{\partial y^2}$ | Auxiliary |
| 15 | $v_{xy}$ | $\frac{\partial^2 \bar{v}}{\partial x \partial y}$ | Auxiliary |

Table 4.1: List of Variables in $Q$

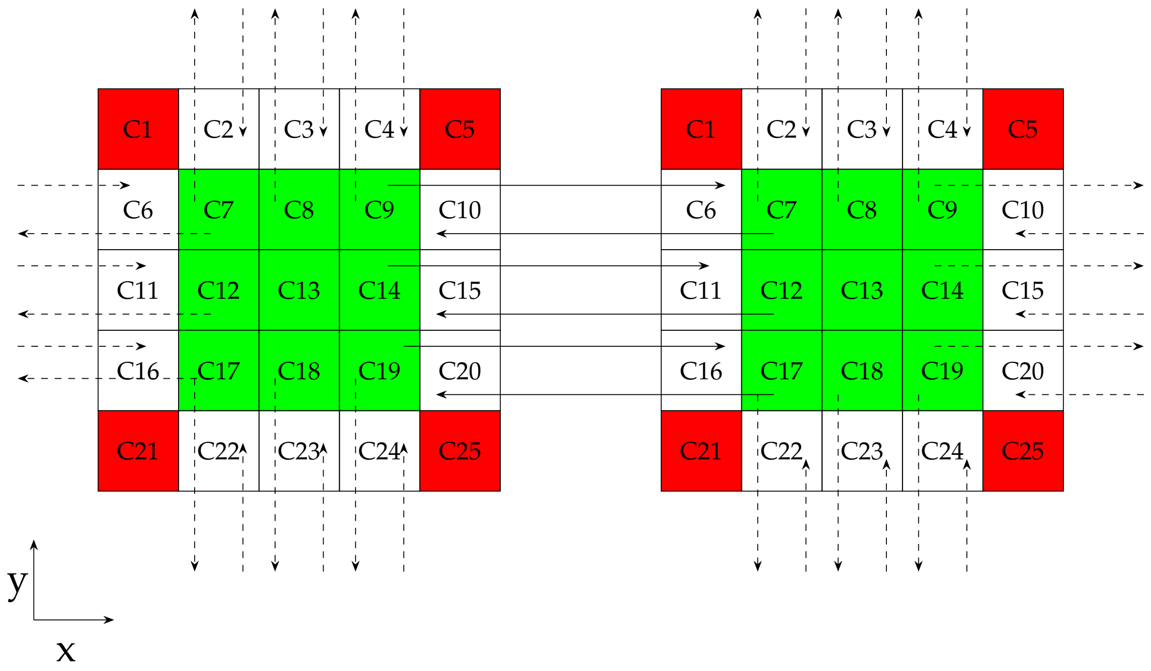


Figure 4.1: Data transfer between halo cells of two patches.

$$V_{C_{25}} = \frac{V_{C_{24}} + V_{C_{20}}}{2}, \tag{4.12}$$

where $V \in \{h, h\bar{u}, h\bar{v}\}$.

Algorithm 7 is included in the `preprocess_reconstruced_patch` routine to extrapolate values in corner halo cells before calculating derivatives (cf. Algorithm 8).

Algorithm 8 calculates single derivatives using the backward difference scheme and double derivatives using the central difference scheme. Note that only the numerator of these schemes is calculated in Algorithm 8 because ExaHyPE 2 does not provide the grid size values $\Delta x$ and $\Delta y$ in the `preprocess_reconstruced_patch` routine. The denominator is applied in the `nonConservativeProduct` function in Algorithm 14 and the `sourceTerm` function in Algorithm 15, where the derivatives are used, and the values for grid sizes $\Delta x$ and $\Delta y$ are also available.

## 4.2 Algorithms for the Depth-Averaged Avalanche Equations

The initial conditions in Algorithm 9 are set such that the height inside the granular material region is $8\ mm$, and outside this region, it is $0$ since there is no material. This condition is mainly used in the following Chapter 5 for evaluations.

Reflective boundary conditions are programmed in Algorithm 10 such that the material height at the boundary is equal to the material height in the cell adjacent to the boundary

**Algorithm 7:** Implementation of the preprocessing function to extrapolate halo cells

```
Input: $C_i$: Memory location of all volume denoted by $Ci$ in Figure 3.1.
       $variableList$: Variable enumerations in each volume as given in Table 4.1.
Output: $patch$: Linearised array containing all volumes with extrapolated corner
        halo cells in the patch.
Data: Volume names are used as per Figure 3.1.
1 Function extrapolateHalo($C_i, variableList$):
2     for $variable \in variableList$ do
3         $C_1[variable] \leftarrow \frac{1}{2}(C_2[variable] + C_6[variable])$
4         $C_5[variable] \leftarrow \frac{1}{2}(C_4[variable] + C_{10}[variable])$
5         $C_{21}[variable] \leftarrow \frac{1}{2}(C_{16}[variable] + C_{22}[variable])$
6         $C_{25}[variable] \leftarrow \frac{1}{2}(C_{20}[variable] + C_{24}[variable])$
7     return $patch$
```

inside the domain. Velocity at the boundary is equal and opposite to the velocity in the cell adjacent to the boundary inside the domain.

In Algorithm 11, for simplicity, we do not refine the volume anywhere in the domain and always return $Keep$. To refine specific parts of the domain we can impose conditions to return $Refine$ based on the coordinates of the volume. Similarly, to coarsen certain parts of the domain, we can return $Erase$.

The eigenvalues of the system are the eigenvalues of the Jacobian of the *flux* function. Since depth-averaged avalanche equations are a system of three PDEs, we get three eigenvalues in each of the two components (dimensions). The eigenvalues in $x$-direction are the eigenvalues of the Jacobian of the $x$-component of *flux* in Equation 4.2 which is $(h\bar{u}, h\bar{u}^2 + \frac{1}{2}g\cos\theta h^2, h\bar{u}\bar{v})^T$. Similarly, the eigenvalues in $y$-direction are the eigenvalues of the Jacobian of the $y$-component of *flux* which is $(h\bar{v}, h\bar{u}\bar{v}, h\bar{v}^2 + \frac{1}{2}g\cos\theta h^2)^T$. Upon calculation, the eigenvalues in $x$- and $y$-direction turn out to be $\lambda_{x_1} = \bar{u}, \lambda_{x_2} = \bar{u} - \sqrt{hg\cos\theta}, \lambda_{x_3} = \bar{u} + \sqrt{hg\cos\theta}$ and $\lambda_{y_1} = \bar{v}, \lambda_{y_2} = \bar{v} - \sqrt{hg\cos\theta}, \lambda_{y_3} = \bar{v} + \sqrt{hg\cos\theta}$, respectively. The `maxEigenvalue` function in Algorithm 12 returns the maximum eigenvalue in the normal-direction.

From Equation 4.2, we can see that the $x$-component of the *flux* is $(hu, hu^2 + \frac{1}{2}g\cos\theta h^2, huv)^T$ and the $y$-component is $(hv, huv, hv^2 + \frac{1}{2}g\cos\theta h^2)^T$. One of the two components is returned by Algorithm 13 based on the value of $n$.

From Equation 4.3 we can see that the $x$-component of the *non-conservative product* is $-(0, \nu\sqrt{h}(\frac{3}{2}\frac{\partial h}{\partial x}\frac{\partial u}{\partial x} + h\frac{\partial^2 u}{\partial x^2}), \frac{1}{2}\nu\sqrt{h}(\frac{3}{2}\frac{\partial h}{\partial x}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) + h(\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2})))^T$ and the $y$-component is $-(0, \frac{1}{2}\nu\sqrt{h}(\frac{3}{2}\frac{\partial h}{\partial y}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) + h(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y})), \nu\sqrt{h}(\frac{3}{2}\frac{\partial h}{\partial y}\frac{\partial v}{\partial y} + h\frac{\partial^2 v}{\partial y^2}))^T$. The loop body of the finite volume Rusanov solver in ExaHyPE 2 passes the average of the variable value in the current volume and the volume on the right in the normal-direction to the `nonConservativeProduct` function in Algorithm 14. The function has another parameter named $deltaQ$, which is the difference between the variable value in the right cell in the normal-direction and the current cell. The true value of the auxiliary variables (derivatives) in $Q$ is retrieved using $Q$ and $deltaQ$ and divided by the respective of `volumeH` to apply the denominator of the finite-difference formulation. This denominator is applied because it is only the numerator which is calculated and stored in Algorithm 8. These derivative values are then used to compute the *non-conservative product* in Algorithm 14.

The *source term* from Equation 4.4 is incorporated in Algorithm 15. The algorithm first calculates the coefficient of basal friction $\mu$ based on Equations 2.19-2.21, which is then used to calculate the *source term*. The input parameters to Algorithm 15 are mostly material parameters derived from experiments.

**Algorithm 8:** Implementation of the preprocessing function to calculate derivatives

```
Input: $C_i$: Memory location of volumes denoted by $Ci$ in Figure 3.1.
Output: $patch$: Linearised array containing all volumes with updated derivatives
        in the patch.
Data: $C_i^{(x,y)}$: Address of the neighbour volume of $C_i$ with relative $x$-position $= x$
      and relative $y$-position $= y$.
      $internalVolumes$: List of volumes inside the patch denoted by green in
      Figure 3.1.
1  Function calculateDerivatives($C_i$):
2      for $C_i \in internalVolumes$ do
           // Derivatives of $h$
3          $C_i[h_x] \leftarrow C_i[h] - C_i^{(-1,0)}[h]$
4          $C_i[h_y] \leftarrow C_i[h] - C_i^{(0,-1)}[h]$
           // Derivatives of $u$
5          $C_i[u_x] \leftarrow \frac{C_i[hu]}{C_i[h]} - \frac{C_i^{(-1,0)}[hu]}{C_i^{(-1,0)}[h]}$
6          $C_i[u_y] \leftarrow \frac{C_i[hu]}{C_i[h]} - \frac{C_i^{(0,-1)}[hu]}{C_i^{(0,-1)}[h]}$
7          $C_i[u_{xx}] \leftarrow \frac{C_i^{(1,0)}[hu]}{C_i^{(1,0)}[h]} - 2\frac{C_i[hu]}{C_i[h]} + \frac{C_i^{(-1,0)}[hu]}{C_i^{(-1,0)}[h]}$
8          $C_i[u_{yy}] \leftarrow \frac{C_i^{(0,1)}[hu]}{C_i^{(0,1)}[h]} - 2\frac{C_i[hu]}{C_i[h]} + \frac{C_i^{(0,-1)}[hu]}{C_i^{(0,-1)}[h]}$
9          $C_i[u_{xy}] \leftarrow \frac{C_i^{(1,1)}[hu]}{C_i^{(1,1)}[h]} - \frac{C_i^{(1,-1)}[hu]}{C_i^{(1,-1)}[h]} - \frac{C_i^{(-1,1)}[hu]}{C_i^{(-1,1)}[h]} + \frac{C_i^{(-1,-1)}[hu]}{C_i^{(-1,-1)}[h]}$
           // Derivatives of $v$
10         $C_i[v_x] \leftarrow \frac{C_i[hv]}{C_i[h]} - \frac{C_i^{(-1,0)}[hv]}{C_i^{(-1,0)}[h]}$
11         $C_i[v_y] \leftarrow \frac{C_i[hv]}{C_i[h]} - \frac{C_i^{(0,-1)}[hv]}{C_i^{(0,-1)}[h]}$
12         $C_i[v_{xx}] \leftarrow \frac{C_i^{(1,0)}[hv]}{C_i^{(1,0)}[h]} - 2\frac{C_i[hv]}{C_i[h]} + \frac{C_i^{(-1,0)}[hv]}{C_i^{(-1,0)}[h]}$
13         $C_i[v_{yy}] \leftarrow \frac{C_i^{(0,1)}[hv]}{C_i^{(0,1)}[h]} - 2\frac{C_i[hv]}{C_i[h]} + \frac{C_i^{(0,-1)}[hv]}{C_i^{(0,-1)}[h]}$
14         $C_i[v_{xy}] \leftarrow \frac{C_i^{(1,1)}[hv]}{C_i^{(1,1)}[h]} - \frac{C_i^{(1,-1)}[hv]}{C_i^{(1,-1)}[h]} - \frac{C_i^{(-1,1)}[hv]}{C_i^{(-1,1)}[h]} + \frac{C_i^{(-1,-1)}[hv]}{C_i^{(-1,-1)}[h]}$
15     return $patch$
```

