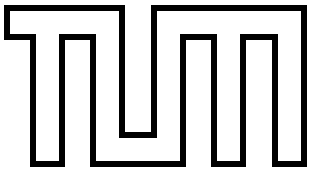


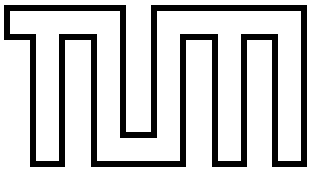
Computational Science and Engineering

Technical University of Munich

Master's Thesis

High-Order Landslide Simulation based on the Voellmy-Salm Friction Model with Stiff Source Term

Nikita Mashaev



Computational Science and Engineering

Technical University of Munich

Master's Thesis in Informatics

High-Order Landslide Simulation based on the Voellmy-Salm Friction Model with Stiff Source Term

Author:	Nikita Mashaev
Supervisor:	Univ.-Prof. Dr. Michael Bader
Advisor:	Mario Wille, M.Sc.
Date:	May 15th, 2025

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, May 15th, 2025 Nikita Mashaev

Acknowledgements

First and foremost, I would like to extend my sincere appreciation to Univ.-Prof. Dr. Michael Bader and my advisor, Mario Wille, for the opportunity to contribute to this research. Their expert guidance, thoughtful feedback, and encouragement were invaluable in shaping both the direction and quality of this work.

I am profoundly grateful to my family for their unwavering support, patience, and care. Their quiet strength and constant belief in me have been the foundation upon which this entire journey rests. In moments of doubt, the knowledge that they stood behind me gave me the resilience to continue moving forward.

I owe special thanks to my girlfriend, Irina Shishkina, for walking this path by my side. Her presence has been the brightest light through the darkest days, and always pointed me direction. It is through her warmth and care that I have found the strength to rise again and again.

I am also deeply thankful to my fellow students and close friends, Paul Rigor and Chia-Chian Chan, for their steadfast companionship. Their kindness, encouragement, and good humor have made this experience far richer and more meaningful than it would have been alone.

Abstract

Accurate numerical simulation of landslides and dry granular avalanches plays a critical role in understanding geophysical hazards and informing risk mitigation strategies. This work presents a high-order numerical framework for modeling such flows based on depth-averaged Shallow Water Equations (SWE) extended by the Voellmy-Salm friction model, which combines Coulomb-type basal resistance and velocity-dependent turbulent drag. The Arbitrary Derivative Discontinuous Galerkin (ADER-DG) method, implemented within the ExaHyPE 2 simulation engine, enables unified high-order discretization in space and time, while explicitly resolving stiff source terms and non-conservative topographic interactions.

Stability near steep gradients and transient wet-dry interfaces is ensured through a modified CFL criterion and an a-posteriori sub-cell Finite Volume limiter. In this approach, regions exhibiting non-smooth behavior are locally projected onto a first-order Finite Volume scheme using interface-based source treatment. Threshold-based limiting prevents nonphysical states, with $h_{\text{dry}} = 10^{-3}\,\text{m}$ for synthetic slopes and $h_{\text{dry}} = 10^{-1}\,\text{m}$ for realistic terrain.

The solver is validated on two representative scenarios. A benchmark case on a 35° inclined slope isolates the roles of frictional components, demonstrating physically consistent flow arrest under Coulomb friction and dissipative behavior under turbulent drag. A second test involving high-resolution digital elevation data from Norway's Tafjord region confirms the framework's robustness on complex topographies. Convergence studies indicate that sixth-order ADER-DG offers an optimal trade-off between accuracy and computational cost.

The results highlight the importance of tightly coupled numerical and physical modeling in granular flow simulations. Future extensions may focus on improved interface fluxes beyond the Rusanov scheme, advanced wetting-drying treatments, adaptive DMP criterion, and multiphase extensions to simulate landslide-generated tsunamis.

Keywords: granular flows, avalanche, landslide, Shallow Water Equations, ADER-DG ExaHyPE 2, Voellmy-Salm friction, sub-cell limiting, a-posteriori sub-cell limiter, stiff source, wetting-drying, geophysical simulation.

Contents

Part I.

Introduction and Background

1. Introduction

Gravity-driven geophysical hazards, such as landslides and snow avalanches involving dry granular flows, pose significant threats to human safety, infrastructure, and ecosystems. These natural phenomena involve the rapid downslope movement of soil, rock, or snow driven primarily by gravity. The primary goal of this thesis is to develop a high-order numerical framework within the ExaHyPE 2 computational engine to accurately simulate such granular flow events. The approach utilizes a depth-averaged continuum model governed by the Voellmy-Salm friction model, enabling detailed simulations that balance computational efficiency with accurate resolution of complex flow behaviors across varying terrains.

1.1. Motivation

The motivation behind studying granular flows in geophysical contexts arises from both practical requirements and existing theoretical limitations. Granular flows, such as avalanches and landslides, exhibit complex behaviors due to their simultaneous solid- and fluid-like properties, making traditional fluid dynamics approaches insufficient for capturing their dynamics fully. The Voellmy-Salm friction model effectively addresses this complexity by incorporating both Coulomb-type friction, which is independent of velocity, and velocity-squared turbulent-like drag, offering a practical and robust empirical description of granular flow behavior.

However, numerically implementing the Voellmy-Salm model poses significant challenges, including handling stiff source terms, non-conservative topographic interactions, and transient wet-dry interfaces. To address these issues, this thesis employs advanced numerical methods, particularly high-order Arbitrary Derivative Discontinuous Galerkin (ADER-DG) schemes. ADER-DG methods are well-suited to accurately resolving sharp gradients and discontinuities that frequently occur in granular flow simulations since they provide a unified space-time discretization order.

To enhance numerical stability and reliability, adaptive limiting strategies are integrated into the framework. Specifically, an a-posteriori sub-cell Finite Volume limiting approach ensures stable and accurate computations, especially in regions of sharp gradients or low flow depth. This adaptive method significantly improves the capability of the model to handle realistic scenarios encountered in practical hazard assessment applications.

By combining advanced theoretical modeling, robust numerical methods, and high-performance computing capabilities, this research aims to establish a scalable computational framework capable of simulating granular flow phenomena under realistic conditions. Ultimately, the developed framework seeks to inform effective hazard mitigation strategies, contribute to resilient infrastructure design, and enhance fundamental understanding of granular material behavior in geophysical contexts.

1.2. Related Work

Modeling of avalanche and granular flow dynamics has undergone substantial development, motivated by the need to assess hazards in mountainous regions and inform protective infrastructure planning. Among the empirical rheological models, the Voellmy-Salm (VS) formulation remains one of the most widely adopted frameworks in operational forecasting. Originally introduced by Voellmy [1] and later refined by Salm [2], [3], the model combines Coulomb-type basal friction with a velocity-squared turbulent drag term. This formulation is embedded in official guidelines for avalanche run-out estimation in Alpine countries and has become a foundation for numerous applied and theoretical studies.

Despite its widespread use, the Voellmy-Salm model introduces considerable numerical challenges, particularly when applied to granular flows over steep or irregular topographies. The turbulent drag friction terms appear to be stiff, resulting in difficulties for traditional explicit methods. A common approach to address stiffness is the use of operator-splitting schemes, where hyperbolic advection and source term contributions are computed in separate stages. However, as highlighted by LeVeque and Yee [4], such decoupling can lead to artifacts in the numerical solution. In granular flows, especially those driven by gravity, applying resistive forces after the advection step may produce nonphysical acceleration, overpredicted run-out distances, and front oscillations near terrain discontinuities.

To overcome these limitations, recent advances in high-order numerical methods have aimed to more accurately resolve stiff source terms while preserving spatial and temporal accuracy. The ADER-DG scheme [5], based on a fully coupled space-time predictor-corrector formulation, achieves arbitrary high-order accuracy and is capable of incorporating complex source terms. A key feature of the ADER-DG framework is the use of an a-posteriori sub-cell Finite Volume limiter [6], which ensures robustness in regions of non-smooth behavior. When the high-order DG polynomial fails to satisfy admissibility criteria - such as positivity preservation or Discrete Maximum Principles (DMP) - it is discarded and replaced by a lower-order Finite Volume (FV) update on a refined sub-grid within the element. This mechanism enables stable simulation of flows with steep gradients, shocks, and wet-dry transitions.

However, the effectiveness of this limiting strategy depends critically on the accuracy and stability of the fallback FV solver. In the presence of stiff source terms, the FV method must resolve their effects reliably. To this end, the present work employs an interface-based source discretization inspired by the upwinding approach of Bouchut et al. [7], in which friction terms are applied at cell interfaces rather than at cell centers. This design improves coupling between source and flux terms, enhances robustness near terrain discontinuities, and ensures consistency with the high-order ADER-DG scheme.

The present study builds on these developments by embedding a Voellmy-Salm model into a high-order ADER-DG framework within the ExaHyPE 2 simulation engine. Key contributions include an adjusted time-step restriction that accounts for stiff source terms and an integrated hybrid DG-FV limiting strategy tailored for granular flows with wet/dry interfaces. These techniques allow for explicit time-stepping, maintain the balance between frictional forces and terrain-induced gradients, and address long-standing numerical challenges in predictive avalanche modeling.

1.3. Structure of the Thesis

This thesis is structured to gradually build up the theoretical, numerical, and implementation framework required to simulate gravity-driven granular flows using high-order ADER-DG methods. The document is organized into seven chapters.

The following Chapter 2 establishes the physical and mathematical basis of the study. It begins with an overview of the Shallow Water Equations, which form the foundation of the depth-averaged formulation used to model landslides. The chapter further elaborates on the representation of friction through source terms, based on the Voellmy-Salm rheology. Physical interpretations of Coulomb basal friction and turbulent drag are discussed in detail, and the final form of the governing equations used throughout the thesis is presented.

Chapter 3 introduces the ADER-DG numerical method. It outlines the formulation in one dimension, followed by its extension to multiple spatial dimensions. Key algorithmic components are described, namely the predictor, corrector, and adaptive time-stepping procedures. Particular attention is given to the weak form discretization and its role in ensuring high-order accuracy for both space and time.

Chapter 4 discusses the main challenges encountered when applying ADER-DG methods to granular landslide simulations. The chapter focuses on a-posteriori sub-cell limiting techniques used to handle solution discontinuities. It describes the projection between ADER-DG and FV representations and introduces additional numerical adaptations required for handling stiff source terms. These include modifications to the time-step criterion and the redistribution of source term evaluation at cell interfaces for the limiter solver.

Chapter 5 provides an overview of the solver configuration and implementation using the ExaHyPE 2 simulation engine. It presents the workflow for setting up and generating solvers via the Python API, detailing the specific adaptations required for landslide modeling. The structure and logic of the ADER-DG and Finite Volume solvers are described, along with the implementation of a-posteriori sub-cell limiting.

Chapter 6 presents numerical results. The first part focuses on a synthetic benchmark case - a granular column collapse on an inclined slope. This setup allows controlled analysis of the physical effects of critical parameters and verifies the correctness of the model through grid and order convergence studies. The second part applies the solver to a realistic scenario based on the topography of the Tafjord region in Norway. Here, the numerical framework is tested under field-scale conditions, and the impact of individual friction terms is analyzed through comparative simulations.

Chapter 7 concludes the thesis by summarizing the main findings and outlining directions for future research.

2. Avalanche and Landslide Modeling Framework

In this work, the terms avalanche and landslide are used to describe the rapid downslope movement of granular material (such as soil or rock) under the influence of gravity. While avalanche traditionally refers to snow or ice flows, here the concept is generalized to encompass landslides - geophysical phenomena involving the collapse and flow of terrestrial granular masses. The material is treated as a continuum that behaves similarly to an incompressible fluid. This approximation assumes that the bulk material density remains constant over time and space, and that the internal structure of the granular flow does not significantly affect the macroscopic flow behavior.