**Algorithm 9:** Implementation of the initial conditions function for DAE

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $volumeCentre$: Vector containing coordinates of the centre of the current
       volume.
Output: $Q$: Vector with initialised unknown and auxiliary variables.
Function initialConditions($Q, volumeCentre$):
    if $volumeCentre \in granularRegion$ then
        $Q[h] \leftarrow 0.008$
    else
        $Q[h] \leftarrow 0.0$
    $Q[hu] \leftarrow 0.0$        // initial velocity in $x$-direction is 0
    $Q[hv] \leftarrow 0.0$        // initial velocity in $y$-direction is 0
    $Q[b] \leftarrow 0.0$                           // assuming flat bottom

    // All the remaining auxiliary variables get initialised
       to 0.0

    $Q[h_x] \leftarrow 0.0$
    $Q[h_y] \leftarrow 0.0$
    $Q[u_x] \leftarrow 0.0$
    $Q[u_y] \leftarrow 0.0$
    $Q[u_{xx}] \leftarrow 0.0$
    $Q[u_{yy}] \leftarrow 0.0$
    $Q[u_{xy}] \leftarrow 0.0$
    $Q[v_x] \leftarrow 0.0$
    $Q[v_y] \leftarrow 0.0$
    $Q[v_{xx}] \leftarrow 0.0$
    $Q[v_{yy}] \leftarrow 0.0$
    $Q[v_{xy}] \leftarrow 0.0$
    return $Q$
```

**Algorithm 10:** Implementation of the reflective boundary conditions function for DAE

```
Input: $Qinside$: Vector containing unknown and auxiliary variables of volume
       located just inside the domain.
       $Qoutside$: Vector containing unknown and auxiliary variables of the
       volume at the boundary.
Output: $Qoutside$: Vector containing unknown and auxiliary variables with
        assigned boundary conditions.
Function boundaryConditions($Qinside, Qoutside$):
    $Qoutside[h] \leftarrow Qinside[h]$
    $Qoutside[hu] \leftarrow -Qinside[hu]$     // reflective boundary condition
    $Qoutside[hv] \leftarrow -Qinside[hv]$     // reflective boundary condition

    // boundary values of auxiliary variables are set to
       inside values, they are never used

    $Qoutside[b] \leftarrow Qinside[b]$
    $Qoutside[h_x] \leftarrow Qinside[h_x]$
    $Qoutside[h_y] \leftarrow Qinside[h_y]$
    $Qoutside[u_x] \leftarrow Qinside[u_x]$
    $Qoutside[u_y] \leftarrow Qinside[u_y]$
    $Qoutside[u_{xx}] \leftarrow Qinside[u_{xx}]$
    $Qoutside[u_{yy}] \leftarrow Qinside[u_{yy}]$
    $Qoutside[u_{xy}] \leftarrow Qinside[u_{xy}]$
    $Qoutside[v_x] \leftarrow Qinside[v_x]$
    $Qoutside[v_y] \leftarrow Qinside[v_y]$
    $Qoutside[v_{xx}] \leftarrow Qinside[v_{xx}]$
    $Qoutside[v_{yy}] \leftarrow Qinside[v_{yy}]$
    $Qoutside[v_{xy}] \leftarrow Qinside[v_{xy}]$
    return $Qoutside$
```

**Algorithm 11:** Implementation of the refinement criterion function for DAE

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $volumeCentre$: Vector containing coordinates of the centre of current
       volume with $\{0 : x, 1 : y\}$.
Output: $refinementCommand \in \{Keep, Refine, Erase\}$: Refinement option.
Function refinementCriterion($Q, volumeCentre$):
    return $Keep$
```

**Algorithm 12:** Implementation of the function calculating max eigenvalue for DAE

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $n \in \{x, y\}$: Normal-direction.
       $\theta$: Angle of inclination of the inclined plane.
Output: $maxEigenvalue$: Maximum eigenvalue in the direction pointed by $n$.
1  Function maxEigenvalue($Q$, $n$, $\theta$):
2      if $Q[h] > 0$ then
3          $g \leftarrow 9.81$
4          $c \leftarrow \sqrt{gQ[h]\cos\theta}$
5          if $n$ is along $x$ then
6              $u \leftarrow Q[hu]/Q[h]$
7              $maxEigenvalue \leftarrow \max(|u - c|, |u + c|)$
8          else if $n$ is along $y$ then
9              $v \leftarrow Q[hv]/Q[h]$
10             $maxEigenvalue \leftarrow \max(|v - c|, |v + c|)$
11         return $maxEigenvalue$
12     else
13         return 0.0
```

**Algorithm 13:** Implementation of the *flux* function for DAE

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $n \in \{x, y\}$: Normal-direction.
       $\theta$: Angle of inclination of the inclined plane.
Output: $F$: Vector containing flux in the normal-direction.
Function flux($Q, n, \theta$):
    if $Q[h] > 0$ then
        $g \leftarrow 9.81$
        if $n$ is along $x$ then
            $F[0] \leftarrow Q[hu]$                                                         // $hu$
            $F[1] \leftarrow Q[h]\left(\frac{Q[hu]}{Q[h]}\right)^2 + \frac{1}{2}g\cos\theta(Q[h])^2$    // $hu^2 + \frac{1}{2}g\cos\theta h^2$
            $F[2] \leftarrow Q[hu]\left(\frac{Q[hv]}{Q[h]}\right)$                                // $huv$
        else if $n$ is along $y$ then
            $F[0] \leftarrow Q[hv]$                                                         // $hv$
            $F[1] \leftarrow Q[hu]\left(\frac{Q[hv]}{Q[h]}\right)$                                // $huv$
            $F[2] \leftarrow Q[h]\left(\frac{Q[hv]}{Q[h]}\right)^2 + \frac{1}{2}g\cos\theta(Q[h])^2$    // $hv^2 + \frac{1}{2}g\cos\theta h^2$
    else
        $F[0] \leftarrow 0.0$
        $F[1] \leftarrow 0.0$
        $F[2] \leftarrow 0.0$
    return $F$
```

**Algorithm 14:** Implementation of the *non-conservative product* function for DAE

```
Input: $Q$: Vector containing forward average values of unknown and auxiliary
       variables.
       $deltaQ$: Vector containing forward differences of unknown and auxiliary
       variables.
       $n \in \{x, y\}$: Normal-direction.
       $\nu$: Coefficient of viscosity.
       $volumeH$: Vector containing grid size in each dimension $\{0 : x, 1 : y\}$.
Output: $BTimesDeltaQ$: Vector containing the component of the non-conservative
        term in the normal-direction.
Function nonConservativeProduct($Q$, $deltaQ$, $n$, $\nu$, $volumeH$):
    if $Q[h] > 0$ then
        // Retrieving Derivatives of $h$ from $Q$
        $h_x \leftarrow (Q[h_x] - 0.5 * deltaQ[h_x])/volumeH(0)$
        $h_y \leftarrow (Q[h_y] - 0.5 * deltaQ[h_y])/volumeH(1)$
        // Retrieving values and calculating derivatives of $u$
        $u_x \leftarrow (Q[u_x] - 0.5 * deltaQ[u_x])/volumeH(0)$
        $u_y \leftarrow (Q[u_y] - 0.5 * deltaQ[u_y])/volumeH(1)$
        $u_{xx} \leftarrow (Q[u_{xx}] - 0.5 * deltaQ[u_{xx}])/(volumeH(0))^2$
        $u_{yy} \leftarrow (Q[u_{xy}] - 0.5 * deltaQ[u_{yy}])/(volumeH(1))^2$
        $u_{xy} \leftarrow (Q[u_{xy}] - 0.5 * deltaQ[u_{xy}])/(4 * volumeH(0) * volumeH(1))$
        // Retrieving values and calculating derivatives of $v$
        $v_x \leftarrow (Q[v_x] - 0.5 * deltaQ[v_x])/volumeH(0)$
        $v_y \leftarrow (Q[v_y] - 0.5 * deltaQ[v_y])/volumeH(1)$
        $v_{xx} \leftarrow (Q[v_{xx}] - 0.5 * deltaQ[v_{xx}])/(volumeH(0))^2$
        $v_{yy} \leftarrow (Q[v_{xy}] - 0.5 * deltaQ[v_{yy}])/(volumeH(1))^2$
        $v_{xy} \leftarrow (Q[v_{xy}] - 0.5 * deltaQ[v_{xy}])/(4 * volumeH(0) * volumeH(1))$
        if $n$ is along $x$ then
            $BTimesDeltaQ[0] \leftarrow 0.0$
            $BTimesDeltaQ[1] \leftarrow -volumeH(0)\nu\sqrt{Q[h]}(\frac{3}{2}h_x u_x + Q[h]u_{xx})$
            $BTimesDeltaQ[2] \leftarrow$
              $-\frac{1}{2}volumeH(0)\nu\sqrt{Q[h]}(\frac{3}{2}h_x(u_y + v_x) + Q[h](u_{xy} + v_{xx}))$
        else if $n$ is along $y$ then
            $BTimesDeltaQ[0] \leftarrow 0.0$
            $BTimesDeltaQ[1] \leftarrow$
              $-\frac{1}{2}volumeH(1)\nu\sqrt{Q[h]}(\frac{3}{2}h_y(u_y + v_x) + Q[h](u_{yy} + v_{xy}))$
            $BTimesDeltaQ[2] \leftarrow -volumeH(1)\nu\sqrt{Q[h]}(\frac{3}{2}h_y v_y + Q[h]v_{yy})$
    else
        $BTimesDeltaQ[0] \leftarrow 0.0$
        $BTimesDeltaQ[1] \leftarrow 0.0$
        $BTimesDeltaQ[2] \leftarrow 0.0$
    return $BTimesDeltaQ$
```

**Algorithm 15:** Implementation of the *source term* function for DAE

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $\theta$: Angle of inclination of the inclined plane.
       $\mu_i \forall i \in \{1,2,3\}, \beta, \gamma, \beta_*, \kappa$: Material parameters derived from experiments.
       $L$: Friction length scale (scaling parameter).
       $volumeH$: Vector containing grid sizes in each dimension $\{0 : x, 1 : y\}$.
Output: $S$: Vector containing source terms.
Function sourceTerm($Q, \theta, \mu_1, \mu_2, \mu_3, \beta, \gamma, \beta_*, \kappa, L$):
    if $Q[h] > 0$ then
        $g \leftarrow 9.81$
        $\bar{u} \leftarrow \sqrt{Q[hu]^2 + Q[hv]^2}/Q[h]$
        $Fr \leftarrow \bar{u}/\sqrt{g \cos\theta Q[h]}$
        if $Fr > \beta_*$ then
            $v_{stop} \leftarrow Q[h] * \beta/(Fr + \gamma)$
            $\mu \leftarrow \mu_1 + (\mu_1 - \mu_2)/(1 + v_{stop}/L)$
        else if $Fr \in (0, \beta_*]$ then
            $v_{start} \leftarrow Q[h]$
            $v_{stop} \leftarrow Q[h]\beta/(\beta_* + \gamma)$
            $\mu_{start} \leftarrow \mu_3 + (\mu_1 - \mu_2)/(1 + v_{start}/L)$
            $\mu_{stop} \leftarrow \mu_1 + (\mu_1 - \mu_2)/(1 + v_{stop}/L)$
            $\mu \leftarrow \mu_{start} + (\mu_{stop} - \mu_{start})(Fr/\beta_*)$
        else if $Fr=0$ then
            $v_{start} \leftarrow Q[h]$
            $\mu_{start} \leftarrow \mu_3 + (\mu_1 - \mu_2)/(1 + v_{start}/L)$
            $frictionVector \leftarrow (\tan\theta - Q[h_x]/volumeH(0), -Q[h_y]/volumeH(1))$
            $frictionVectorMagnitude \leftarrow \|frictionVector\|_2$
            $\mu \leftarrow \min(\mu_{start}, frictionVectorMagnitude)$
        if $\bar{u} = 0$ then
            $S[0] \leftarrow 0$
            $S[1] \leftarrow gQ[h]\sin\theta - \mu \frac{frictionVector(0)}{frictionVectorMagnitude} gQ[h]\cos\theta$
            $S[2] \leftarrow -\mu \frac{frictionVector(1)}{frictionVectorMagnitude} gQ[h]\cos\theta$
        else
            $S[0] \leftarrow 0.0$
            $S[1] \leftarrow gQ[h]\sin\theta - \mu \frac{Q[hu]}{\bar{u}} g\cos\theta$
            $S[2] \leftarrow -\mu \frac{Q[hv]}{\bar{u}} g\cos\theta$
    else
        $S[0] \leftarrow 0.0$
        $S[1] \leftarrow 0.0$
        $S[2] \leftarrow 0.0$
    return $S$
```

# 5 Results

This chapter presents the results of the various evaluations that were done to validate and explore the features of the implementation of the depth-averaged avalanche equations (DAE) in ExaHyPE 2. The results of the evaluations are presented as 2D-surface plots and the line plots. All simulation results were obtained using the finite volume method with Rusanov flux and adaptive time stepping.

## 5.1 Validation

Validation is a critical step in the development of scientific code. It ensures that the code produces accurate and reliable results that align with the theoretical predictions or experimental data. A scientific code is built based on mathematical models representing natural processes. Validation helps verify that these models are implemented correctly in the code. It ensures that the mathematical algorithms used to describe the phenomena are translated accurately in the computational routines.

This section discusses the steps taken for the validation of the implementation of the depth-averaged avalanche equations (Equations 2.56-2.58) in ExaHyPE 2.

### 5.1.1 Convergence to the Shallow-Water Equations

In the depth-averaged avalanche equations (Equations 2.56-2.58), if we make the angle of inclination zero ($\theta = 0$), the coefficient of viscosity zero ($\nu = 0$), and the source terms $S_x$ and $S_y$ zero ($S_x = 0$ and $S_y = 0$), then the equations converge to the shallow-water equations (Equations 2.1-2.3). By making the parameters mentioned above zero in the code, the simulation results should yield the solution of shallow-water equations. To observe this, we apply the equations to a square domain of size $2\ m \times 2\ m$ and initialise the unknown variables as per Equations 5.1-5.3 using Algorithm 1.

$$h(x,y,0) = \begin{cases} 1.5 & \sqrt{x^2+y^2} < 0.25 \\ 1.0 & \sqrt{x^2+y^2} \geq 0.25 \end{cases}, \forall\{x,y\} \in [-1,1] \times [-1,1], \tag{5.1}$$

$$h\bar{u}(x,y,0) = 0, \forall\{x,y\} \in [-1,1] \times [-1,1], \tag{5.2}$$

$$h\bar{v}(x,y,0) = 0, \forall\{x,y\} \in [-1,1] \times [-1,1]. \tag{5.3}$$

The initial height ($h$) is plotted in Figure 5.1. It can be seen that the height is $1.5\ m$ in a circle of radius $0.25\ m$ located at the origin and $1.0\ m$ in the rest of the domain. The initial conditions resemble a dam-break scenario. Reflective boundary conditions are applied at the boundaries using Algorithm 10.

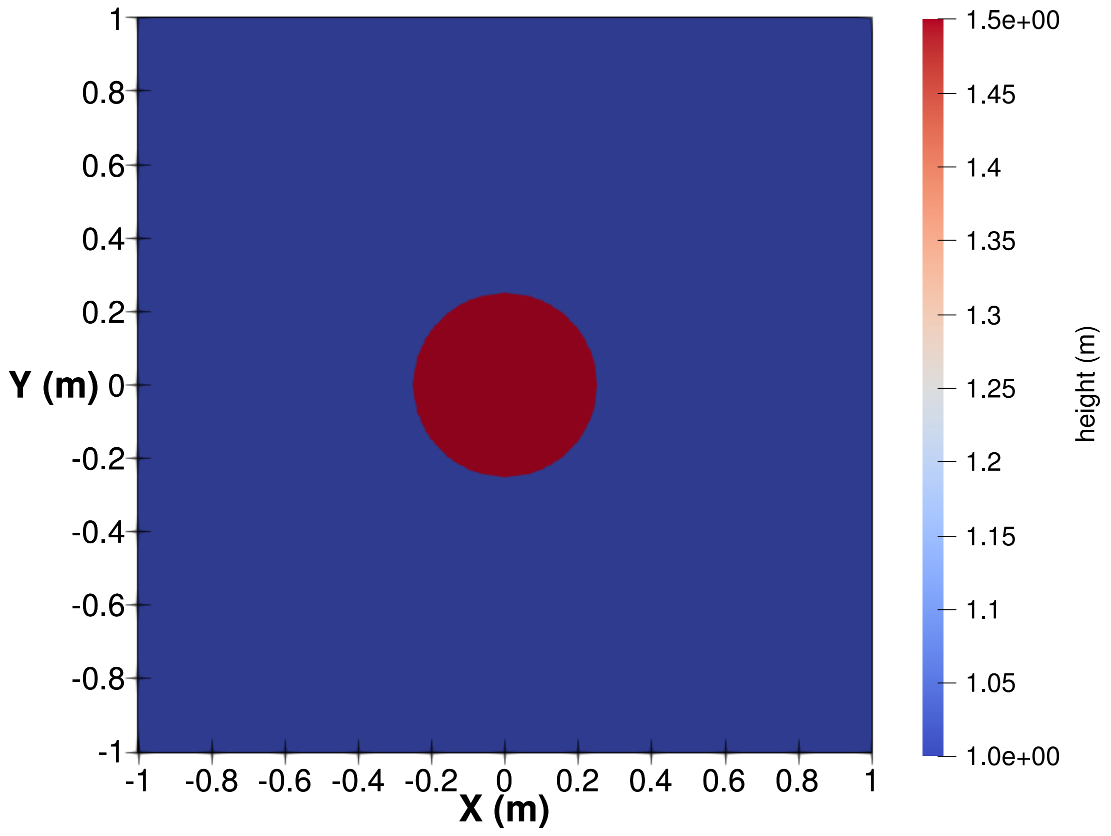
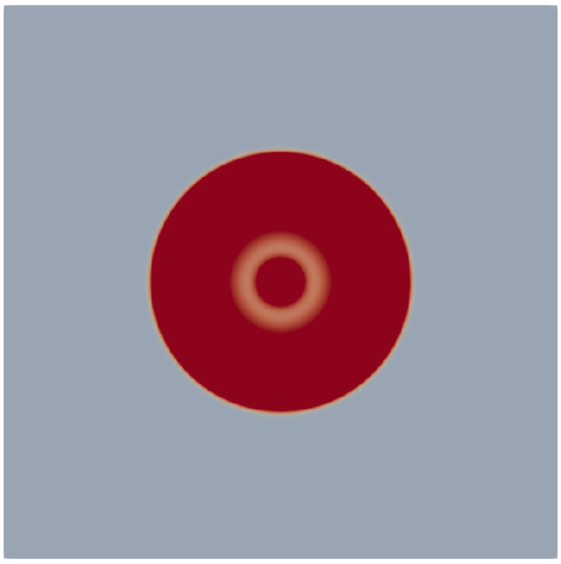
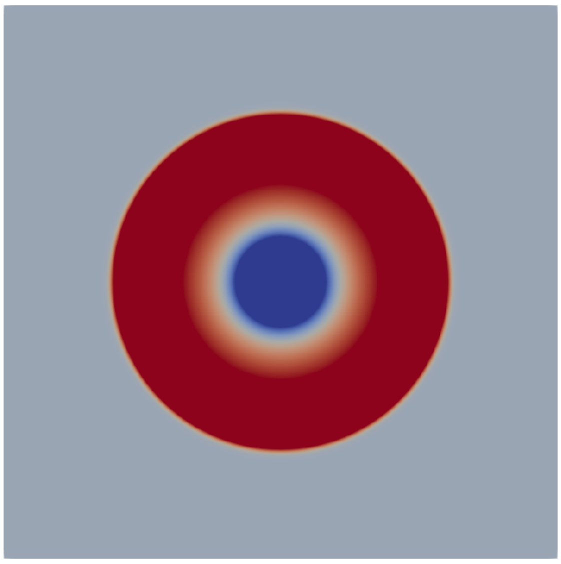
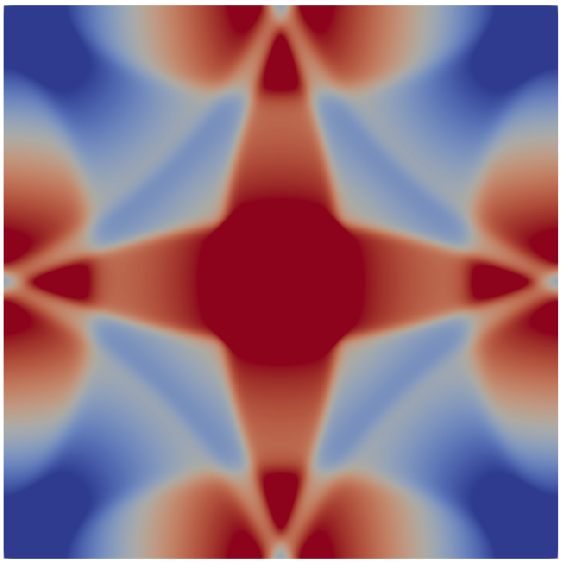


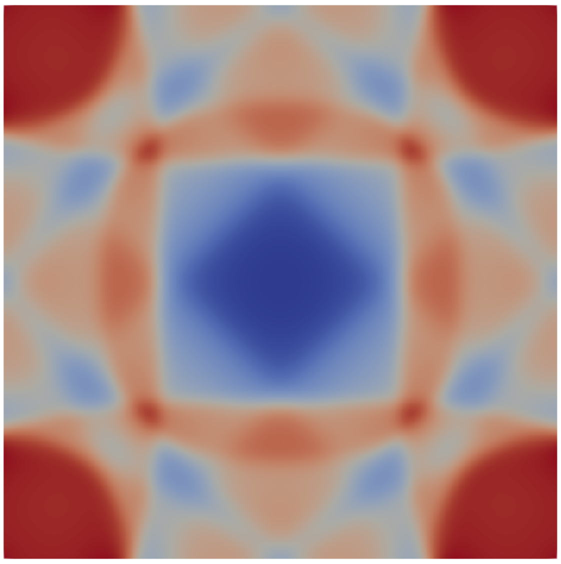
Figure 5.1: 2D-surface plot of material height at $t = 0\ s$.