Consequently, the physical model used to simulate avalanches is based on the **S**hallow **W**ater **E**quations (**SWE**), extended to incorporate the Voellmy-Salm friction model, where the friction resistance of the granular material is represented as a source term. The Voellmy-Salm model accounts for both velocity-independent Coulomb friction and turbulent-like drag, which are critical for accurately simulating granular flows. This section provides a detailed overview of the mathematical formulation, along with the main assumptions necessary for the model to be applicable.

2.1. The Shallow Water Equations

The Shallow Water Equations (SWE) represent a depth-averaged simplification of the Navier-Stokes equations, particularly effective for flows where the vertical scale (depth) is much smaller than horizontal dimensions as depicted in Figure 2.1. These equations were first introduced by Saint-Venant in 1871 [8] and retain the essential dynamics of wave propagation while significantly reducing computational complexity.

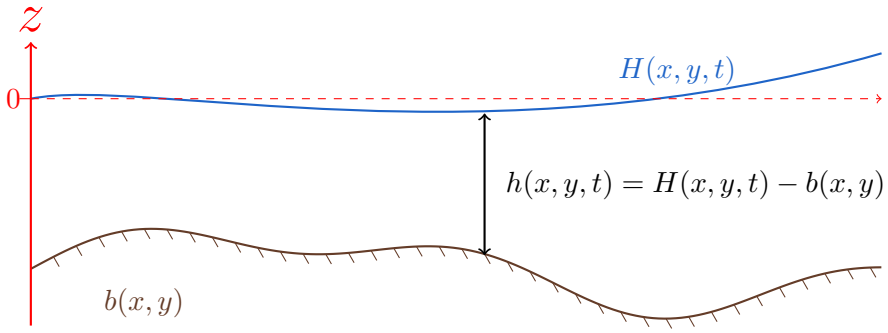
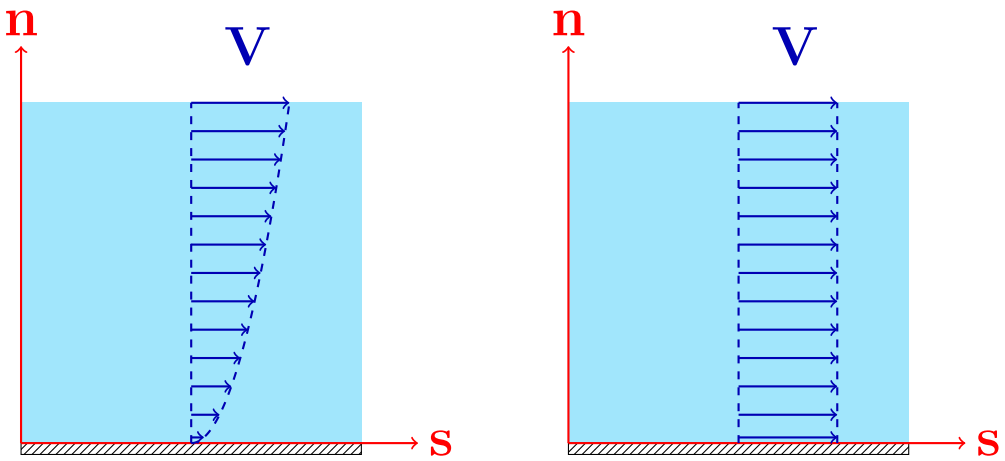


Figure 2.1.: Schematic representation of the flow domain in the SWE model. The free surface is denoted by $H(x, y, t)$, the bottom topography by $b(x, y)$, and the flow thickness by $h(x, y, t) = H(x, y, t) - b(x, y)$.

2.1.1. Key Assumptions

The derivation of SWE relies on several critical assumptions.

1. **Inviscid Flow:** Viscous effects are neglected, eliminating the viscous term from the momentum equations:
$$\nabla \cdot (\mu \nabla \mathbf{V}) \to 0, \tag{2.1}$$
where μ is dynamic viscosity coefficient and $\mathbf{V} = (u, v, w)^\top$ is the velocity vector. This simplification reduces the spatial order of the system and permits the use of free-slip boundary conditions (see Figure 2.2b and Figure 2.2a).



(a) The tangential velocity at the boundary is zero. The velocity profile exhibits a typical boundary layer with strong shear near the wall.

(b) Only the normal component of velocity is zero at the boundary. The tangential velocity remains unconstrained at the boundary.

Figure 2.2.: Comparison between viscous and inviscid flows.

2. **Long-Wave Approximation:** The horizontal scale L is much larger than the vertical scale (the flow depth) h:
$$\frac{h}{L} \ll 1, \tag{2.2}$$
implying negligible vertical acceleration:
$$\frac{dw}{dt} = \frac{\partial w}{\partial t} + \frac{\partial w}{\partial x}u + \frac{\partial w}{\partial y}v + \frac{\partial w}{\partial z}w \approx 0, \tag{2.3}$$
and ensuring the free surface remains nearly horizontal.
3. **Incompressibility:** Assuming constant fluid density simplifies the continuity equation to:
$$\nabla \cdot \mathbf{V} = 0. \tag{2.4}$$
This allows the density to be treated as a constant and reduces the number of unknown variables in the system.

2.1.2. Boundary Conditions for Vertical Axis

In the derivation of SWE, certain boundary conditions are imposed along the vertical direction shown in Figure 2.1.

- **Free surface** $z = H(x, y, t)$**:** The kinematic condition ensures that fluid particles remain on the surface

$$V_n|_{z=H(x,y,t)} = \mathbf{n} \cdot \mathbf{V}|_{z=H(x,y,t)} = \frac{dH}{dt} = \frac{\partial H}{\partial t} + u\frac{\partial H}{\partial x} + v\frac{\partial H}{\partial y} + w\frac{\partial H}{\partial z}, \tag{2.5}$$

  while the dynamic condition sets the pressure equal to external (typically atmospheric) pressure:

$$p|_{z=H(x,y,t)} = p_{\text{external}}. \tag{2.6}$$

  In this work, the external pressure p_{external} is assumed to be zero.

- **Bottom** $z = b(x, y)$**:** The inviscid flow assumption (2.1) allows a free-slip boundary condition to be applied

$$V_n|_{z=b(x,y)} = \mathbf{n} \cdot \mathbf{V}|_{z=b(x,y)} = \frac{db}{dt} = u\frac{\partial b}{\partial x} + v\frac{\partial b}{\partial y} + w\frac{\partial b}{\partial z}, \tag{2.7}$$

  assuming a stationary bed $\dfrac{\partial b}{\partial t} = 0$.

2.1.3. Depth-Averaged Formulation and Derivation Framework

A detailed and modern derivation of the SWE from the depth-averaged Euler system is provided in [9]. Applying the depth-averaging operator:

$$\langle\phi\rangle(x, y, t) = \frac{1}{H - b}\int_b^H \phi(x, y, z, t)dz = \frac{1}{h}\int_b^H \phi(x, y, z, t)dz, \tag{2.8}$$

to the Navier-Stokes equations, together with assumptions (2.1), (2.2), and (2.4), and using the Leibniz integral rule along with (2.5), (2.6) and (2.7), yields the SWE in conservative form:

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (h\langle u\rangle)}{\partial x} + \dfrac{\partial (h\langle v\rangle)}{\partial y} = 0 \\[2ex] \dfrac{\partial (h\langle u\rangle)}{\partial t} + \dfrac{\partial}{\partial x}\left(h\langle u\rangle^2 + \dfrac{gh^2}{2}\right) + \dfrac{\partial (h\langle u\rangle\langle v\rangle)}{\partial y} = -gh\dfrac{\partial b}{\partial x} \\[2ex] \dfrac{\partial (h\langle v\rangle)}{\partial t} + \dfrac{\partial (h\langle v\rangle\langle u\rangle)}{\partial x} + \dfrac{\partial}{\partial y}\left(h\langle v\rangle^2 + \dfrac{gh^2}{2}\right) = -gh\dfrac{\partial b}{\partial y}. \end{cases} \tag{2.9}$$

Here, $h = (x, y, t) = H(x, y, t) - b(x, y)$ is the flow thickness, $H(x, y, t)$ denotes the free surface, $b(x, y)$ is the bottom topography (bathymetry), and $\langle u\rangle$, $\langle v\rangle$ are the depth-averaged velocity components in the x- and y- directions. The term gh^2 represents the hydrostatic

pressure term, arising from the assumption of negligible vertical acceleration (2.3), with the vertical pressure distribution approximated by $p(z) = g(H - z)$.

For simplicity, in this work the variables u and v will be used to denote the depth-averaged velocities, i.e., $u = \langle u \rangle$, $v = \langle v \rangle$. With this notation, the SWE system can be written as:

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (hu)}{\partial x} + \dfrac{\partial (hv)}{\partial y} = 0 \\ \dfrac{\partial (hu)}{\partial t} + \dfrac{\partial}{\partial x}\left(hu^2 + \dfrac{gh^2}{2}\right) + \dfrac{\partial (huv)}{\partial y} = -gh\dfrac{\partial b}{\partial x} \\ \dfrac{\partial (hv)}{\partial t} + \dfrac{\partial (hvu)}{\partial x} + \dfrac{\partial}{\partial y}\left(hv^2 + \dfrac{gh^2}{2}\right) = -gh\dfrac{\partial b}{\partial y}. \end{cases} \tag{2.10}$$

2.2. Friction Model

The Voellmy-Salm rheological law, initially developed by Voellmy (1955) [1] and later extended by Salm (1968) [2] for dense granular flows, is widely adopted in avalanche and landslide modeling. This semi-empirical framework decomposes the basal shear stress τ into two physically distinct components:

$$\tau = \underbrace{\mu\sigma}_{\text{Coulomb friction}} + \underbrace{\frac{\rho g V^2}{\xi}}_{\text{Turbulent-like drag}}, \tag{2.11}$$

where:

- σ is the normal stress at the flow base;
- μ is the basal friction coefficient (distinct from the dynamic viscosity μ in (2.1));
- ρ is the bulk density of the granular material;
- g is the gravitational acceleration;
- ξ is the Voellmy turbulence coefficient $[m/s^2]$;
- $\mathbf{V}$ is the velocity vector, with $V^2 = \mathbf{V}^\top \cdot \mathbf{V}$.

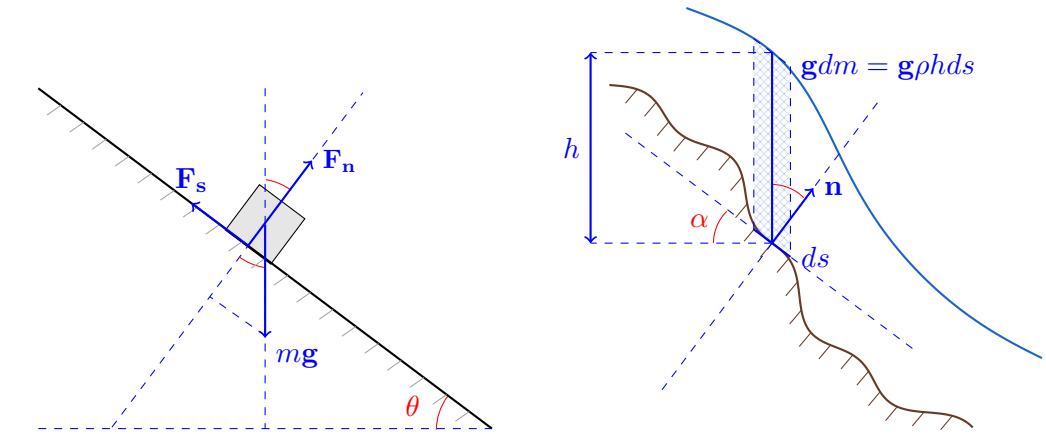
The Voellmy-Salm model combines two contributions: a Coulomb-type friction term that is independent of velocity, and a velocity-squared resistance term that accounts for turbulence-like effects. These components are discussed in more detail below.

2.2.1. Physical Interpretation: Coulomb Friction

The Voellmy-Salm model (2.11) synergistically combines the Coulomb-type dry friction and the velocity-squared turbulent drag. The Coulomb term represents the quasi-static friction at the flow base, where the granular material interacts with the underlying topography. This component is proportional to the normal stress σ and independent of flow velocity. It is expressed as:

$$\tau_{\text{Coulomb}} = \mu\sigma, \tag{2.12}$$

where $\mu = \tan(\theta)$, with θ being the material-dependent basal friction angle.