From the surface plots in Figure 5.2, it can be seen that the solution preserves symmetry at all time steps, which is expected in such a dam-break scenario with symmetric initial conditions. In Figure 5.3, the solution obtained here using the depth-averaged avalanche equations' application (DAE) is compared with the one obtained using the already present shallow-water equations' application (SWE) in ExaHyPE 2. The comparison is done through line plots of $h$ (granular material height) along the $x$-axis. It can be seen from Figure 5.3 that the DAE curve (blue) completely overlaps the SWE curve (red) at all time steps. Hence, only the DAE curve (blue) is visible. The line plots quantitatively validate the convergence of depth-averaged avalanche equations to the shallow water equations in our application.


(a) $t = 0.06s$


(b) $t = 0.1\ s$


(c) $t = 0.58\ s$


(d) $t = 1.0\ s$

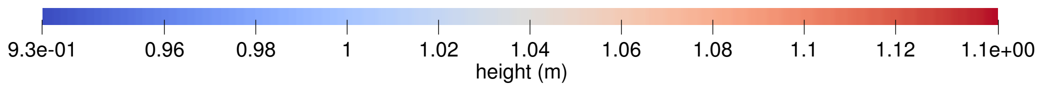
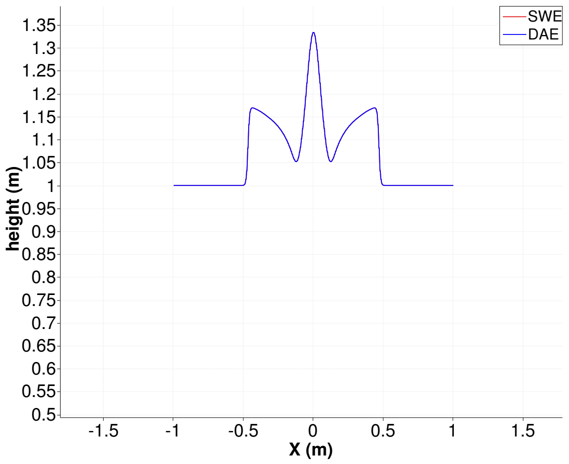
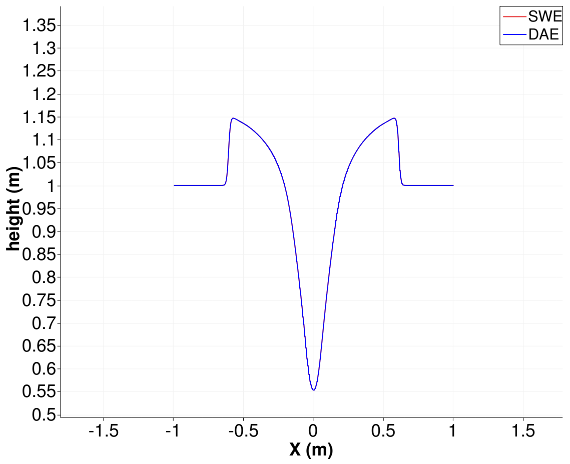


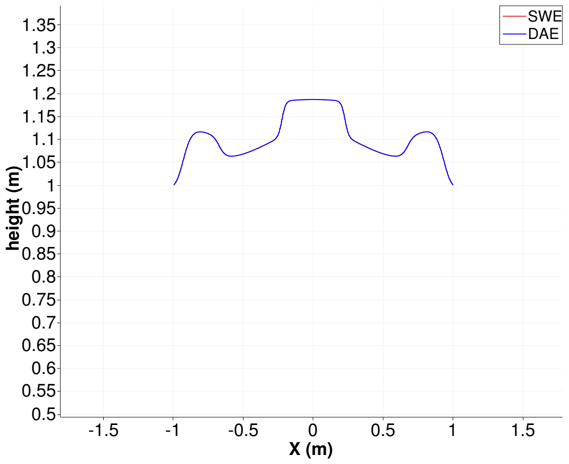
Figure 5.2: 2D-surface plot of granular material height at different points in time.



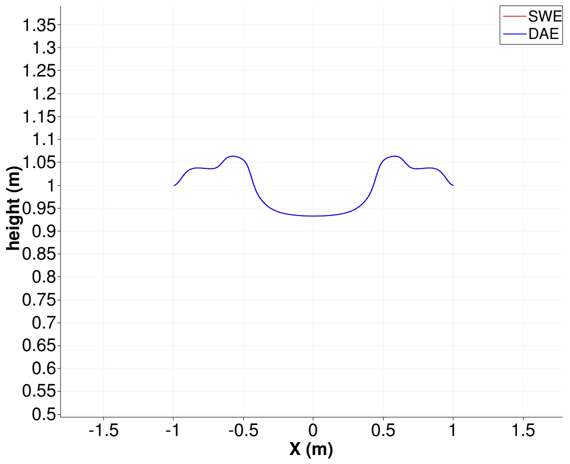
(a) $t = 0.06\ s$



(b) $t = 0.1\ s$



(c) $t = 0.58\ s$



(d) $t = 1.0\ s$

Figure 5.3: Line plot of granular material height along $x$-axis comparing results from SWE and DAE at different points in time.

### 5.1.2 Simulation Using the Complete Depth-Averaged Avalanche Equations

The depth-averaged avalanche equations (Equations 2.56-2.58) were simulated on a rectangular plane of size $1.58\ m \times 0.7\ m$ with $x \in [0, 1.58]$ and $y \in [0, 0.7]$, inclined at an angle $\theta = 35°$. The initial conditions were applied such that there is a circular patch of granular material with thickness $h = 8\ mm$ centred at $(0.15, 1.0)$ as shown in Figure 5.4. The mathematical formulation of the initial conditions for unknown variables is given in Equations 5.4-5.6. These equations were programmed using Algorithm 9. Note that all the following sections and subsections use this domain setup and the same initial and boundary conditions as used here.

$$h(x,y,0) = \begin{cases} 0.008 & \sqrt{(x-0.15)^2 + (y-1.0)^2} < 0.1 \\ 0 & \sqrt{(x-0.15)^2 + (y-1.0)^2} \geq 0.1 \end{cases}, \forall\{x,y\} \in [0,1.58] \times [0,0.7], \tag{5.4}$$

$$h\bar{u}(x,y,0) = 0, \forall\{x,y\} \in [0,1.58] \times [0,0.7], \tag{5.5}$$

$$h\bar{v}(x,y,0) = 0, \forall\{x,y\} \in [0,1.58] \times [0,0.7]. \tag{5.6}$$

Reflective boundary conditions were programmed using Algorithm 10. Masonry sand

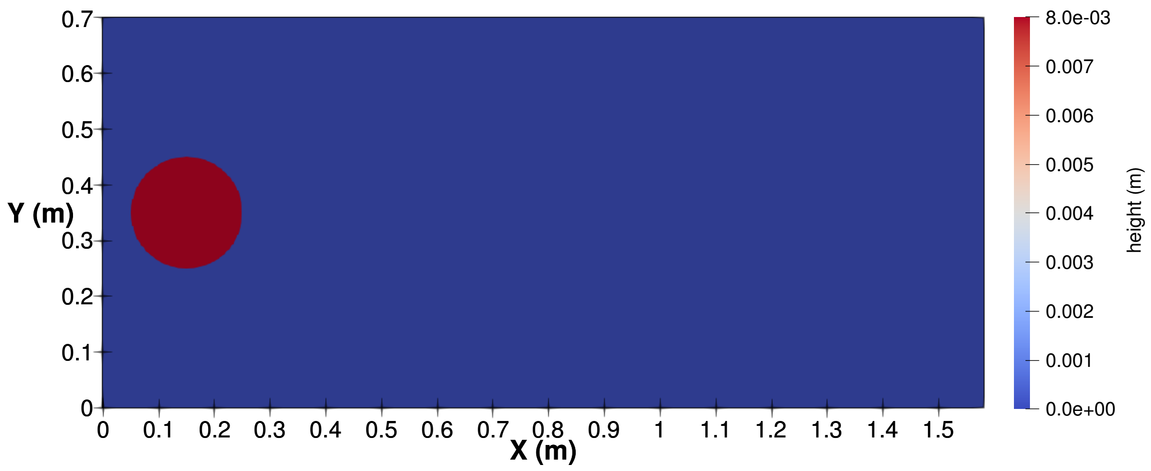


Figure 5.4: 2D-surface plot of granular material height at $t = 0\ s$.

was used as the granular material whose parameters are listed in Table 5.1.

From the surface plots in Figure 5.5, it can be seen that the material flows in the downslope direction ($x$-axis) due to the acceleration caused by gravity. The line plots for velocity ($\bar{u}$) in Figure 5.6 indicate that the velocity increases as the material propagates further in the downslope direction. Also, the material has the lowest velocity ($\bar{u}$) at the location where the material height is the highest. This can be attributed to the internal friction between

| $\theta_1$ | $\theta_2$ | $\theta_3$ | $L$ | $\beta$ | $\gamma$ | $\kappa$ | $\beta_*$ |
|---|---|---|---|---|---|---|---|
| $28.95°$ | $44.09°$ | $31.81°$ | $0.35\ mm$ | 1.07 | 2.01 | 1 | 0.06 |

Table 5.1: Experimentally determined friction law parameters for masonry sand of grain diameter $300 - 400\ \mu m$ [25].

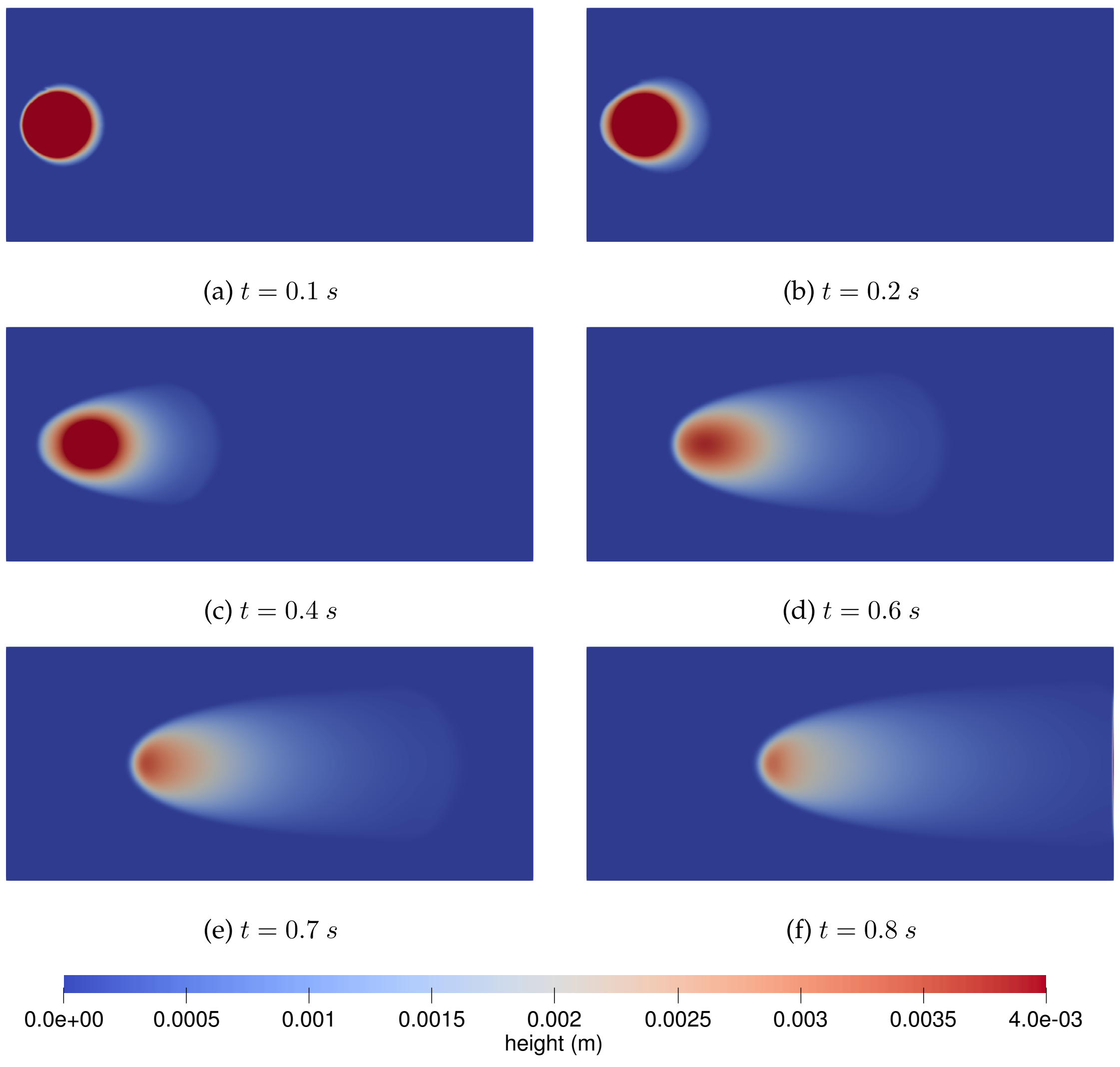
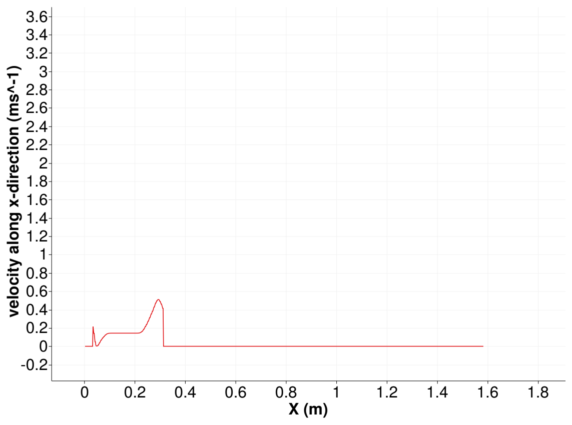
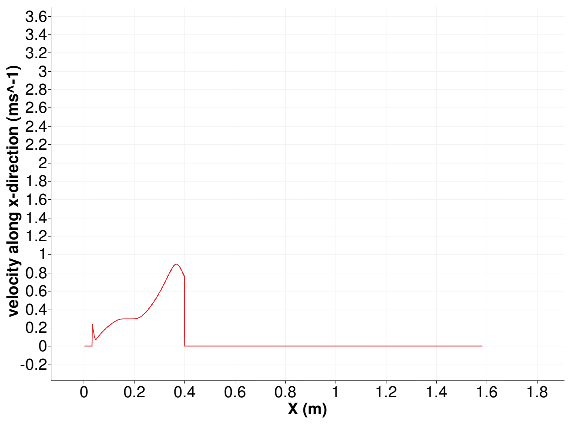


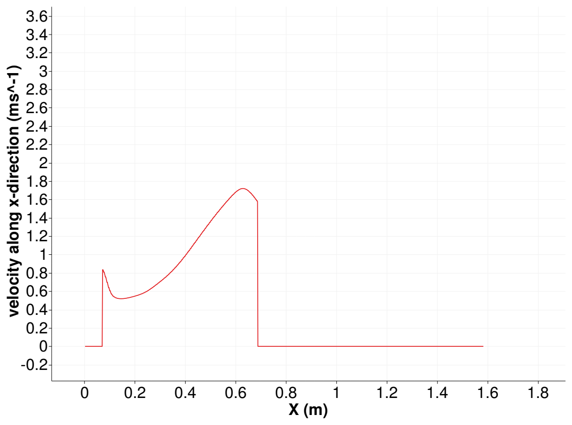
Figure 5.5: 2D-surface plot of granular material height (sand) at different points in time.



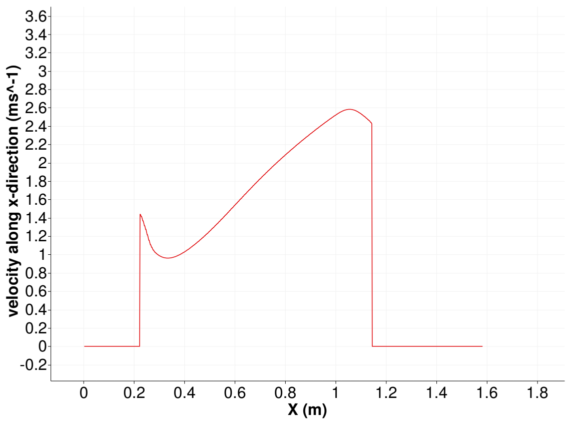
(a) $t = 0.1\ s$



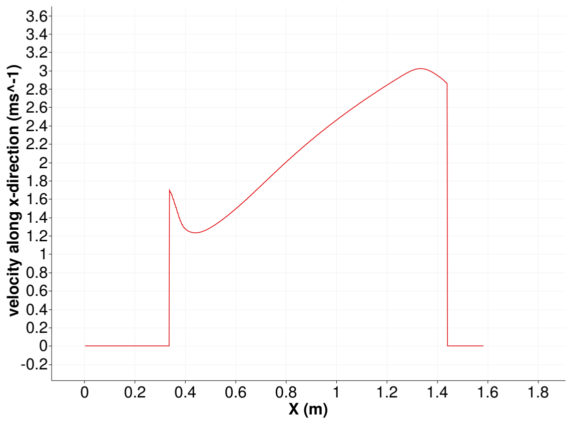
(b) $t = 0.2\ s$



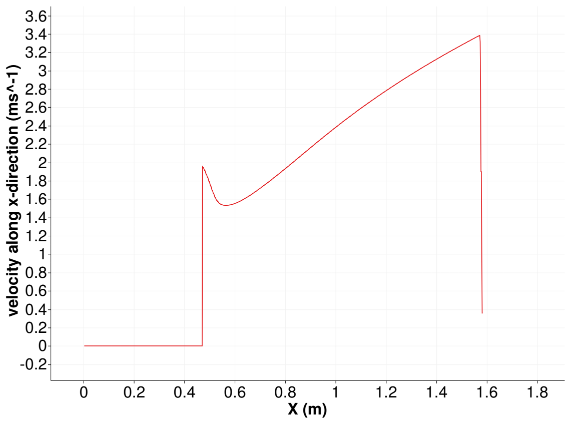
(c) $t = 0.4\ s$



(d) $t = 0.6\ s$



(e) $t = 0.7\ s$



(f) $t = 0.8\ s$

Figure 5.6: Line plot of $\bar{u}$ (velocity along $x$-direction) of sand along the centreline $y = 0.35$ $m$ at different points in time.

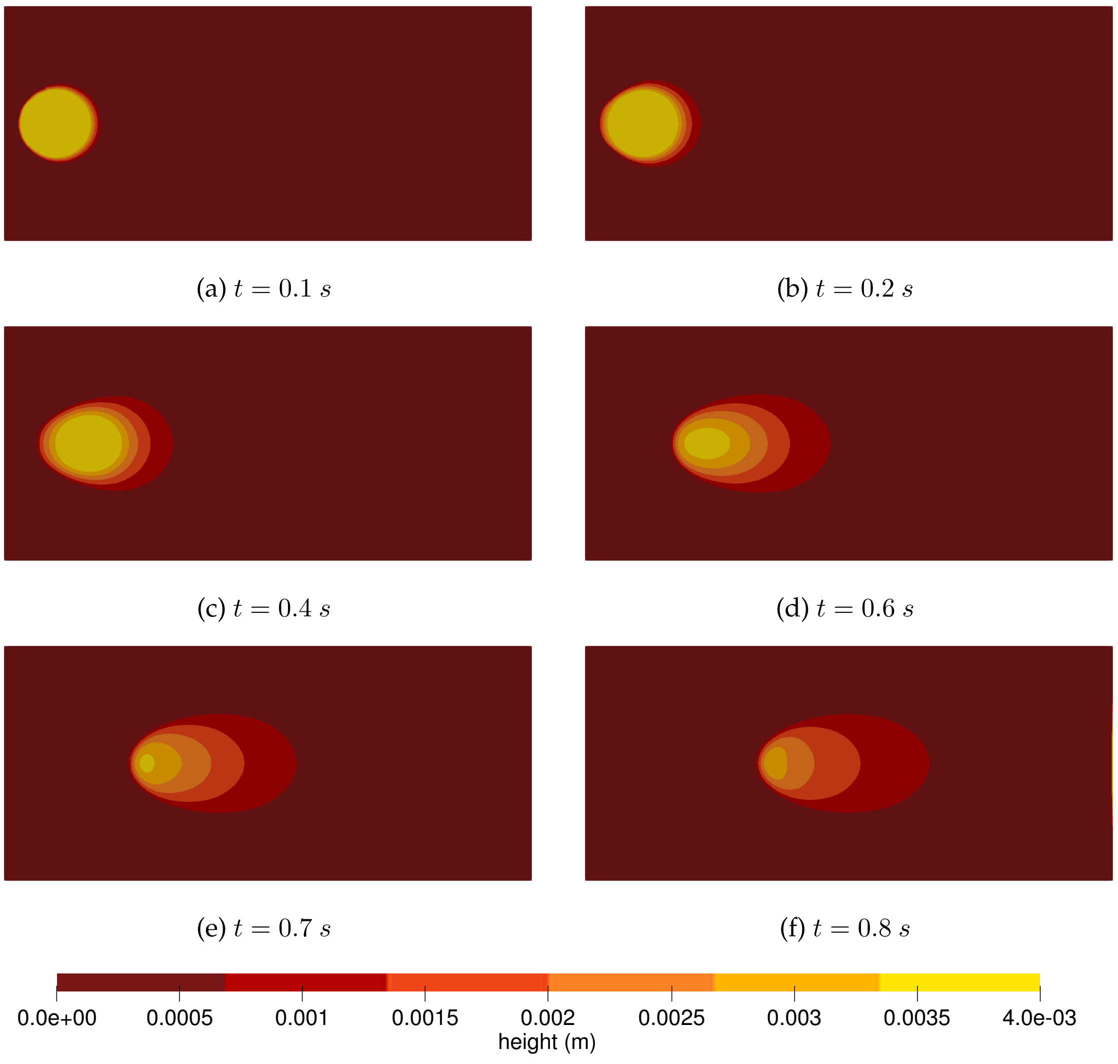


Figure 5.7: 2D-surface plot of granular material (sand) height plotted using a step colour palette to highlight crescentic contours at different points in time.

grains, which is highest in the location with the highest material density. Vice versa, we see that the peak velocity ($\bar{u}$) is obtained in the region which is furthest down the slope with a very low material height. In Figure 5.7, the same results from Figure 5.5 are shown using a step colour palette to highlight the contours. We see crescent-shaped contours for granular material with height decaying along the downslope direction. These contours are qualitatively similar to what was obtained by Pouliquen and Forterre [21] in their experiments and simulations. The contours obtained by them are shown in Figure 5.8. These observations confirm that the source term is indeed functioning correctly.

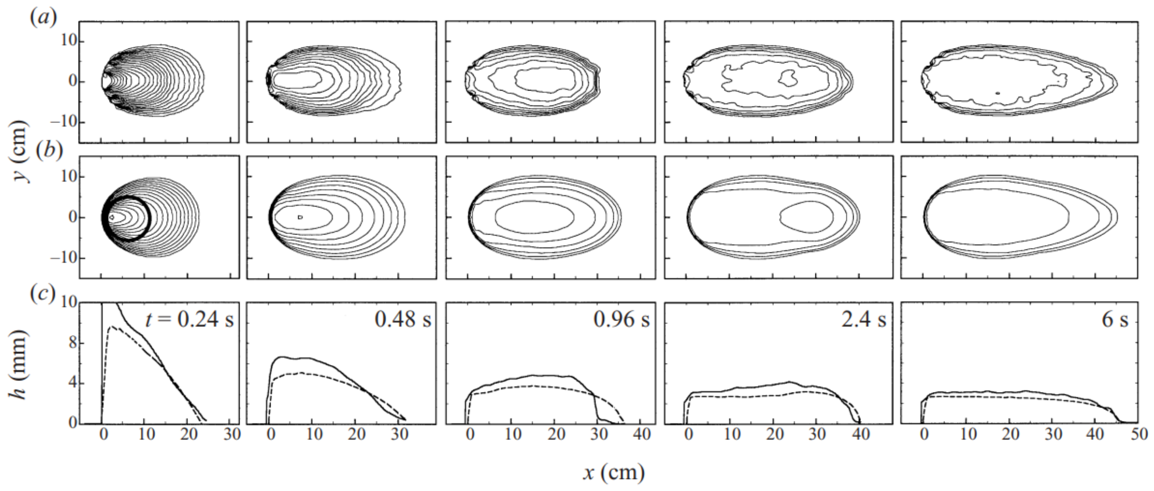


Figure 5.8: Contours of constant thickness every $0.5\ mm$ obtained from $(a)$ experiments, and $(b)$ simulations. $(c)$ Line plot of granular material height along the centreline; —, experiments; simulations, - - - [21]. The bold circle on the first plot of the simulation in $(b)$ represents the initial position of the granular material patch.

### 5.1.3 Simulation With Vanishing Friction

The simulation was repeated with the same parameters and conditions used in Section 5.1.2. The only change was that the basal friction coefficient $\mu$ was forced to a value of 0. By doing so, we compare the simulation with the one in Section 5.1.2 to check whether the friction force is indeed causing resistance to the flow. From the surface plots in Figure 5.9, it can be seen that the contours are circular rather than crescentic. This is expected, as it is the friction law which is responsible for the crescentic contours. In the absence of friction, the material spreads evenly in all directions, due to which we see circular contours. By making the frictional force vanish, it is expected that the granular material accelerates faster and has a higher velocity as compared to results obtained in Section 5.1.2. From the line plots of velocity ($\bar{u}$) in Figure 5.10, it is evident that the velocity without friction is always

| $\theta_1$ | $\theta_2$ | $\theta_3$ | $L$ | $\beta$ | $\gamma$ | $\kappa$ | $\beta_*$ |
|---|---|---|---|---|---|---|---|
| $31.1°$ | $47.5°$ | $32.7°$ | $0.44\ mm$ | 0.63 | 0.4 | 1 | 0.466 |

Table 5.2: Experimentally determined friction law parameters for carborundum particles of diameter $750 - 1000\ \mu m$ [6].

higher except in Figure 5.10f at the end of the domain. This exception is due to the reflective boundary conditions which try to oppose the flow at the end of the inclined plane, due to which the velocity of the material decreases.

### 5.1.4 Comparison Between Sand and Carborundum Granules

To check the dependence of the coefficient of basal friction ($\mu$) on the material parameters, the flow of carborundum particles was simulated and compared with the flow of sand particles from Section 5.1.2. The material parameters used for carborundum are given in Table 5.2. The simulation of carborundum was performed using the same domain, initial conditions and boundary conditions as used in Section 5.1.2. From the surface plots in Figure 5.11, it can be seen that we again obtain crescentic contours. This is expected since it is the friction law which is responsible for their occurrence. As discussed in Section 5.1.2, the particles in the densest region have the lowest velocity ($\bar{u}$), and the ones in the sparsest region furthest in the downslope direction have the highest velocity ($\bar{u}$). From the line plots of velocity ($\bar{u}$) in Figure 5.12, it can be observed that the minimum velocity ($\bar{u}$) of sand is higher than that of carborundum while the maximum velocity ($\bar{u}$) of sand is lower than that of carborundum. This implies that the sand particles in dense regions propagate faster than the carborundum particles in dense regions. In contrast, the particles in sparse regions of sand propagate slower than carborundum particles in sparse regions. This difference in acceleration behaviour illustrates the dependence of the coefficient of friction ($\mu$) on material parameters. Hence, our application qualitatively demonstrates this behaviour.

## 5.2 Simulation With Static Adaptive Mesh Refinement

In this section, the results with static adaptive mesh refinement (static AMR) are presented to show the effect of this feature. The simulation parameters are the same as the ones used in Section 5.1.2. The static AMR was applied using Algorithm 16, which refines the domain when $x < 0.79\ m$ and coarsens when $x >= 0.79\ m$. The `minimum_volume` is set in the code to perform coarsening up to three levels. Coarsened cells can be seen near the end of the domain in Figures 5.13e-5.13f. Figure 5.14 compares the velocity along the $x$-direction ($\bar{u}$) obtained with and without static AMR. From Figures 5.14c-5.14d, it can be seen that the velocity curves differ greatly as the particles with the highest velocity transition into the coarsened part of the domain. The simulation using static AMR was also attempted for carborundum particles, but the eigenvalues blew up after just two time steps. The static AMR for rectangular domains is under development in ExaHyPE 2, hence it does not always yield a solution in its current state of implementation.

**Algorithm 16:** Implementation of the refinement criterion function where one half of the domain is refined while the other half is coarsened