(a) Analogy for Coulomb friction: A rigid block on an inclined plane. Sliding initiates when $\tan(\theta) \geq \mu$, where θ is the slope angle.

(b) Normal stress σ depends on the slope-normal component of gravitational force $\rho g h \cos(\alpha)$, where α is the local slope angle.

Figure 2.3.: Mechanisms of Coulomb friction in granular flows.

The Coulomb term characterizes the threshold force required to initiate sliding. A classic analogy (Figure 2.3a) considers a rigid block on an inclined plane: sliding begins when the downslope component of gravitational force exceeds the frictional resistance. Assuming m is the block's mass, g is gravitational acceleration, and F_n is the normal force, the onset of motion occurs when:

$$mg\sin(\theta) \geq \mu F_n = mg\cos(\theta) \quad \Rightarrow \quad \mu = \tan(\theta_{\text{crit}}), \tag{2.13}$$

where θ_{crit} is the critical slope angle. This relation forms the basis for experimental determination of μ using tilt tests or shear-cell measurements. For brevity, we will omit the subscript in subsequent sections and denote the critical angle simply as θ. Thus, the friction coefficient is defined as

$$\mu = \tan(\theta). \tag{2.14}$$

The normal stress σ is determined by the slope-normal component of gravitational force (Figure 2.3b):

$$\sigma = \rho g h \cos(\alpha), \tag{2.15}$$

where α is the local slope angle. Substituting (2.15) into (2.12) yields:

$$\tau_{\text{Coulomb}} = \mu \rho g h \cos(\alpha). \tag{2.16}$$

2.2.2. Physical Interpretation: Velocity-Squared Turbulent Drag

The second term in the Voellmy model (2.11) represents a velocity-dependent resistance, often interpreted as a turbulent drag component. Mathematically, it is expressed as:

$$\tau_{\text{vd}} = \frac{\rho g \mathbf{V}^\top \cdot \mathbf{V}}{\xi} = \frac{\rho g V^2}{\xi}, \tag{2.17}$$

where ξ (units: m/s^2) is the Voellmy turbulence coefficient, which is determined experimentally, inversely related to basal roughness. While the quadratic velocity dependence resembles classical turbulent drag, its physical interpretation in granular flows differ fundamentally. As emphasized by [3], this term does not imply fully developed turbulence (e.g., energy cascades or eddy mixing), which would contradict the stratified internal structures observed in avalanche deposits [10]. Instead, it phenomenologically captures energy dissipation mechanisms arising from:

- Particle collisions and random motion within the granular mass;
- Interactions with rough basal topography;
- Shear-induced agitation in the flow core.

Following [11], the term V^2/ξ approximates the conversion of organized downslope kinetic energy into disordered motion. This coarse-grained approach avoids resolving microscale interactions while preserving bulk flow behavior.

2.3. Shallow Water Equations with Friction as a Source

The SWE (2.10), augmented with the Friction Law (2.11) as a source term, forms the governing equations for gravity-driven granular avalanches:

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (hu)}{\partial x} + \dfrac{\partial (hv)}{\partial y} = 0 \\ \dfrac{\partial (hu)}{\partial t} + \dfrac{\partial}{\partial x}\left(hu^2 + \dfrac{gh^2}{2}\right) + \dfrac{\partial (huv)}{\partial y} = -gh\dfrac{\partial b}{\partial x} - \dfrac{u}{V}\left(\mu gh\cos(\alpha) + g\dfrac{V^2}{\xi}\right) \\ \dfrac{\partial (hv)}{\partial t} + \dfrac{\partial (hvu)}{\partial x} + \dfrac{\partial}{\partial y}\left(hv^2 + \dfrac{gh^2}{2}\right) = -gh\dfrac{\partial b}{\partial y} - \dfrac{v}{V}\left(\mu gh\cos(\alpha) + g\dfrac{V^2}{\xi}\right), \end{cases} \tag{2.18}$$

where:

- $\mathbf{V} = (u, v)^T$ is the depth-averaged horizontal velocity vector;
- $V = \sqrt{u^2 + v^2}$ is the magnitude of $\mathbf{V}$;
- α is the local slope angle, determined by the surface topography $b(x, y)$;
- $\mu = \tan(\theta)$ is the basal friction coefficient, assumed constant;
- ξ is the Voellmy turbulence coefficient, assumed constant.

The angle α represents the inclination between the vertical axis OZ and the normal vector $\mathbf{n}_{\text{loc}}$ to the surface (see Figure 2.3b), given by:

$$\mathbf{n}_{\text{loc}} = \left(-\frac{\partial b}{\partial x}, -\frac{\partial b}{\partial y}, 1\right)^T.$$

Taking the scalar product with the vertical unit vector $(0, 0, 1)^T$, the $\cos(\alpha)$ is obtained:

$$\cos(\alpha) = \frac{1}{\sqrt{\left(\frac{\partial b}{\partial x}\right)^2 + \left(\frac{\partial b}{\partial y}\right)^2 + 1}}. \tag{2.19}$$

The friction term on the right-hand side of the momentum equations (2.18) is applied in the direction opposite to the velocity vector. This is achieved by multiplying the scalar friction magnitude by the depth-averaged velocity components normalized by V.

The bathymetry $b(x, y)$ in classical SWE is replaced by the elevation $z(x, y)$, emphasizing its role as the basal surface in granular flow modeling. The substitution preserves the mathematical structure while clarifying the physical interpretation of terrain interaction.

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (hu)}{\partial x} + \dfrac{\partial (hv)}{\partial y} = 0 \\ \dfrac{\partial (hu)}{\partial t} + \dfrac{\partial}{\partial x}\left(hu^2 + \dfrac{gh^2}{2}\right) + \dfrac{\partial (huv)}{\partial y} = -gh\dfrac{\partial z}{\partial x} - \dfrac{u}{V}\left(\mu gh\cos(\alpha) + g\dfrac{V^2}{\xi}\right) \\ \dfrac{\partial (hv)}{\partial t} + \dfrac{\partial (hvu)}{\partial x} + \dfrac{\partial}{\partial y}\left(hv^2 + \dfrac{gh^2}{2}\right) = -gh\dfrac{\partial z}{\partial y} - \dfrac{v}{V}\left(\mu gh\cos(\alpha) + g\dfrac{V^2}{\xi}\right). \end{cases} \tag{2.20}$$

3. The ADER-DG Method

Modern computational fluid dynamics (CFD) increasingly relies on high-order numerical methods for accurate and efficient solutions to partial differential equations (PDEs). Among these methods, the ADER-DG (Arbitrary DERivative Discontinuous Galerkin) approach has emerged as a powerful framework for unified space-time discretization of hyperbolic PDEs in the general form:

$$\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) + \mathcal{B}(Q) \cdot \nabla Q = S(Q), \tag{3.1}$$

where Q represents unknown variables, F denotes the flux function, $\mathcal{B}(Q) \cdot \nabla Q$ stands for the non-conservative product (NCP), and $S(Q)$ is the source term. This formulation is of particular importance in the context of the present work, as the SWE with friction effects, given in (2.20), can be expressed as (3.1) through the following notation:

$$\begin{aligned} Q = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix} \qquad \mathbf{F}(Q) \quad &= \begin{pmatrix} hu & hv \\ hu^2 + \frac{1}{2}gh^2 & huv \\ huv & hv^2 + \frac{1}{2}gh^2 \end{pmatrix} \\ \mathcal{B}(Q) \cdot \nabla Q = \begin{pmatrix} 0 \\ gh\frac{\partial z}{\partial x} \\ gh\frac{\partial z}{\partial y} \end{pmatrix} \qquad S(Q) \quad &= \begin{pmatrix} 0 \\ -\frac{u}{V}\left(\mu gh\cos\alpha + \frac{gV^2}{\xi}\right) \\ -\frac{v}{V}\left(\mu gh\cos\alpha + \frac{gV^2}{\xi}\right) \end{pmatrix}, \end{aligned} \tag{3.2}$$

where elevation z is stored as an auxiliary variable.

A distinguishing feature of the ADER-DG method is the ability to achieve high-order accuracy in both space and time within a single computational step. In contrast to classical DG methods, which apply separate spatial and temporal discretization (typically using multi-stage Runge-Kutta schemes), the ADER-DG method enables the reconstruction of the solution directly in the unified space-time domain using a local space-time polynomial basis.

A single ADER-DG time-step comprises two principal stages: the predictor and the corrector. The predictor stage computes a local-in-time evolution within each cell independently, while the corrector stage resolves inter-element interactions through numerical flux computations at cell interfaces.

This chapter outlines the fundamental components of the ADER-DG method, following the theoretical frameworks established in [12], [13], and [5].

3.1. The 1D Formulation of the ADER-DG Method

First, the application of the ADER-DG method to the one-dimensional (1D) form of (3.1) is considered:

$$\frac{\partial Q}{\partial t} + \frac{\partial F(Q)}{\partial x} + \mathcal{B}(Q)\frac{\partial Q}{\partial x} = S(Q), \tag{3.3}$$

where:

- **Flux term** $\partial F(Q)/\partial x$: Characterizes spatial transport of conserved quantities due to advective and pressure-driven mechanisms. Examples include momentum flux in fluid dynamics or energy flux in thermal systems.
- **Non-conservative product** $\mathcal{B}(Q) \cdot \partial Q/\partial x$: Arises in systems that lack a fully conservative formulation, such as models including topographic effects. These terms require specialized numerical treatment to ensure consistency and stability.
- **Source term** $S(Q)$: Encodes external influences modifying system dynamics, including gravitational forces, friction effects, or chemical reactions.

The one-dimensional formulation provides a clear and concise framework for demonstrating all key aspects of the numerical method. It maintains conceptual simplicity while preserving the straightforward path for generalizing the results to multidimensional problems.

The computational domain is discretized into a uniform grid of non-overlapping (e) cells $[x_{i-1/2}, x_{i+1/2}]$ with theirs centers at x_i, as illustrated in Figure 3.1. This structured partitioning simplifies both implementation and analysis while preserving the essential numerical characteristics of the method.

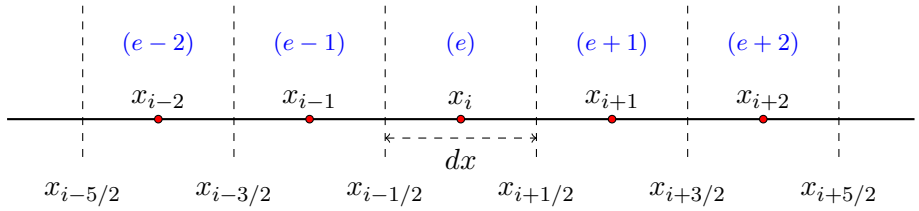


Figure 3.1.: Schematic of the uniform 1D computational grid showing cell centers (red circles) and interfaces (dashed lines). The characteristic cell width dx remains constant throughout the domain.

3.1.1. ADER-DG Prerequisites: Polynomial Bases, Quadrature, and Reference Cell Mapping

Before constructing the ADER-DG method in detail, it is necessary to introduce several mathematical tools that form the foundation of the discretization framework. This section of work describes the polynomial basis functions, numerical quadrature techniques, and the mapping between physical and reference elements. These components are essential for the space-time predictor and corrector stages.

In Discontinuous Galerkin (DG) methods, the approximate solution within each computational cell is sought as an expansion over a set of locally defined polynomial basis functions.

Each cell evolves independently, and the solution is represented as a finite linear combination of basis functions defined on a reference domain.