```
Input: $Q$: Vector containing unknown and auxiliary variables.
       $volumeCentre$: Vector containing coordinates of the centre of current
       volume with $\{0 : x, 1 : y\}$.
Output: $refinementCommand \in \{Keep, Refine, Erase\}$: Refinement option.
1 Function refinementCriterion($Q$, $volumeCentre$):
2     if $volumeCentre(0) < 0.79$ then
3         return $Refine$
4     else
5         return $Erase$
```

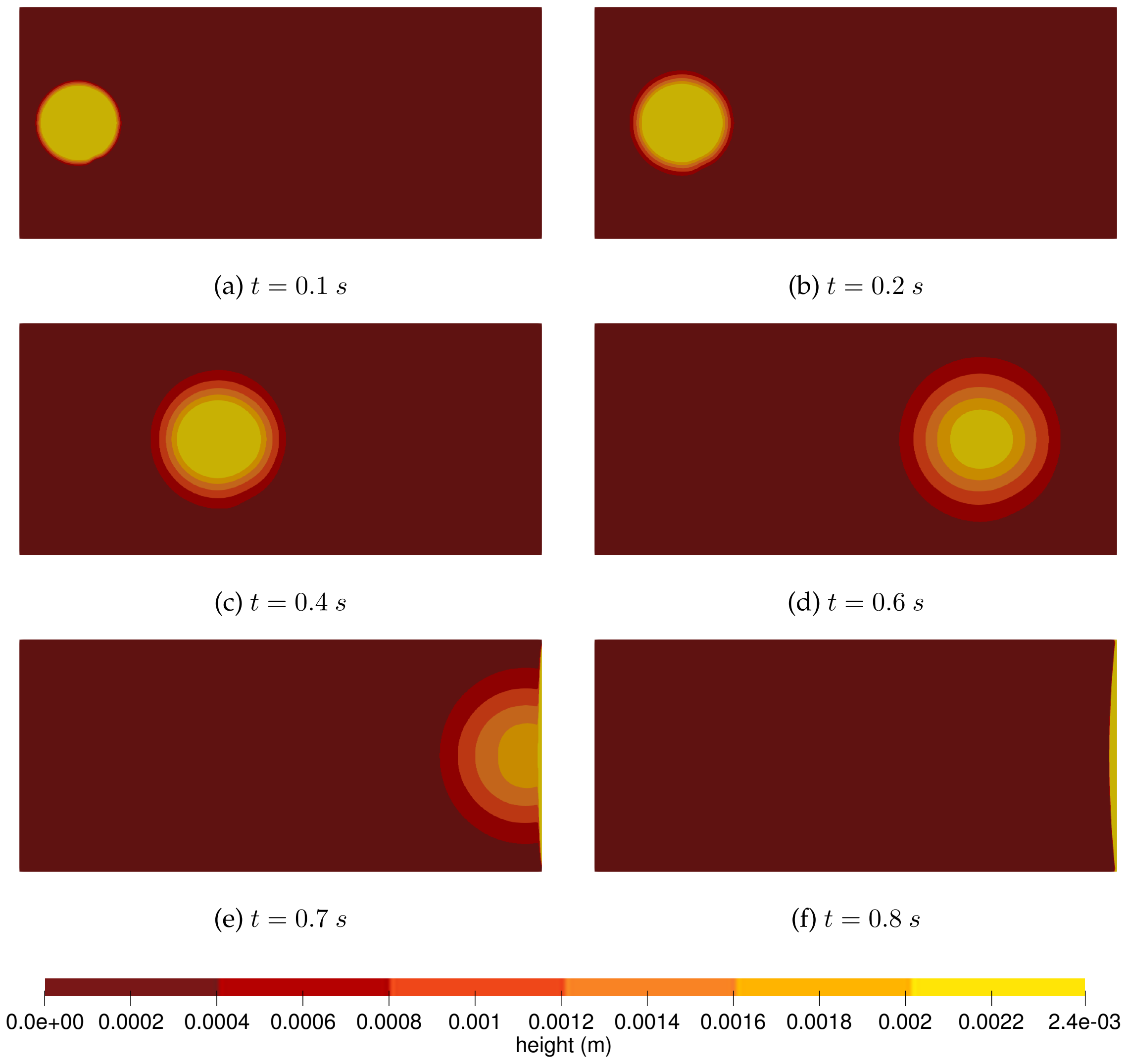
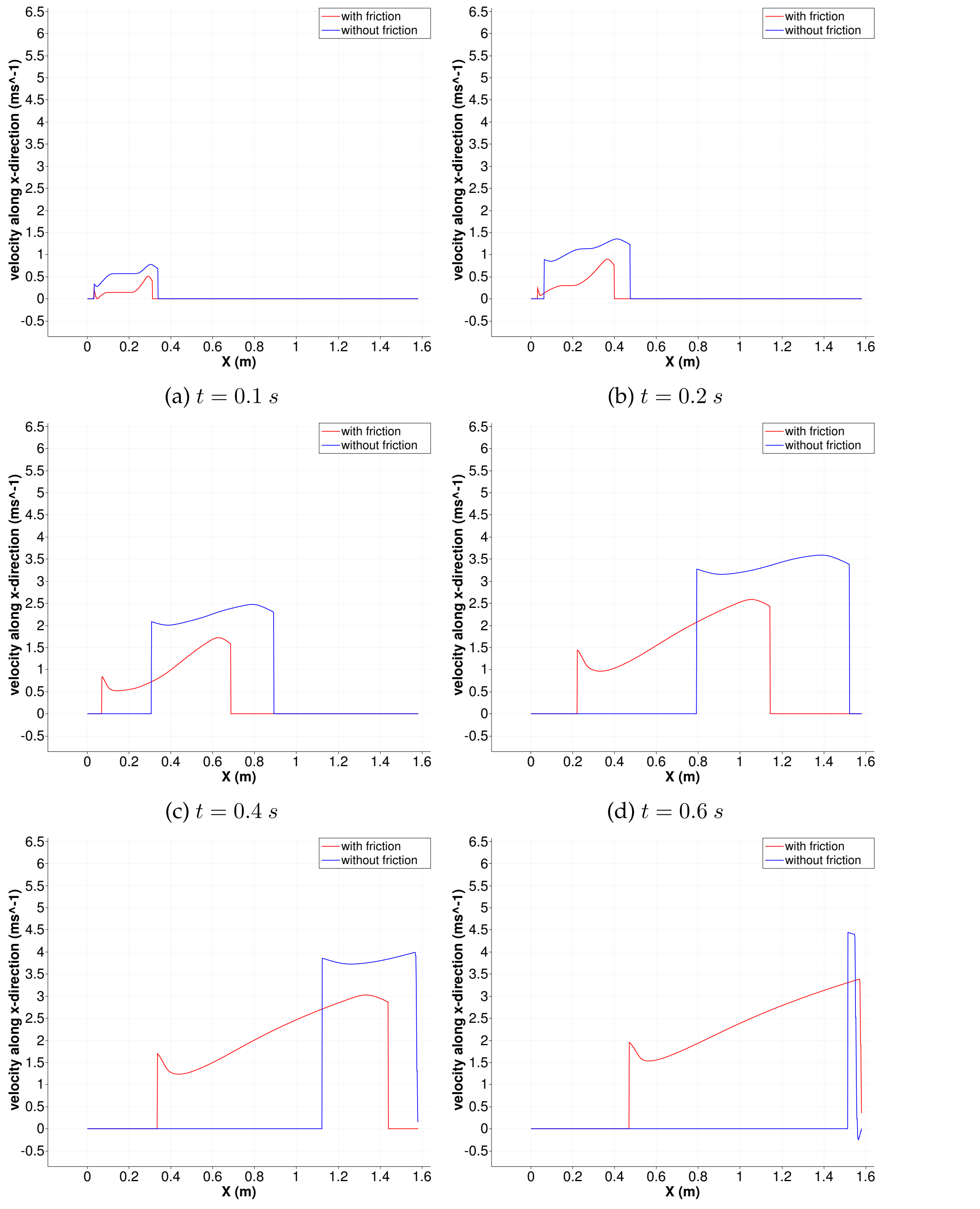


Figure 5.9: 2D-surface plot of granular material height (sand) on a frictionless plane, plotted using a step colour palette to highlight crescentic contours at different points in time.

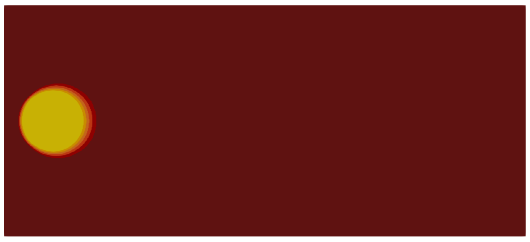


(a) $t = 0.1\ s$ (b) $t = 0.2\ s$

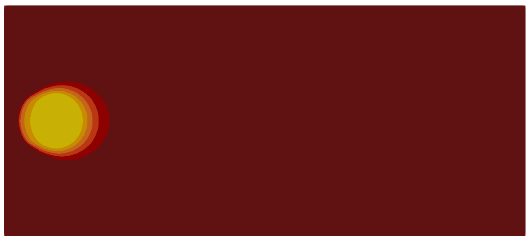
(c) $t = 0.4\ s$ (d) $t = 0.6\ s$

(e) $t = 0.7\ s$ (f) $t = 0.8\ s$

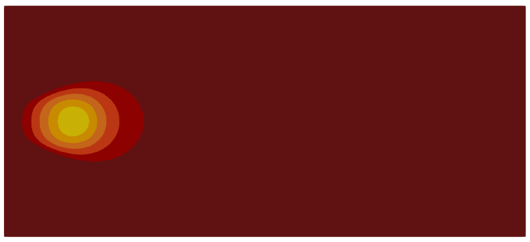
Figure 5.10: Line plot comparing $\bar{u}$ (velocity along $x$-direction) of sand with and without friction along the centreline $y = 0.35\ m$ at different points in time.



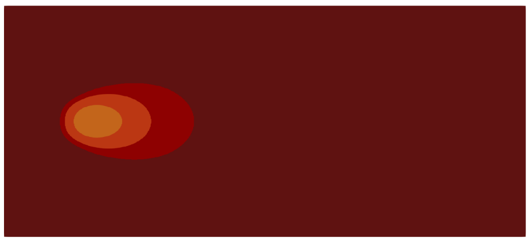
(a) $t = 0.1\ s$



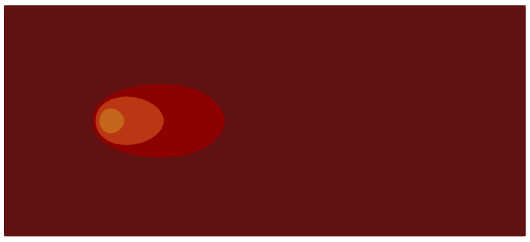
(b) $t = 0.2\ s$



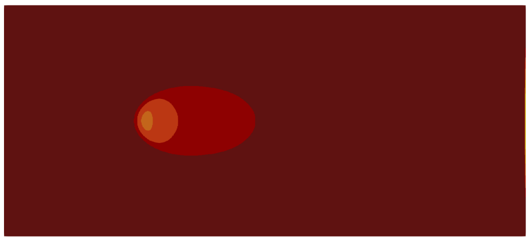
(c) $t = 0.4\ s$



(d) $t = 0.6\ s$



(e) $t = 0.7\ s$



(f) $t = 0.8\ s$

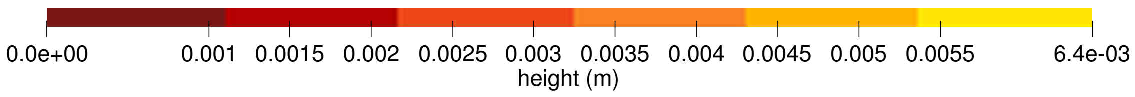
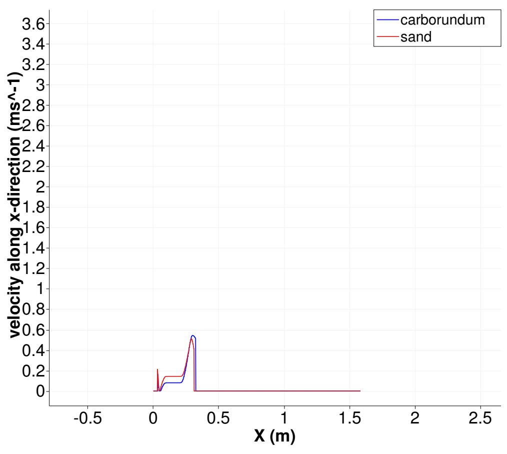
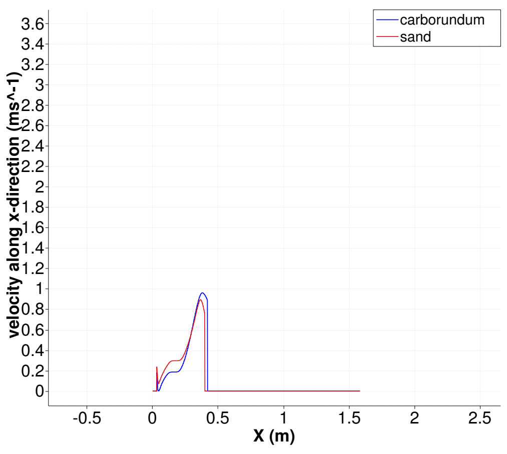


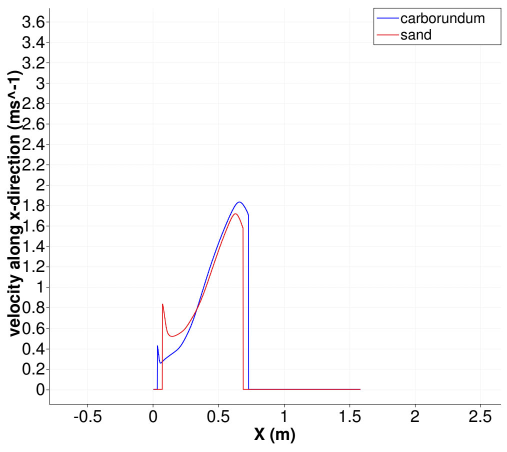
Figure 5.11: 2D-surface plot of granular material height (carborundum) plotted using a step colour palette to highlight crescentic contours at different points in time.



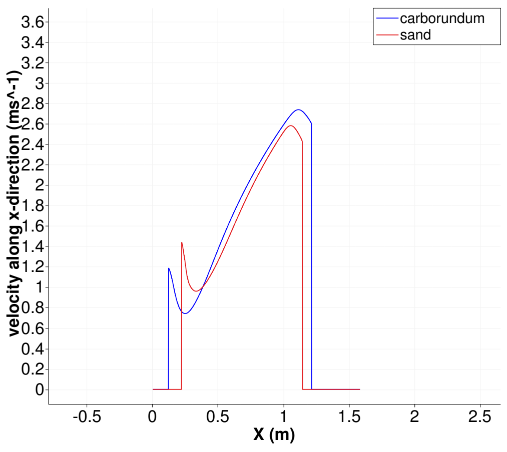
(a) $t = 0.1\ s$



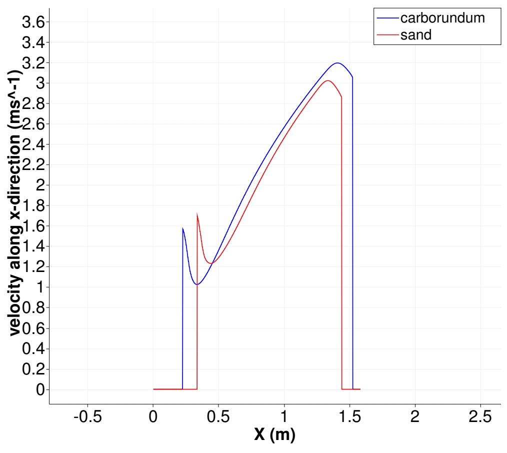
(b) $t = 0.2\ s$



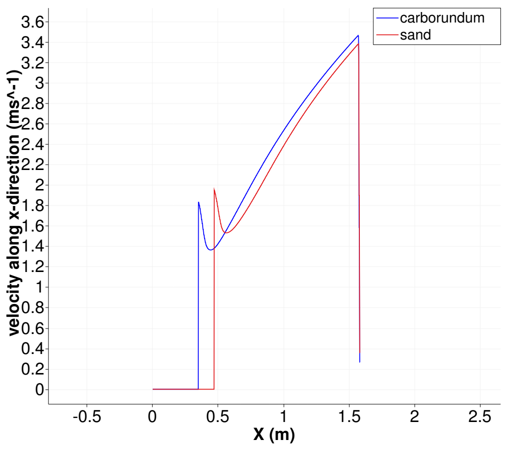
(c) $t = 0.4\ s$



(d) $t = 0.6\ s$



(e) $t = 0.7\ s$



(f) $t = 0.8\ s$

Figure 5.12: Line plot comparing $\bar{u}$ (velocity along $x$-direction) of sand particles and carborundum particles, along the centreline $y = 0.35\ m$ at different points in time.

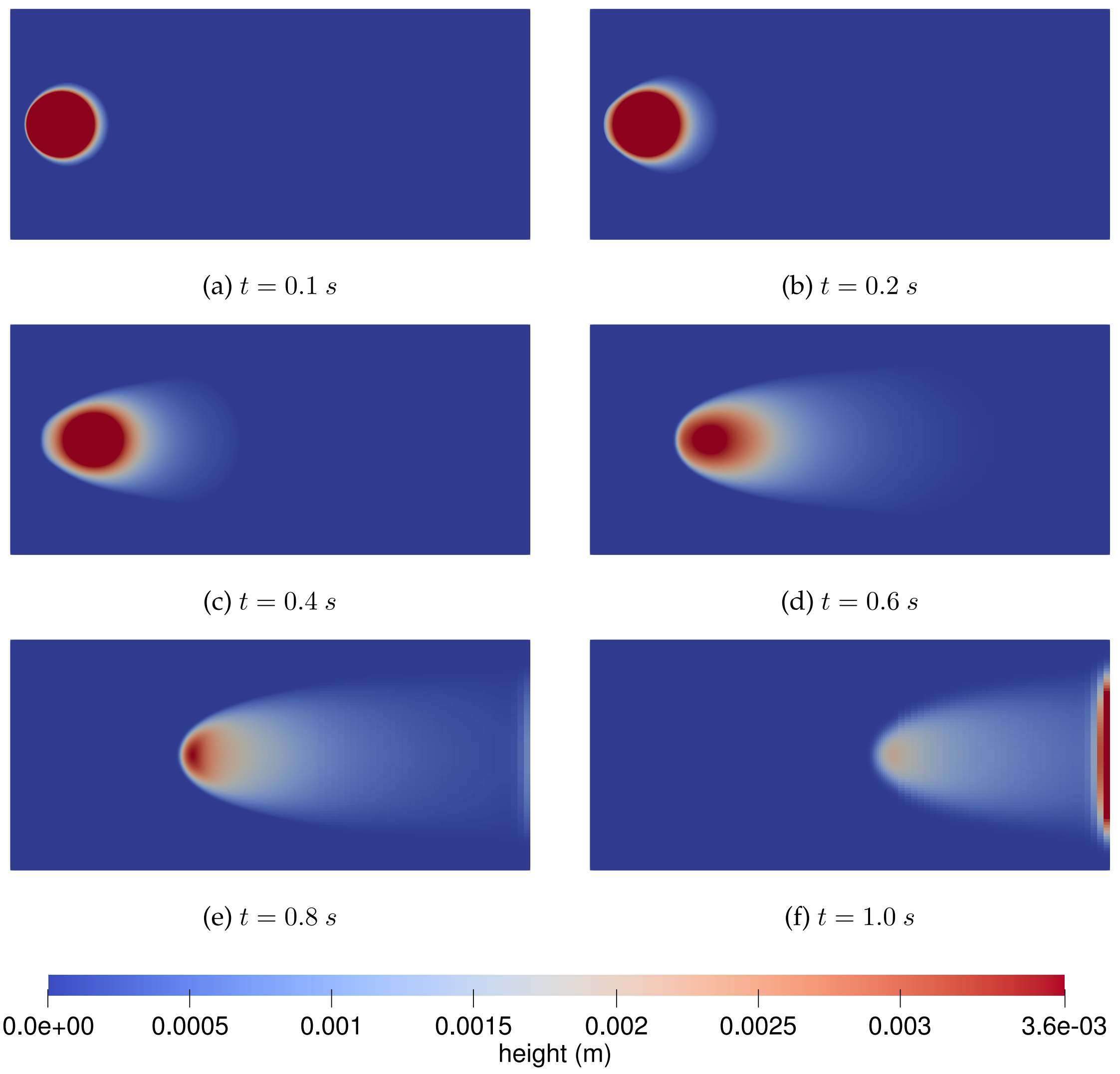
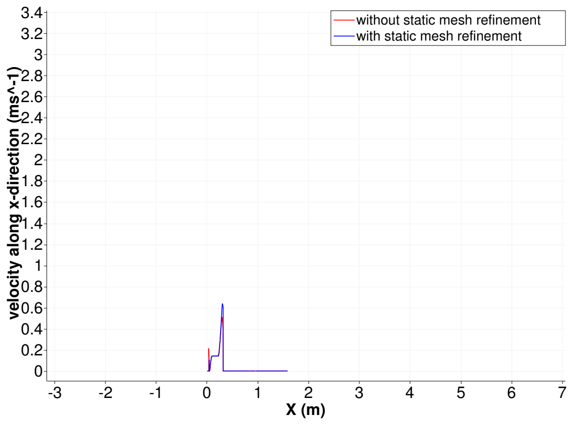
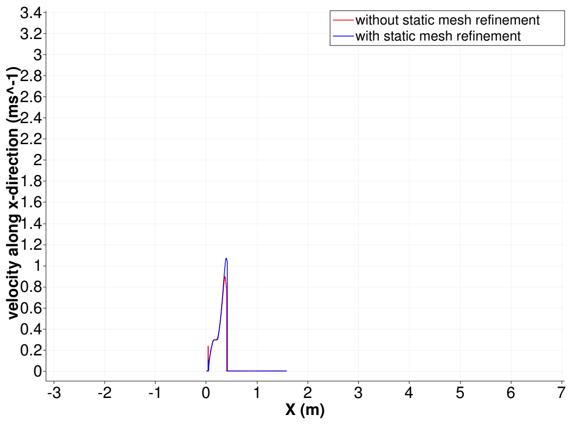


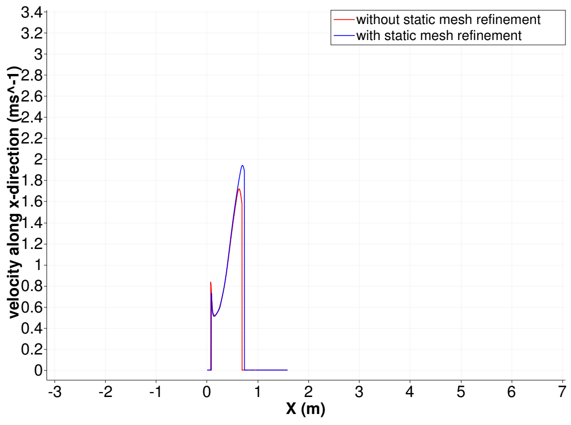
Figure 5.13: 2D-surface plot of granular material (sand) height, simulated using static AMR at different points in time.



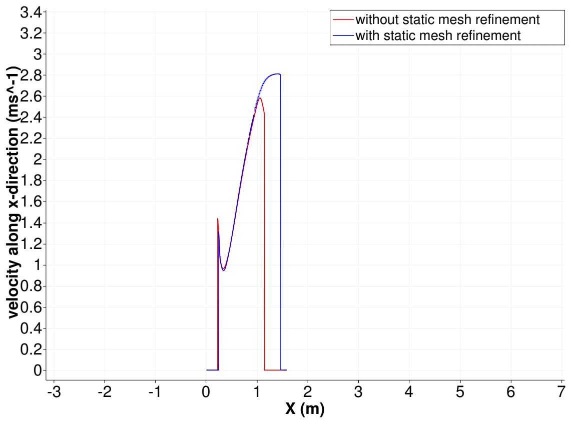
(a) $t = 0.1\ s$



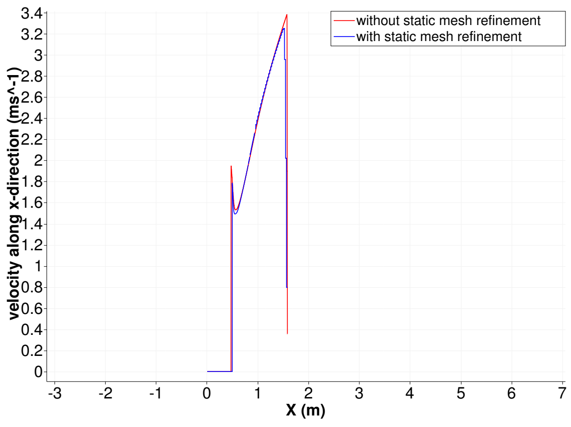
(b) $t = 0.2\ s$



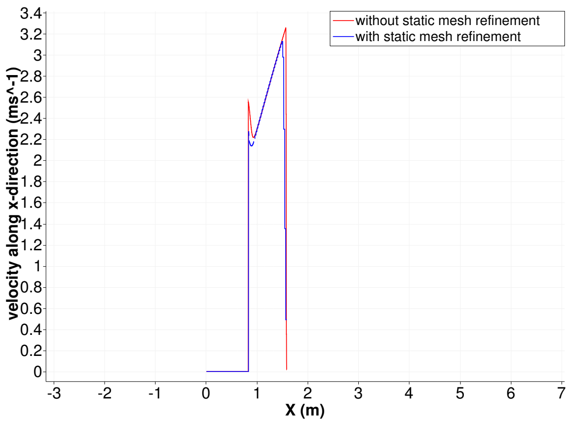
(c) $t = 0.4\ s$



(d) $t = 0.6\ s$



(e) $t = 0.8\ s$



(f) $t = 1.0\ s$

Figure 5.14: Line plot comparing $\bar{u}$ (velocity along $x$-direction) with and without static AMR, along the centreline $y = 0.35\ m$ at different points in time.

# 6 Conclusion and Discussion

In this thesis, the depth-averaged avalanche equations were implemented in ExaHyPE 2 to develop a solver application that simulates avalanches using the finite volume method with Rusanov flux and adaptive time stepping. The challenges caused by the complexity of the *non-conservative product* were resolved by applying a preprocessing routine on every patch before every time step. Several validation cases were also presented to ensure that the application simulates correct physics as per the governing equations.

The first validation case checks for convergence of the depth-avalanche equations to the shallow-water equations upon making the inclination, viscosity and the *source term* vanish. Initial conditions were applied to replicate a typical circular dam-break scenario at the centre of the domain, and reflective boundary conditions were applied at the boundaries. The results were then compared with the ones obtained from an already present shallow-water application in ExaHyPE 2 and were found to match.