In the ADER-DG method, this approach is extended to a unified space-time discretization framework, with distinct representations for the predictor and corrector stages:

- **Predictor stage:** The space-time solution $\hat{Q}^{(e)}(x,t)$ is expanded over a tensor-product basis consisting of spatial and temporal polynomials:

$$\hat{Q}^{(e)}(x,t) = \sum_{i=0}^{n_p-1} \ell_i^{(e)}(r(x))\ell_j^{(e)}(t(x))\hat{q}_{ij}^{(e)}, \tag{3.4}$$

  where $\ell_i^{(e)}(r)$, $\ell_j^{(e)}(\tau)$ are Lagrange interpolation polynomials, and $\hat{q}_{ij}^{(e)}$ are unknown space-time coefficients, independent of x and t.

- **Corrector stage:** The solution $Q^{(e)}(x,t)$ is represented by a purely spatial expansion with time-dependent coefficients:

$$Q^{(e)}(x,t) = \sum_{j=0}^{n_p-1} \ell_i^{(e)}(r(x))q_i^{(e)}(t), \tag{3.5}$$

  where $q_i^{(e)}(t)$ are the corrector coefficients evolving in time.

Properties of the Basis Functions

The basis functions used in this work are Lagrange interpolation polynomials defined over the reference interval $[0,1]$:

$$\ell_i(r) = \prod_{\substack{j=0 \\ j\neq i}}^{n_p-1} \frac{r-r_j}{r_i-r_j}, \tag{3.6}$$

where r_j denote the interpolation nodes.

Two key mathematical properties characterize the basis:

1. **Interpolation property:** Each basis function $\ell_i^{(e)}$ evaluates to unity at its associated node $r_i^{(e)}$ and zero at all others:

$$\ell^{(e)}\left(r_j^{(e)}\right) = \delta_{ij}, \quad \delta_{ij} = \begin{cases} 1, & i=j \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

   In (3.7) δ_{ij} denotes the Kronecker delta.

2. L^2 **Orthogonality:** Basis functions are mutually orthogonal with respect to the L^2 inner product over the reference element:

$$\int_0^1 \ell_i^{(e)}(r)\ell_j^{(e)}(r)dr = \delta_{ij}\ell_i^{(e)}(r)\ell_i^{(e)}(r)dr. \tag{3.8}$$

These properties enhance numerical stability and simplify the computation of integral terms appearing in the weak formulations.

Numerical Integration Using Gauss-Legendre Quadrature

Accurate and efficient numerical integration is critical for DG-like methods. In this work, the Gauss-Legendre quadrature rule [14] is employed for the evaluation of integrals over the reference interval $[0, 1]$. The quadrature rule approximates an integral as:

$$\int_0^1 f(r)dr = \sum_{j=0}^{n_p-1} w_j f(r_j) + O(2n_p - 1), \tag{3.9}$$

where w_j and r_j are the quadrature weights and nodes, respectively, chosen such that the quadrature is exact for polynomials of degree up to $2n_p - 1$. It enables efficient numerical integration without sacrificing precision. Figure 3.2 illustrates the distribution of the quadrature nodes within the reference element for $n_p = 4$.

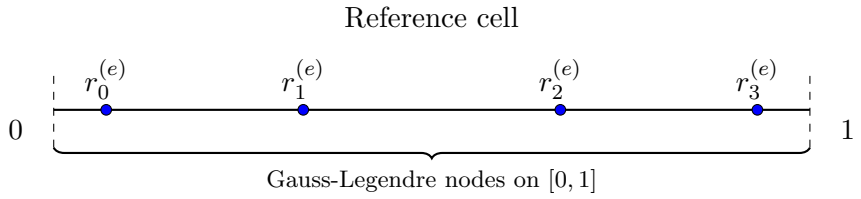
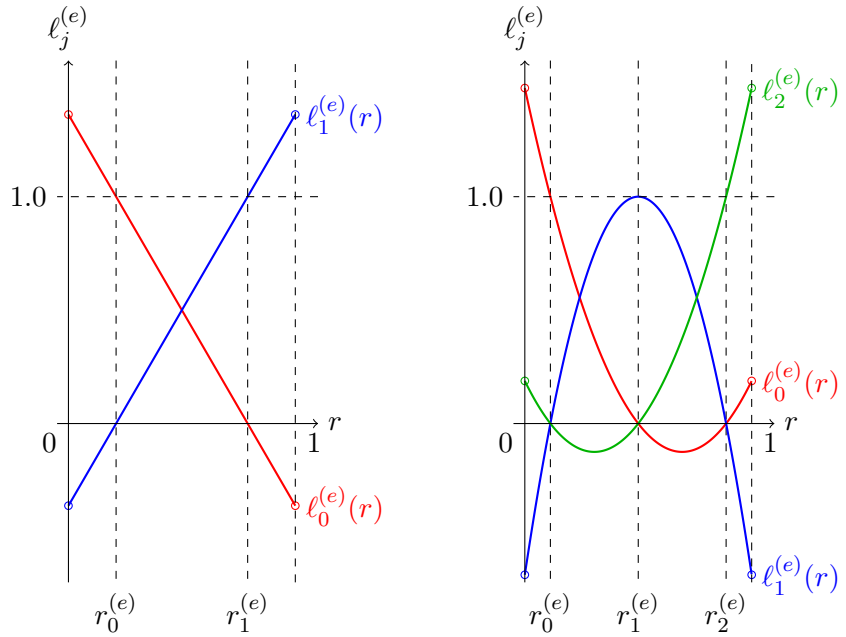


Figure 3.2.: Gauss-Legendre quadrature nodes r_j for $n_p = 4$.



(a) Linear basis ($n_p = 1$). (b) Quadratic basis ($n_p = 2$).

Figure 3.3.: Examples of Lagrange basis functions constructed on Gauss-Legendre nodes.

The Gauss-Legendre nodes offer two key advantages that are particularly important for high-order discretizations. First, their non-uniform distribution within the interval $[0, 1]$ helps

to mitigate Runge's phenomenon [15], which can arise when using equally spaced interpolation points. Second, all nodes lie strictly within the open interval $(0, 1)$, which aligns with the philosophy of discontinuous methods that treat interface fluxes separately and avoid the ambiguity of including boundary points in element-local formulations. Figure 3.3 illustrates the construction of the Lagrange interpolation polynomials on Gauss-Legendre nodes.

Mapping to the Reference Cell

Although the basis functions and quadrature rules are defined on the fixed reference interval $[0, 1]$, physical computational cells may vary in size and location. To ensure uniform treatment across all cells, an affine mapping between physical and reference coordinates is introduced.

The spatial mapping from a reference coordinate $r \in [0, 1]$ to a physical coordinate x is defined as:

$$x(r) = x_C + \Delta x(r - r_C), \tag{3.10}$$

where x_C is the center of the physical cell, Δx its size and $r_C = 0.5$ the center of the reference cell. The spatial mapping is illustrated in Figure 3.4.

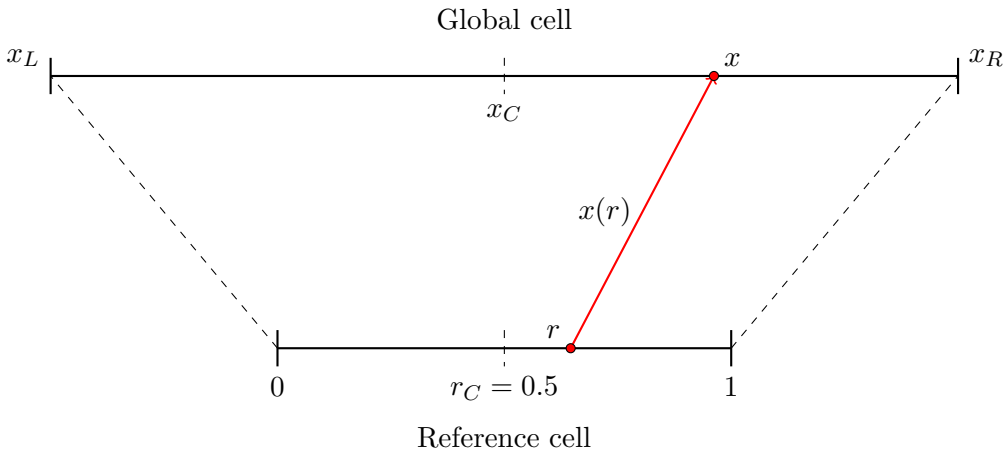


Figure 3.4.: Affine mapping from a reference cell $[0, 1]$ to the global cell $[x_L, x_R]$.

Similarly, the temporal mapping between $\tau \in [0, 1]$ and physical $t \in [t_k, t_{k+1}]$ reads:

$$t(\tau) = t_k + \Delta t\tau, \tag{3.11}$$

where t_k is the starting time of the current time-step, and $\Delta t = t_{k+1} - t_k$ is the current time-step size.

These transformations allow the usage of (3.9) through variable substitution. Typical substitutions include:

- Volume integrals of the function:

$$\int_{\Omega_e} \ell_i^{(e)}(r(x)) f(x) dx = \Delta\Omega^{(e)} \int_0^1 \ell_i^{(e)}(r) f(x(r)) dr, \tag{3.12}$$

  where $\Delta\Omega_e$ denotes the size of the physical cell.

- Volume integrals of the first derivative:

$$\int_{\Omega_e} \ell_i^{(e)}(r(x)) \frac{\partial f}{\partial x} dx = \frac{\Delta\Omega^{(e)}}{\Delta x} \int_0^1 \ell_i^{(e)}(r) \frac{\partial f(x(r))}{\partial r} dr. \tag{3.13}$$

Thus, all numerical operations (basis evaluation, integration, differentiation) are consistently performed within the reference coordinate system.

3.1.2. ADER-DG Predictor

The Weak Formulation of the Predictor

The predictor stage of the ADER-DG method aims to compute a local-in-time, local-in-space solution $\hat{Q}^{(e)}(x,t)$ within each cell, independently of its neighbors. During this stage, inter-element fluxes are ignored, and each cell evolves solely according to its internal dynamics.

The predictor solution is expanded over spatio-temporal basis functions, as defined in (3.4). Due to the interpolation property (3.7), each unknown coefficient $\hat{q}_{ij}^{(e)}$ represents the value of $\hat{Q}^{(e)}$ at the node (r_i, τ_j). Thus, finding the predictor reduces to determining the coefficients $\hat{q}_{ij}^{(e)}$.

Analogously, predictor expansion for the flux and source terms are introduced:

$$\begin{aligned} \hat{F}^{(e)}(x,t) &= \sum_{i=0}^{n_p-1} \sum_{j=0}^{n_p-1} \ell_i^{(e)}(r(x)) \ell_j^{(e)}(t(x)) \hat{f}_{ij}^{(e)}, \\ \hat{S}^{(e)}(x,t) &= \sum_{i=0}^{n_p-1} \sum_{j=0}^{n_p-1} \ell_i^{(e)}(r(x)) \ell_j^{(e)}(t(x)) \hat{s}_{ij}^{(e)}, \end{aligned} \tag{3.14}$$

where the flux and source coefficients are evaluated as $\hat{f}_{ij}^{(e)} = F(\hat{q}_{ij}^{(e)})$ and $\hat{s}_{ij}^{(e)} = S(\hat{q}_{ij}^{(e)})$, respectively. The NCP is absorbed into the source term at this stage. If $n_p = 1$, corresponding to constant polynomials, spatial derivatives vanish, and the NCP must be treated during the corrector stage instead.

To derive the weak form, the governing PDE (3.3) is rewritten by replacing Q, $F(Q)$, and $S(Q)$ with their predictor counterparts. The resulting equation is multiplied by the spatio-temporal test function

$$\psi_{mn}^{(e)}(r,\tau) = \ell_m^{(e)}(r) \ell_n^{(e)}(\tau), \tag{3.15}$$

and integrated over the space-time reference element.

Applying the transformation (3.12) and (3.13) to the corresponding integrals, the weak form becomes:

$$\int_0^1 \int_0^1 \psi_{mn}^{(e)} \left(\frac{\partial \hat{Q}^{(e)}}{\partial \tau} + \Delta t \frac{\Delta\Omega^{(e)}}{\Delta x^{(e)}} \frac{\partial \hat{F}^{(e)}}{\partial r} - \Delta t \hat{S}^{(e)} \right) dr\, d\tau = 0. \tag{3.16}$$

The first term, involving the time derivative, is integrated by parts with respect to τ:

$$\int_0^1 \int_0^1 \psi_{mn}^{(e)} \frac{\partial \hat{Q}^{(e)}}{\partial \tau} \, dr \, d\tau = \int_0^1 \psi_{mn}^{(e)}(r,\tau) \hat{Q}^{(e)}(r,t(\tau)) \, dr \Big|_{\tau=0}^{\tau=1} - \int_0^1 \int_0^1 \frac{\partial \psi_{mn}^{(e)}}{\partial \tau} \hat{Q}^{(e)} \, dr \, d\tau. \tag{3.17}$$

The boundary term introduces a temporal flux between the current and next time-steps. Following the upwinding principle in time:

- The flux at $\tau = 0$ uses the known corrector solution $Q^{(e)}(r, t(0))$ from the previous time-step $t(0) = t_k$.
- The flux at $\tau = 1$ involves the predictor solution at the next time-step.

Substituting (3.17) into (3.16), the full weak formulation of the predictor reads:

$$\int_0^1 \psi_{mn}^{(e)}(r,1) \hat{Q}^{(e)}(r,t(1)) \, dr - \int_0^1 \int_0^1 \frac{\partial \psi_{mn}^{(e)}}{\partial \tau} \hat{Q}^{(e)} \, dr \, d\tau = \int_0^1 \psi_{mn}^{(e)}(r,0) Q^{(e)}(r,t(0)) \, dr + \Delta t \int_0^1 \int_0^1 \left(\hat{S}^{(e)} - \frac{\Delta \Omega^{(e)}}{\Delta x^{(e)}} \frac{\partial \hat{F}^{(e)}}{\partial r} \right) \psi_{mn}^{(e} \, dr \, d\tau. \tag{3.18}$$

The Numerical Discretization of the Predictor

To obtain a fully discrete version of the (3.18), the following steps should be applied:

1. Substitute expansions (3.4), (3.14), and (3.5);
2. Approximate the integrals using Gauss-Legendre quadrature (3.9);
3. Apply the interpolation (3.7) and orthogonality (3.8) properties of the basis functions.

This procedure yields the discrete form of the predictor:

$$w_m \sum_{j=0}^{n_p-1} \left(\ell_n^{(e)} \ell_j(1) - w_j \frac{\partial \ell_n^{(e)}}{\partial \tau}(\tau_j) \right) \hat{q}_{mj}^{(e)} = = w_m \ell_n^{(e)}(0) q_m^{(e)}(t(0)) + \Delta t w_m w_n \hat{s}_{mn}^{(e)} - \Delta t \frac{\Delta \Omega^{(e)}}{\Delta x} w_m w_n \sum_{i=0}^{n_p-1} \hat{f}_{in}^{(e)} \frac{\partial \ell_i^{(e)}}{\partial r}(r_m). \tag{3.19}$$

To solve the nonlinear system (3.19), an iterative scheme is introduced:

$$w_m \sum_{j=0}^{n_p-1} \left(\ell_n^{(e)} \ell_j(1) - w_j \frac{\partial \ell_n^{(e)}}{\partial \tau}(\tau_j) \right) \hat{q}_{mj}^{(e,s+1)} = = w_m \ell_n^{(e)}(0) q_m^{(e)}(t(0)) + \Delta t w_m w_n \hat{s}_{mn}^{(e,s)} - \Delta t \frac{\Delta \Omega^{(e)}}{\Delta x} w_m w_n \sum_{i=0}^{n_p-1} \hat{f}_{in}^{(e,s)} \frac{\partial \ell_i^{(e)}}{\partial r}(r_m), \tag{3.20}$$

where s denotes the iteration index. The iteration is initialized by setting all predictor coefficients equal to the corrector values at the previous time-step, as proposed in [12]:

$$\hat{q}_{ij}^{(e,0)} = q_j^{(e)}(t(0)). \tag{3.21}$$

The Matrix Form of the Predictor

For clarity and efficient numerical implementation, the discrete predictor scheme (3.20) can be reformulated in a compact matrix-vector form.

The predictor coefficients $\hat{q}_{mj}^{(e)}$ naturally form a matrix of size $n_p \times n_p$, where:

- The row index m corresponds to spatial quadrature nodes r_m;
- The column index j corresponds to temporal quadrature nodes τ_j.

To express the system as a matrix-vector equation, the matrix $\hat{q}_{mj}^{(e)}$ is vectorized into a column vector $\hat{\mathbf{q}}^{(e)}$ by stacking spatial slices for each temporal node sequentially:

$$\hat{\mathbf{q}}^{(e)} = \left(\hat{q}_{00}^{(e)}, \hat{q}_{10}^{(e)}, \ldots, \hat{q}_{n_p-1,n_p-1}^{(e)}\right)^{\mathrm{T}}. \tag{3.22}$$

The same vectorization procedure is applied to the source and flux predictors, resulting in vector $\hat{\mathbf{s}}^{(e)}$ and $\hat{\mathbf{f}}^{(e)}$.

Since the corrector coefficients $q_m^{(e)}(t(0))$ depend only on space, they form a vector of size n_p. To match the size of the predictor vector, the corrector solution is extended by repeating its spatial values for each temporal node:

$$\mathbf{q}_{\mathrm{ext}}^{(e)} = \left(q_0^{(e)}, q_1^{(e)}, \ldots, q_{n_p-1}^{(e)}, \ldots, q_0^{(e)}, \ldots, q_{n_p-1}^{(e)}\right)^{\mathrm{T}}. \tag{3.23}$$

With this notation, the initial guess (3.21) can be rewritten as follows:

$$\hat{\mathbf{q}}^{(e,0)} = \mathbf{q}_{\mathrm{ext}}^{(e)}(t(0)). \tag{3.24}$$

The same vectorization procedure is applied to the source $\hat{s}_{mj}^{(e)}$ and flux $\hat{f}_{mj}^{(e)}$ predictors, resulting in the vectors $\hat{\mathbf{s}}^{(e)}$ and $\hat{\mathbf{f}}^{(e)}$, respectively.

To express the discrete predictor system in a structured matrix form, the following auxiliary matrices are introduced:

- **Weight matrices:**

$$\mathbf{W} = \mathrm{diag}(w_0, w_1, \ldots, w_{n_p-1}), \qquad \mathbf{W}_{\mathrm{sp}} = \mathbf{W}, \qquad \mathbf{W}_{\mathrm{t}} = \mathbf{W}, \tag{3.25}$$

  where w_i are Gauss-Legendre quadrature weights.

- **Temporal coupling matrix:**

$$(\mathbf{K}_{\mathrm{t}})_{ij} = \ell_j^{(e)}(1)\ell_i^{(e)}(1) - w_i \frac{\partial \ell_j^{(e)}}{\partial \tau}(\tau_i), \tag{3.26}$$

  encoding contributions from the temporal flux and time derivative.

- **Spatial derivative matrix:**

$$(\mathbf{K}_{\mathrm{x}})_{ij} = w_j \frac{\partial \ell_i^{(e)}}{\partial r}(r_j), \tag{3.27}$$

  related to the flux divergence term in the weak form.

- **Identity matrices:**

$$\mathbf{I} \in \mathbb{R}^{n_p \times n_p} \qquad \mathbf{I}_{\mathrm{sp}} = \mathbf{I} \qquad \mathbf{I}_{\mathrm{t}} = \mathbf{I}. \tag{3.28}$$

- **Temporal basis matrix at initial time:**

$$\mathbf{\Phi}_{\mathrm{t}}^{(e)}(0) = \mathrm{diag}\left(\ell_0^{(e)}(0), \ell_1^{(e)}(0), \ldots, \ell_{n_p-1}^{(e)}(0)\right). \tag{3.29}$$

With these definitions, the fully discrete predictor system can be rewritten using Kronecker products:

$$\begin{aligned}
&(\mathbf{I}_{\mathrm{t}} \otimes \mathbf{W}_{\mathrm{sp}}) \cdot \left(\mathbf{K}_{\mathrm{t}}^{\mathrm{T}} \otimes \mathbf{I}_{\mathrm{sp}}\right) \cdot \hat{\mathbf{q}}^{(e,s+1)} \\
&\quad = \left(\mathbf{\Phi}_{\mathrm{t}}^{(e)}(0) \otimes \mathbf{I}_{\mathrm{sp}}\right) \cdot (\mathbf{I}_{\mathrm{t}} \otimes \mathbf{W}_{\mathrm{sp}}) \cdot \mathbf{q}_{\mathrm{ext}}^{(e)}(t(0)) \\
&\quad + \Delta t \, (\mathbf{W}_{\mathrm{t}} \otimes \mathbf{I}_{\mathrm{sp}}) \cdot (\mathbf{I}_{\mathrm{t}} \otimes \mathbf{W}_{\mathrm{sp}}) \cdot \hat{\mathbf{s}}^{(e,s)} \\
&\quad - \Delta t \frac{\Delta \Omega^{(e)}}{\Delta x} (\mathbf{W}_{\mathrm{t}} \otimes \mathbf{I}_{\mathrm{sp}}) \cdot \left(\mathbf{I}_{\mathrm{t}} \otimes \mathbf{K}_{\mathrm{x}}^{\mathrm{T}}\right) \cdot \hat{\mathbf{f}}^{(e,s)}.
\end{aligned} \tag{3.30}$$

An important advantage of the matrix formulation (3.30) is that all auxiliary matrices (including their inverses and transposes) depend solely on the choice of basis functions and quadrature nodes. Therefore, they can be precomputed once at the beginning of the simulation and reused for all time-steps, significantly reducing computational overhead.

Moreover, the Kronecker product structure allows these matrices to be applied to a vector without explicitly assembling large matrices. Instead, efficient matrix-vector multiplication is achieved by reshaping the vectors and performing two smaller matrix multiplications:

$$(\mathbf{A} \otimes \mathbf{B})\, \mathbf{X} = \mathrm{vec}\left(\mathbf{B}\mathbf{X}\mathbf{A}^{\mathrm{T}}\right),$$

where $\mathrm{vec}(\cdot)$ denotes vectorization. This drastically reduces memory consumption and improves computational efficiency, particularly for large-scale problems or higher-order approximations.

Spatial Derivatives of the Predictor

In certain applications, such as the evaluation of NCP, it is necessary to compute the spatial derivative of the predictor solution $\hat{q}^{(e)}$ at the quadrature nodes.

Recall that the predictor is represented as a space-time expansion (3.4). Differentiating with respect to the spatial variable r yields:

$$\frac{\partial \hat{q}^{(e)}}{\partial r}(r,\tau) = \sum_{i=0}^{n_p-1} \sum_{j=0}^{n_p-1} \frac{\partial \ell_i^{(e)}}{\partial r}(r) \ell_j^{(e)}(\tau) \hat{q}_{ij}^{(e)}. \tag{3.31}$$

Evaluating this expression at a given space-time quadrature node (r_m, τ_n) and applying the interpolation property (3.7) in the temporal direction leads to:

$$\left.\frac{\partial \hat{q}^{(e)}}{\partial r}\right|_{(r_m,\tau_n)} = \sum_{i=0}^{n_p-1} \frac{\partial \ell_i^{(e)}}{\partial r}(r_m) \hat{q}_{in}^{(e)}. \tag{3.32}$$

This operation can be written in matrix-vector form using a precomputed derivative matrix $\mathbf{D}^{(e)}$, defined as:

$$\left(\mathbf{D}^{(e)}\right)_{mi} = \left.\frac{d\ell_i^{(e)}}{dr}\right|_{r=r_m}. \tag{3.33}$$

With the predictor coefficients $\hat{\mathbf{q}}^{(e)}$ vectorized as in (3.22), the full spatial derivative over all space-time nodes of the quadrature can be compactly expressed as:

$$\frac{\partial \hat{\mathbf{q}}^{(e)}}{\partial r} = \left(\mathbf{I}_\mathrm{t} \otimes \mathbf{D}^{(e)}\right) \hat{\mathbf{q}}^{(e)}, \tag{3.34}$$

where $\mathbf{I}_\mathrm{t}$ is the identity matrix acting on the temporal dimension, defined in (3.28).