In the second validation case, the flow of sand down a rectangular inclined plane was simulated. It was verified from the line plots of velocity that the material accelerates down the inclined plane because of the effect of the *source term*. The crescentic contours on the 2D-surface plots of granular material height qualitatively confirm that the friction law is implemented correctly.

The third validation case presents the simulation of sand on a frictionless inclined plane. The results were compared with those obtained in the second validation case (simulation with friction) to verify that friction is indeed causing the grains to decelerate and not the other way around.

In the fourth validation case, the granular material was changed from sand to carborundum, and the simulation was performed on the same domain as what was used for sand in the second validation case. Upon comparing the results of sand and carborundum, it was observed that both exhibit different behaviour, which is expected since the coefficient of friction depends on material parameters.

Finally, the static adaptive mesh refinement (static AMR) feature was demonstrated by refining one half and coarsening the other half of the inclined plane. The comparison of results obtained with and without static AMR was also presented. The static AMR feature functions correctly when sand is used as the granular material, but fails when carborundum granules are used. In the case of carborundum granules, the eigenvalues blow up, due to which the solver yields an incorrect solution. The static AMR for rectangular domains is under development in ExaHyPE 2.

The solver application can also be compiled to perform simulations using the Arbitrary Derivative Discontinuous Galerkin (ADER-DG) method. However, the ADER-DG method in its current state in ExaHyPE 2 does not allow the implementation of preprocessing routines. Hence, the depth-averaged avalanche equations cannot be simulated using this method.

The future work for this thesis would be to couple the avalanches with a water body to simulate tsunami events caused by avalanches.

# Bibliography

[1] B. Andreotti, Y. Forterre, and O. Pouliquen. *Granular media: between fluid and solid*. Cambridge University Press, 2013.

[2] B. Ataie-Ashtiani and S. Yavari-Ramshe. Numerical simulation of wave generated by landslide incidents in dam reservoirs. *Landslides*, 8:417–432, 12 2011.

[3] J. Baker, T. Barker, and J. Gray. A two-dimensional depth-averaged μ(i)-rheology for dense granular avalanches. *Journal of Fluid Mechanics*, 787:367–395, 01 2016.

[4] M. O. Bristeau, A. Mangeney, J. Sainte-Marie, and Seguin N. An energy-consistent depth-averaged euler system: derivation and properties, 2016.

[5] A.J.C.B. de Saint-Venant. *Théorie du mouvement non permanent des eaux, avec application aux crues des rivières et à l'introduction des marées dans leur lit*. Comptes rendus hebdomadaires des séances de l'Académie des sciences. Gauthier-Villars, 1871.

[6] A. N. Edwards, A. S. Russell, C. G. Johnson, and J. M. N. T. Gray. Frictional hysteresis and particle deposition in granular free-surface flows. *Journal of Fluid Mechanics*, 875:1058–1095, 2019.

[7] A. N. Edwards, S. Viroulet, B. P. Kokelaar, and J. M. N. T. Gray. Formation of levees, troughs and elevated channels by avalanches on erodible slopes. *Journal of Fluid Mechanics*, 823:278–315, 2017.