The matrix $\mathbf{D}^{(e)}$ depends only on the spatial quadrature nodes and the basis functions and can thus be precomputed once and reused throughout the simulation. Note that for $n_p = 1$, the basis is constant in space, and all spatial derivatives vanish. In this case, it is obvious from (3.33) that the matrix $\mathbf{D}^{(e)}$ is a zero matrix, and NCP terms must be handled entirely during the corrector stage.

3.1.3. ADER-DG Corrector

The Weak Formulation of the Corrector

In the ADER-DG method, the corrector stage is responsible for incorporating the influence of neighboring elements through numerical fluxes at cell interfaces. In contrast to the predictor stage, which computes the local-in-time evolution within a single element, the corrector takes into account inter-element interactions by resolving discontinuities at cell faces.

The solution in the corrector stage is represented solely as an expansion in the spatial basis, as defined in (3.5). The objective is to determine the coefficients $q_m^{(e)}$ of this spatial expansion at the next time layer. This is achieved by deriving a weak spatial formulation of the governing PDE (3.3) and discretizing it appropriately.

The weak form is constructed by multiplying the governing equation by a spatial test function $\ell_m^{(e)}(r)$ and integrating over the space-time cell. The variable substitution is applied to map the physical domain to the reference element of type (3.12) and (3.13). The physical flux and source terms are then replaced by their predictor counterparts, $\hat{F}^{(e)}$ and $\hat{S}^{(e)}$, obtained from the previous stage:

$$\int_0^1 \int_0^1 \ell_m^{(e)} \left(\frac{\partial Q^{(e)}}{\partial \tau} + \frac{\Delta t}{\Delta x^{(e)}} \frac{\partial \hat{F}^{(e)}}{\partial r} - \Delta t \hat{S}^{(e)} \right) dr d\tau = 0. \tag{3.35}$$

Since the test function depends only on the spatial coordinate r, the first term in (3.35) involving the time derivative can be integrated exactly:

$$\int_0^1 \int_0^1 \ell_m^{(e)}(r) \frac{\partial Q^{(e)}}{\partial \tau} d\tau dr = \int_0^1 \ell_m^{(e)} \left(Q^{(e)}(r, t(1)) - Q^{(e)}(r, t(0)) \right) dr. \tag{3.36}$$

The remaining terms are integrated in time to yield time averages. For the flux term, this results in:

$$\frac{\Delta t}{\Delta x^{(e)}} \int_0^1 \int_0^1 \ell_m^{(e)}(r) \frac{\partial \hat{F}^{(e)}}{\partial r} dr d\tau = \frac{\Delta t}{\Delta x^{(e)}} \int_0^1 \ell_m^{(e)}(r) \frac{\partial \tilde{F}^{(e)}}{\partial r}(r) dr, \tag{3.37}$$

where $\tilde{F}^{(e)}(r)$ denotes the time-averaged predictor flux:

$$\tilde{F}^{(e)}(r) = \int_0^1 \hat{F}^{(e)}(r,\tau)d\tau. \tag{3.38}$$

An analogous procedure is also applied to the source term, yielding the time-averaged source predictor $\tilde{S}^{(e)}(r)$.

To incorporate flux contributions at cell interfaces, the flux divergence term is integrated by parts with respect to space. The resulting expression constitutes the weak spatial formulation of the corrector:

$$\begin{aligned}\int_0^1 \ell_m^{(e)}(r)Q^{(e)}(r,t(1))\,dr = &\int_0^1 \ell_m^{(e)}(r)Q^{(e)}(r,t(0))\,dr \\ &- \frac{\Delta t}{\Delta x^{(e)}}\left[\tilde{F}^{(*)}(1)\ell_m^{(*)}(1) - \tilde{F}^{(*)}(0)\ell_m^{(*)}(0)\right] \\ &+ \frac{\Delta t}{\Delta x^{(e)}}\int_0^1 \ell_m^{(e)}(r)\frac{\partial \tilde{F}^{(e)}}{\partial r}(r)\,dr + \Delta t\int_0^1 \ell_m^{(e)}(r)\tilde{S}^{(e)}(r)\,dr.\end{aligned} \tag{3.39}$$

This formulation provides the foundation for the numerical discretization of the corrector step in the ADER-DG.

Riemann Solver

The corrector stage requires resolving discontinuities at cell interfaces after integrating by parts in (3.39). This necessitates a numerical flux $\tilde{F}^{(*)}$ that couples the time-averaged predictor solutions of adjacent elements. The flux is determined by solving a one-dimensional Riemann problem at each interface.

Let the unit normal n, predefined for each interface, point outward from the current element (e) to its neighbor $(e+1)$. The left state Q^L and right stage Q^R are reconstructed from the predictor solutions of elements (e) and $(e+1)$, respectively:

$$Q^L = \hat{Q}^{(e)}(x(1),t(\tau)), \qquad Q^R = \hat{Q}^{(e+1)}(x(0),t(\tau)), \tag{3.40}$$

where $x(1)$ and $x(0)$ denote the right and left boundaries of the reference cell, and $\hat{Q}^{(e)}$ is the predictor expansion defined in (3.4).

Figure 3.5 shows a typical Riemann problem.

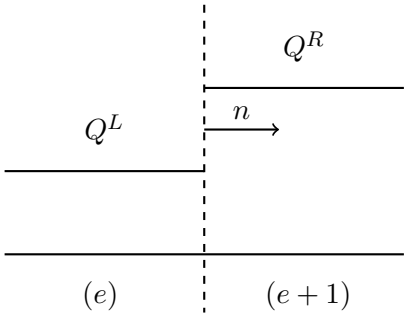


Figure 3.5.: Interface Riemann problem: left state Q^L (reconstructed from element (e)) and right state Q^R (reconstructed from element $(e+1)$). The normal n points outward from (e).

The present work adopts the Rusanov flux, also known as the local Lax-Friedrichs flux [16]:

$$F^{(*)}(Q^L, Q^R) = \frac{1}{2}\left(F(Q^L) + F(Q^R)\right) - \frac{1}{2}\lambda_{\max}(Q^L, Q^R)\left(Q^R - Q^L\right), \tag{3.41}$$

where $F(Q^{L/R})$ are the physical fluxes evaluated at the interface states. The first term in (3.41) represents the arithmetic mean of the physical fluxes, exact for smooth solutions. The second term introduces dissipation proportional to the solution jump $Q^R - Q^L$, stabilizing discontinuities. The coefficient $\lambda_{\max}$, defined as:

$$\lambda_{\max}(Q^L, Q^R) = \max\left(V_n^L + \sqrt{gh^L}, V_n^R + \sqrt{gh^R}\right) \tag{3.42}$$

is the largest characteristic speed for the system, given by (2.20); here $h^{L/R}$ is the water depth and $V_n^{L/R}$ the velocity component normal to the interface n.

For first-order approximations $n_p = 1$, spatial derivatives vanish within elements, necessitating the evaluation of NCP term at interfaces. A centered discretization is applied:

$$\text{NCP} = \frac{1}{2\Delta x^{(e)}}\left(\mathcal{B}(Q^L) + \mathcal{B}(Q^R)\right)(Q^R - Q^L), \tag{3.43}$$

where $\mathcal{B}$ is the NCP operator from (3.2). The boundary term in (3.39) is replaced by the numerical flux contributions from all element interfaces. For an element with $N_{\text{faces}} = 2$ interfaces, the summation takes the form:

$$\sum_{f=0}^{N_{\text{faces}}-1} n_f \left[\tilde{F}^{(*)}(r_f)\ell_m^{(*)}(r_f)\right] + \Delta x \cdot \text{NCP}_f = F_m^{(*)}, \tag{3.44}$$

where

- n_f is the outward-pointing unit normal vector at interface f ($n_0 = -1$, $n_1 = 1$);
- r_f is the node on the interface ($r_0 = 0$, $r_1 = 1$).
- The NCP term is incorporated into the flux only for $n_p = 1$.

The Numerical Discretization of the Corrector

The corrector stage is discretized by projecting the time-averaged predictor quantities $\tilde{F}^{(e)}$ and $\tilde{S}^{(e)}$ onto the spatial basis functions and applying numerical quadrature. The time-averaged fluxes and source predictor coefficients are computed as:

$$\tilde{f}_i^{(e)} = \sum_{j=0}^{n_p-1} w_j \hat{f}_{ij}^{(e)}, \qquad \tilde{s}_i^{(e)} = \sum_{j=0}^{n_p-1} w_j \hat{s}_{ij}^{(e)}, \tag{3.45}$$

where the summations integrate the predictor expansions (3.14) over the temporal dimension.

Substituting these expansions into the weak formulation (3.39), approximating integrals via numerical quadrature (3.9), and leveraging the interpolation (3.7) and orthogonality (3.8)

properties of the basis functions yield the discrete update for each spatial mode m in element (e):

$$w_m q_m^{(e)}(t(1)) = w_m q_m^{(e)}(t(0)) + w_i \sum_{i=0}^{n_p-1} \frac{\Delta t}{\Delta x} \frac{\partial \ell_m^{(e)}}{\partial r}(r_i) \tilde{f}_i^{(e)} + \Delta t w_m \tilde{s}_m^{(e)} - \frac{\Delta t}{\Delta x} F_m^{(*)}. \tag{3.46}$$

Here, $F_m^{(*)}$ denotes the interface flux contribution defined in (3.44). The orthogonality of the basis functions induces a diagonal mass matrix, decoupling the updates for each spatial mode m, which makes the corrector step explicit.

The Matrix Form of the Corrector Discretization

The discrete update equation (3.46) can be compactly expressed in matrix-vector notation. Let:

- $\mathbf{W}_{\text{sp}}$ be the diagonal spatial weighting matrix defined in (3.25);
- $\mathbf{K}_{\text{x}}$ be the spatial derivative matrix from (3.27);
- $\mathbf{q}^{(e)}(t(1))$, $\mathbf{q}^{(e)}(t(0))$ denote the vector of spatial coefficients at the next and current time-steps, respectively;
- $\tilde{\mathbf{f}}^{(e)}$, $\tilde{\mathbf{s}}^{(e)}$ represent the time-averaged flux and source predictor coefficients vector;
- $\mathbf{F}_m^{(*)}$ be the interface flux vector computed vie the Riemann solver.

The matrix form of the corrector stage is:

$$\mathbf{W}_{\text{sp}} \mathbf{q}^{(e)}(t(1)) = \mathbf{W}_{\text{sp}} \mathbf{q}^{(e)}(t(0)) + \frac{\Delta t}{\Delta x} \mathbf{K}_{\text{x}} \tilde{\mathbf{f}}^{(e)} + \Delta t \mathbf{W}_{\text{sp}} \tilde{\mathbf{s}}^{(e)} - \frac{\Delta t}{\Delta x} \mathbf{F}^{(*)}. \tag{3.47}$$

3.2. Extension to Multiple Dimensions

The ADER-DG framework, rigorously developed for the one-dimensional case in (3.3), generalizes systematically to higher spatial dimensions. The governing system (3.1) is expressed in a dimensionally split form to emphasize directional flux contributions:

$$\frac{\partial Q}{\partial t} + \sum_{d=1}^{\dim} \frac{\partial F_d}{\partial x_d} = S(Q) - \mathcal{B}(Q) \cdot \nabla Q, \tag{3.48}$$

where $\mathbf{F}_d$ represents the physical flux along the d-th spatial coordinate x_d, and dim denotes the spatial dimensionality.

3.2.1. ADER-DG Prerequisites: Extension to Multiple Dimensions

The numerical framework developed for the one-dimensional case extends systematically to multidimensional problems by means of tensor-product constructions. The key algorithmic components - polynomial basis functions, quadrature rules, and coordinate transformations - preserve their fundamental properties while adapting to Cartesian elements in arbitrary spatial dimensions. In the present context, each computational cell is assumed to be a hyperrectangle aligned with the coordinate axes.

Tensor-Product Basis Functions

In multiple spatial dimensions, the basis functions are constructed as tensor products of one-dimensional Lagrange polynomials (3.6) defined on the reference interval $[0, 1]$. For an element (e) in dimension dim, the multidimensional basis function is given by:

$$\varphi_m^{(e)}(\mathbf{x}) = \prod_{d=1}^{\dim} \ell_{i_d}^{(e)}(x_d), \tag{3.49}$$

where $\mathbf{x} = (x_1, \ldots, x_{\dim})^{\mathrm{T}}$ denotes the physical coordinates and $\ell_{i_d}^{(e)}(x_d)$ is the 1D Lagrange basis function (3.6) along the d-th direction. The tensor-product structure inherits key properties (3.7) and (3.8) from the one-dimensional case. The multi-index $m = (i_1, i_2, \ldots, i_{\dim})$ is used to enumerate all basis functions, typically starting with the first coordinate as depicted in Figure 3.6 for the 2D case.

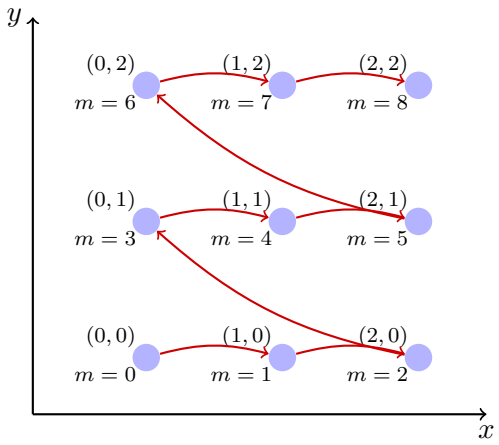


Figure 3.6.: 2D node traversal with global index $m = i_1 + n_p \cdot i_2$ ($n_p = 3$, $i_1 = x$, $i_2 = y$).

Multidimensional Quadrature

Numerical integration over the reference hypercube $[0, 1]^{\dim}$ is performed using a tensor-product extension of the Gauss-Legendre quadrature rule (3.9). For each spatial direction d, let $\{r_{i_d}, w_{i_d}\}_{i_d=0}^{n_p-1}$ denote the quadrature nodes and weights. The full set of multidimensional nodes and weights is then given by:

$$\mathbf{r}_m = (r_{i_1}, \ldots, r_{i_{\dim}}), \qquad w_m = \prod_{d=1}^{\dim} w_{i_d}, \tag{3.50}$$

ensuring exact integration of polynomials of degree up to $2n_p - 1$ in each direction.

Figure 3.7 illustrates the construction of quadrature nodes for $\dim = 2$ and $n_p = 4$.

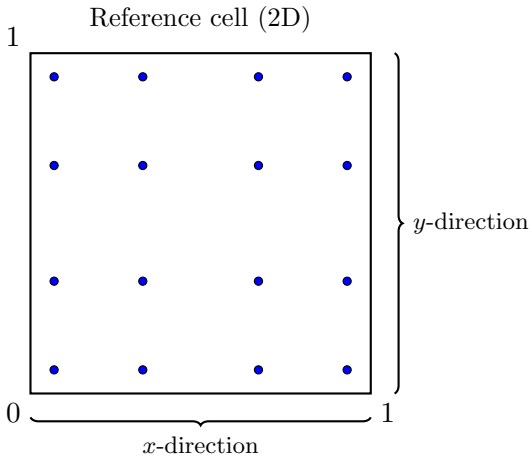


Figure 3.7.: Tensor product of 1D Gauss-Legendre nodes for $n_p = 4$ in the reference square $[0,1]^2$. The quadrature weights are computed as $w_{ij} = w_i w_j$, where w_i are 1D weights from (3.9).

Affine Coordinate Transformation

The mapping between the physical and reference elements generalizes the one-dimensional transformation (3.10). Each physical cell is represented as an affine image of the reference element:

$$\mathbf{x}(\mathbf{r}) = \mathbf{x}_C + \mathrm{diag}(\Delta\mathbf{x})(\mathbf{r} - \mathbf{r}_C), \tag{3.51}$$

where:

- $\mathbf{x}_C$ denotes the physical center of the element;
- $\Delta\mathbf{x} = (\Delta x_1, \ldots, \Delta x_{\mathrm{dim}})^{\mathrm{T}}$ are the element sizes in each direction;
- $\mathbf{r}_C = (0.5, \ldots, 0.5)^{\mathrm{T}}$ is the center of the reference element.

The Jacobian determinant of the mapping is constant and simplifies to:

$$|J| = \prod_{d=1}^{\mathrm{dim}} \Delta x_d = \Delta\Omega^{(e)}, \tag{3.52}$$

facilitating exact and efficient transformation of integrals.

Volume integrals corresponding to (3.12) and (3.13) in the multidimensional setting take the following form:

$$\int_{\Omega_e} f(\mathbf{x}) d\mathbf{x} = \Delta\Omega^{(e)} \int_{[0,1]^{\mathrm{dim}}} f(\mathbf{x}(\mathbf{r})) d\mathbf{r}, \tag{3.53}$$

and directional derivatives are handled via:

$$\int_{\Omega_e} \varphi_i^{(e)}(\mathbf{r}(\mathbf{x})) \frac{\partial f(\mathbf{x})}{\partial x_d} d\mathbf{x} = \frac{\Delta\Omega^{(e)}}{\Delta x_d} \int_{[0,1]^{\mathrm{dim}}} \varphi_i^{(e)}(\mathbf{r}) \frac{\partial f(\mathbf{x}(\mathbf{r}))}{\partial r_d} d\mathbf{r}. \tag{3.54}$$

3.2.2. Extension of the Predictor to Multiple Dimensions

The predictor stage in the multidimensional case retains the core structure of the 1D formulation (3.30) but extends spatial operators via Kronecker products constructions. This preserves computational efficiency while accommodating arbitrary spatial dimensions.

Tensor-Product Representation of Spatial Operators

The discrete spatial operators are generalized through Kronecker products of their 1D counterparts:

- **Weight matrix**:
$$\mathbf{W}_{\text{sp}} = \bigotimes_{d=1}^{\dim} \mathbf{W}, \tag{3.55}$$
where $\mathbf{W} \in \mathbb{R}^{n_p \times n_p}$ is the diagonal weight matrix from (3.25).
- **Identity matrix**:
$$\mathbf{I}_{\text{sp}} = \bigotimes_{d=1}^{\dim} \mathbf{I}, \tag{3.56}$$
where $\mathbf{I} \in \mathbb{R}^{n_p \times n_p}$ defined in (3.28).
- **Directional derivative operators:** For each spatial direction d define:
$$(\mathbf{K}_{\text{x}_d})_{ij} = w_{j_d} \frac{\partial \ell_i^{(e)}}{\partial r_d}(r_{j_d}), \tag{3.57}$$
where r_{j_d} and w_{j_d} are the quadrature nodes and weights along axis d. Due to axis-aligned symmetry, $\mathbf{K}_{\text{x}_d}$ is identical for all directions, requiring storage of only one instance.
The full multidimensional derivative operator becomes:
$$\mathbf{K}_d = \bigotimes_{k=1}^{\dim} \begin{cases} \mathbf{K}_{\text{x}_d}, & k = d, \\ \mathbf{W}, & k \neq d, \end{cases} \tag{3.58}$$
applying differentiation in direction d and integration in the others.
- **Basis differentiation matrix:** The 1D derivative matrix $\mathbf{D}^{(e)}$ from (3.33) generalizes to:
$$\left(\mathbf{D}_{x_d}^{(e)}\right)_{mi} = \left. \frac{\partial \ell_{i_d}^{(e)}}{\partial r_d} \right| r = r_{m_d}, \tag{3.59}$$
where $\mathbf{D}_{x_d}^{(e)}$ contains spatial derivatives of 1D basis functions. The full multidimensional derivative operator becomes:
$$\mathbf{D}_d = \bigotimes_{k=1}^{\dim} \begin{cases} \mathbf{D}_{\text{x}_d}, & k = d, \\ \mathbf{I}, & k \neq d, \end{cases} \tag{3.60}$$

replacing $\mathbf{D}^{(e)}$ from (3.34) in the multidimensional context:

$$\frac{\partial \hat{\mathbf{q}}^{(e)}}{\partial r_d} = \left(\mathbf{I}_\mathrm{t} \otimes \mathbf{D}_d^{(e)}\right) \hat{\mathbf{q}}^{(e)}, \tag{3.61}$$

This operator applies differentiation only along the d-th axis while leaving other directions unchanged.

Multidimensional ADER-DG Predictor

With these definitions, the multidimensional predictor equation retains the same structure as in the one-dimensional case, with only changes in the spatial matrices. The discrete predictor equation becomes:

$$\begin{aligned}\left(\mathbf{I}_\mathrm{t} \otimes \mathbf{W}_\mathrm{sp}\right)\left(\mathbf{K}_\mathrm{t}^\mathrm{T} \otimes \mathbf{I}_\mathrm{sp}\right) \hat{\mathbf{q}}^{(e,s+1)} =& \left(\mathbf{\Phi}_\mathrm{t}^{(e)}(0) \otimes \mathbf{I}_\mathrm{sp}\right)\left(\mathbf{I}_\mathrm{t} \otimes \mathbf{W}_\mathrm{sp}\right)\mathbf{q}_\mathrm{ext}^{(e)}(t(0)) \\ &+ \left(\mathbf{W}_\mathrm{t} \otimes \mathbf{I}_\mathrm{sp}\right)\left(\mathbf{I}_\mathrm{t} \otimes \mathbf{W}_\mathrm{sp}\right)\hat{\mathbf{s}}^{(e,s)} \\ &+ \sum_{d=1}^{\dim}\left(\mathbf{W}_\mathrm{t} \otimes \mathbf{I}_\mathrm{sp}\right)\left(\mathbf{I}_\mathrm{t} \otimes \mathbf{K}_d^\mathrm{T}\right)\hat{\mathbf{f}}_d^{(e,s)}.\end{aligned} \tag{3.62}$$

Here, the flux predictor $\hat{\mathbf{f}}_d^{(e,s)}$ corresponds to the contribution of the flux in the d-th direction. Temporal terms $\mathbf{K}_\mathrm{t}$, $\mathbf{\Phi}_\mathrm{t}^{(e)}(0)$, and $\mathbf{I}_\mathrm{t}$ are inherited unchanged from 1D (see (3.26), (3.29), (3.28)). This formulation preserves the block structure and enables efficient implementation using Kronecker product identities and matrix reshaping techniques.

3.2.3. Extension of the Corrector to Multiple Dimensions

The corrector stage extends naturally to multidimensional problems by generalizing the 1D formulation in (3.47). The discrete update equation becomes:

$$\mathbf{W}_\mathrm{sp}\mathbf{q}^{(e)}(t(1)) = \mathbf{W}_\mathrm{sp}\mathbf{q}^{(e)}(t(0)) + \Delta t \mathbf{W}_\mathrm{sp}\tilde{\mathbf{s}}^{(e)} + \sum_{d=0}^{\dim} \frac{\Delta t}{\Delta x_d}\left(\mathbf{K}_\mathrm{d}\tilde{\mathbf{f}_\mathbf{d}}^{(e)} - \mathbf{F}_d^{(*)}\right), \tag{3.63}$$

where

- $\mathbf{W}_\mathrm{sp}$ is the multidimensional weight matrix defined in (3.55);
- $\mathbf{K}_\mathrm{d}$ is the directional derivative operator from (3.58);
- $\tilde{\mathbf{f}}_d^{(e)}$ represents the time-averaged flux predictor in the d-th spatial direction;
- $\mathbf{F}_d^{(*)}$ contains the numerical fluxes computed via the Riemann solver at interfaces orthogonal to the d-th axis.

Numerical Flux Computation on Multidimensional Interfaces

The Riemann solver framework, originally developed for 1D interfaces, extends systematically to multidimensional domains through directional decoupling and face-normal projections.

For Cartesian grids aligned with coordinate axes, the numerical flux retains its core structure while addressing geometric complexities inherent to higher dimensions.