[8] A. N. Edwards and J. M. N. T. Gray. Erosion–deposition waves in shallow granular free-surface flows. *Journal of Fluid Mechanics*, 762:35–67, 2015.

[9] C. Escalante and T. M. de Luna. A general non-hydrostatic hyperbolic formulation for boussinesq dispersive shallow flows and its numerical approximation. *Journal of Scientific Computing*, 83(3):62, 2020.

[10] Y. Forterre and O. Pouliquen. Long-surface-wave instability in dense granular flows. *Journal of Fluid Mechanics*, 486:21–50, 2003.

[11] Y. Forterre and O. Pouliquen. Flows of dense granular media. *Annual Review of Fluid Mechanics*, 40(1):1–24, 2008.

[12] GDR MiDi. On dense granular flows. *Eur. Phys. J. E*, 14(4):341–365, 2004.

[13] J. M. N. T. Gray and A. N. Edwards. A depth-averaged $\mu(i)$-rheology for shallow granular free-surface flows. *Journal of Fluid Mechanics*, 755:503–534, 2014.

[14] F. Heslot, T. Baumberger, B. Perrin, B. Caroli, and C. Caroli. Creep, stick-slip, and dry-friction dynamics: Experiments and a heuristic model. *Physical review E*, 49(6):4973, 1994.

[15] P. Jop, Y. Forterre, and O. Pouliquen. Crucial role of sidewalls in granular surface flows: consequences for the rheology. *Journal of Fluid Mechanics*, 541:167–192, 2005.

[16] C. Longchamp, O. Caspar, M. Jaboyedoff, Y. Podladchikov, et al. Saint-venant equations and friction law for modelling self-channeling granular flows: from analogue to numerical simulation. *Applied Mathematics*, 6(07):1161, 2015.

[17] G. Ma, J. T. Kirby, T. J. Hsu, and F. Shi. A two-layer granular landslide model for tsunami wave generation: Theory and computation. *Ocean Modelling*, 93:40–55, 2015.

[18] S. Nasuno, A. Kudrolli, and J. P. Gollub. Friction in granular layers: Hysteresis and precursors. *Physical Review Letters*, 79(5):949, 1997.

[19] A. K. Patra, A. C. Bauer, C. C. Nichita, E. B. Pitman, M. F. Sheridan, M. Bursik, B. Rupp, A. Webber, A. J. Stinton, L. M. Namikawa, and C. S. Renschler. Parallel adaptive numerical simulation of dry avalanches over natural terrain. *Journal of Volcanology and Geothermal Research*, 139(1):1–21, 2005. Modeling and Simulation of Geophysical Mass Flows.

[20] O. Pouliquen. Scaling laws in granular flows down rough inclined planes. *Physics of fluids*, 11(3):542–548, 1999.

[21] O. Pouliquen and Y. Forterre. Friction law for dense granular flows: application to the motion of a mass down a rough inclined plane. *Journal of fluid mechanics*, 453:133–151, 2002.

[22] M. Rauter, S. Viroulet, S. Gylfadottir, W. Fellin, and F. Løvholt. Granular porous landslide tsunami modelling – the 2014 lake askja flank collapse. *Nature Communications*, 13, 02 2022.

[23] A. Reinarz, D. E. Charrier, M. Bader, L. Bovard, M. Dumbser, K. Duru, F. Fambri, A. A. Gabriel, J. M. Gallard, S. Köppel, L. Krenz, L. Rannabauer, L. Rezzolla, P. Samfass, M. Tavelli, and T. Weinzierl. Exahype: An engine for parallel dynamically adaptive simulations of wave problems. *Computer Physics Communications*, 254:107251, 2020.

[24] S. B. Savage and K. Hutter. The motion of a finite mass of granular material down a rough incline. *Journal of Fluid Mechanics*, 199:177–215, 1989.

[25] C. Tregaskis, C. G. Johnson, X. Cui, and J. M. N. T. Gray. Subcritical and supercritical granular flow around an obstacle on a rough inclined plane. *Journal of Fluid Mechanics*, 933:A25, 2022.

[26] T. Weinzierl. The peano software—parallel, automaton-based, dynamically adaptive grid traversals. *ACM Transactions on Mathematical Software*, 45(2):14, 2019.

# List of Figures

# List of Tables

# List of Algorithms