The interface flux contribution (3.44) generalizes to

$$\sum_{f=0}^{N_{\text{faces}}-1} \mathbf{n}_f \cdot \left[\int_{\partial_f [0,1]^{\text{dim}}} \tilde{F}_f^{(*)}(r) \varphi_m^{(*)}(r) dS \right] + \Delta x_f \cdot \text{NCP}_f = F_m^{(*)}, \tag{3.64}$$

where

- $\mathbf{n}_f \in \mathbb{R}^{\text{dim}}$ is the outward unit normal vector at f-th interface;
- $\partial_f [0,1]^{\text{dim}}$ denotes the f-th face of the reference hypercube;
- $\tilde{F}_f^{(*)}(r)$ is the time-averaged flux predictor at the interface computed via the Rusanov solver (3.41), orthogonal to the $d = f$-th axis;
- NCP_f is the NCP term contribution, which is only considered if $n_p = 1$ (otherwise it is 0 in the corrector).

The surface integrals in (3.64) are evaluated using tensor-product quadrature rules restricted to the interface. For a face orthogonal to the d-th axis:

$$\int_{\partial_f [0,1]^{\text{dim}}} g(\mathbf{r}) dS = \sum_{m=0}^{n_p^{\text{dim}-1}-1} w_m^{\text{face}} g(\mathbf{r}_m^{\text{face}}), \tag{3.65}$$

where the nodes $\mathbf{r}_m^{\text{face}}$ are constructed by fixing the d-th coordinate at the face position ($r_d = 0$ or 1) and combining 1D quadrature nodes in other directions:

$$\mathbf{r}_m^{\text{face}} = (r_{i_1}, \ldots, r_d^{\text{face}}, \ldots, r_{i_{\text{dim}}}), \tag{3.66}$$

with weights

$$w_m^{\text{face}} = \prod_{\substack{j=0 \\ j \neq d}}^{\text{dim}} w_{i_j}. \tag{3.67}$$

For the first-order approximation $n_p = 1$, the NCP term computation (3.43) generalizes to:

$$\text{NCP}_f = \frac{1}{2\Delta x_f^{(e)}} \left(\mathcal{B}(Q^L) + \mathcal{B}(Q^R) \right) \left(Q^R - Q^L \right), \tag{3.68}$$

where $\Delta x_f^{(e)} = \Delta x_d^{(e)}$ is the cell size along the d-th axis orthogonal to face f. This corresponds to a central difference approximation of the NCP term in the face-normal direction.

This formulation preserves the algorithmic structure of the 1D case while leveraging tensor-product operations for computational efficiency in multidimensions.

3.3. The Adaptive Time-Step

To ensure numerical stability, the time-step size Δt in the ADER-DG method is constrained by a CFL-type condition. This condition accounts for both the local wave speeds and the polynomial degree of the basis functions.

$$\Delta t = \mathrm{CFL} \frac{\min(\Delta x_d)}{\max(|\lambda|)} \beta(N), \tag{3.69}$$

where

- $\mathrm{CFL} \in (0, 1]$ is a user-defined relaxation parameter that controls the fraction of the maximum stable time-step;
- $\min(\Delta x_d)$ is the minimum characteristic element size across all spatial dimensions d;
- $\max(|\lambda|)$ is the maximum spectral radius (i.e., maximum absolute wave speed) evaluated via (3.42) at all quadrature points in all elements;
- $\beta(N) = (2N + 1)^{-1}$ is the stability scaling factor depending on the polynomial degree N;
- $N = n_p - 1$ is the degree of the basis polynomials used for the solution representation.

The factor $\beta(N)$ arises from von Neumann stability analysis of modal DG schemes [12], [17] and reflects the increasing stiffness of higher-order approximations. As N increases, $\beta(N)$ decreases, making the stability condition more restrictive as shown in Figure 3.8.

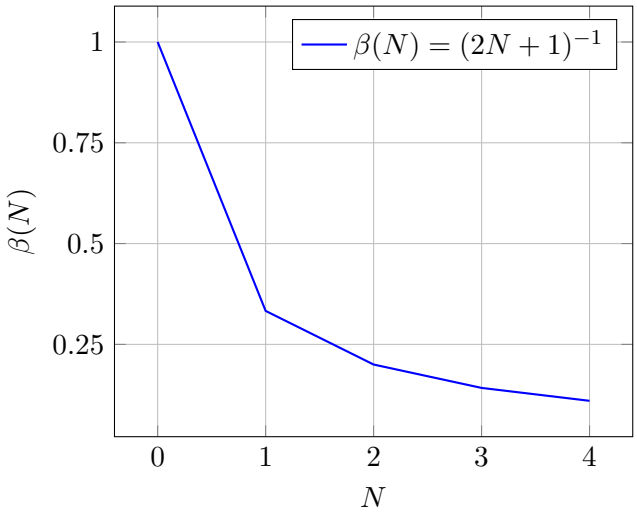


Figure 3.8.: Stability factor $\beta(N)$

For a comprehensive and rigorous analysis of the CFL condition in the context of space-time DG methods and the Riemann solver, the reader is referred to [18].

4. Addressing Challenges in ADER-DG Landslide Simulations

Although the ADER-DG method offers high-order accuracy and efficiency in smooth regions of the flow, its application to landslide modeling introduces several numerical challenges. In particular, steep gradients, wet/dry interfaces, and the presence of stiff source terms give rise to non-physical oscillations and instabilities, which compromise both the stability and physical reliability of the simulation.

To mitigate these issues, a combination of adaptive limiting techniques and numerical enhancements is required. Among them, the a-posteriori sub-cell Finite Volume limiter plays a crucial role by selectively reducing the local approximation order in troubled cells, thus maintaining stability without significantly degrading global accuracy.

Furthermore, the inclusion of stiff source terms necessitates a modification of the classical CFL time-step restriction to ensure stable time integration. In the context of this work, a supplementary constraint on the time-step has been derived and incorporated into the adaptive time-stepping procedure to account for source-term-induced stiffness.

This chapter presents an overview of the main challenges encountered during the implementation of the ADER-DG method for landslide simulations. It discusses the limiting strategy, wetting-drying treatment, stiff source term handling, and the proposed refinement of the time-step condition introduced to enhance robustness in the presence of stiffness.

4.1. A-Posteriori Sub-Cell Limiting

High-order numerical schemes such as ADER-DG are well known for their ability to achieve high accuracy with minimal numerical dissipation in smooth regions of the solution domain. Nevertheless, the application of such methods to hyperbolic conservation laws with discontinuities presents significant challenges related to nonlinear stability. As stated by Godunov's theorem [19], [20], any linear numerical scheme that satisfies a monotonicity-preserving property is restricted to first-order accuracy. Consequently, while high-order methods excel in capturing smooth solution features, they are prone to spurious oscillations in the vicinity of discontinuities and shock waves, which can degrade both stability and physical fidelity.

To reconcile high-order accuracy with the need for stability near non-smooth features, a-posteriori sub-cell limiting has been introduced as a hybrid approach [6]. The method begins with a global time update using the standard high-order ADER-DG scheme. Subsequently, candidate troubled cells are identified through a set of a-posteriori detection criteria. In these cells, the high-order solution is discarded, and a more robust Finite Volume scheme is applied on a finer sub-cell mesh to recompute the solution, thereby suppressing non-physical oscillations.

The implementation of the a-posteriori sub-cell limiter involves three key stages: the detection of troubled cells, the selection and application of a suitable low-order scheme

within these regions, and the seamless coupling of sub-cell solutions back into the high-order DG framework. This technique enables stable and accurate simulation of complex multiscale processes such as those encountered in landslide dynamics, where sharp fronts and strong gradients are frequently present.

4.1.1. Two-Stage Troubled-Cell Detection Framework

High-order methods like ADER-DG exhibit intrinsic oscillatory behavior due to the fundamental mismatch between smooth polynomial representations and discontinuous solutions. This spectral phenomenon, known as the Gibbs effect [21], generates nonphysical high-frequency oscillations near discontinuities. These oscillations can produce unphysical states (e.g., negative layer thickness h, pressures, or densities) and potentially destabilize numerical simulations.

To address these challenges, a-posteriori sub-cell limiting [6] implements a hierarchical detection strategy combining two complementary criteria.

Physical Admissibility Criterion

The primary detection stage evaluates fundamental physical constraints through a binary admissibility test. For the SWE, this focuses on maintaining non-negative granular layer thickness throughout the spatial domain of each cell (e):

$$\text{is-admissible}^{(e)} = \begin{cases} \text{true}, & h(x, t_{k+1}) \geq h_{\text{Threshold}} \quad \forall x \in (e), \\ \text{false}, & \text{otherwise}, \end{cases} \tag{4.1}$$

where $h_{\text{Threshold}}$ is some small positive threshold value close to zero.

Violation of (4.1) implies the landslide's free surface position below the basal topography $z(x, y)$, creating an ill-posed mathematical formulation that requires immediate correction.

Discrete Maximum Principle

The secondary detection stage applies a relaxed Discrete Maximum Principle (DMP) [6] to control oscillatory artifacts in admissible solutions. This heuristic compares the current solution $Q^{(e)}(x, t_{k+1})$ against a reference extreme from the previous time-step within a neighborhood $\mathcal{N}^{(e)}$, defined as element (e) and its face-adjacent neighbors. The solution must satisfy:

$$\forall x \in (e) : \min_{y \in \mathcal{N}^{(e)}} Q^{(e)}(y, t_{k+1}) - \delta \leq Q^{(e)}(x, t_{k+1}) \leq \max_{y \in \mathcal{N}^{(e)}} Q^{(e)}(y, t_k) - \delta, \tag{4.2}$$

where the dynamic tolerance δ combines absolute and relative components:

$$\delta = \max \left(b, k \cdot \left(\max_{y \in \mathcal{N}^{(e)}} \left(Q^{(e)}(y, t_k) \right) - \min_{y \in \mathcal{N}^{(e)}} \left(Q^{(e)}(x, t_k) \right) \right) \right). \tag{4.3}$$

The parameters governing this criterion are:

- b: Absolute threshold preventing over-limiting in near-constant regions;
- k: Relative scaling factor adapting to local solution gradients.

This dual-threshold approach permits controlled overshoots proportional to local solution variability, balancing oscillation suppression with high-order accuracy preservation. While effective for shock capturing, the DMP remains heuristic by design and may erroneously flag cells containing smooth but steep gradients, particularly when user-defined parameters are suboptimally tuned.

4.1.2. Sub-Cell Limiting via Finite Volume Projection

Finite Volume Method as a Low-Order Solver

The Finite Volume Method (FVM) [22] provides the robust low-order foundation for the a-posteriori limiting framework, leveraging its intrinsic conservation properties critical for shock-capturing. Starting from the integral form of governing equations (3.1) over the control volume (Ω):

$$\int_{\mathcal{V}} \left(\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) + \mathcal{B}(Q) \cdot \nabla Q \right) d\mathbf{x} = \int_{\mathcal{V}} S(Q) d\mathbf{x}, \tag{4.4}$$

the Gauss's theorem transforms flux divergence into surface integrals:

$$\int_{\mathcal{V}} \nabla \cdot \mathbf{F}(Q) d\mathbf{x} = \oint_{\partial \mathcal{V}} \mathbf{n} \cdot \mathbf{F}(Q) ds = \sum_{f} \int_{\partial \mathcal{V}_f} \mathbf{n}_f \cdot \mathbf{F}_f(Q) ds, \tag{4.5}$$

where f indexes cell faces and $\mathbf{n}_f$ denotes outward normal $\mathbf{n}$ of f-th face.

By the mean value theorem for integrals, a point exists within the cell where the integrand equals the volume-averaged value scaled by the volume. A first-order Taylor expansion around this point yields a first-order discretization scheme, where unknowns are interpreted as cell-averaged quantities $\bar{Q}^{(\Omega)}(t)$ located at volume centers. Therefore, within the FVM framework, the solution $Q(x,t)$ is approximated as piecewise constant $\bar{Q}^{(\Omega)}(t)$ across each Finite Volume, as illustrated in Figure 4.1.

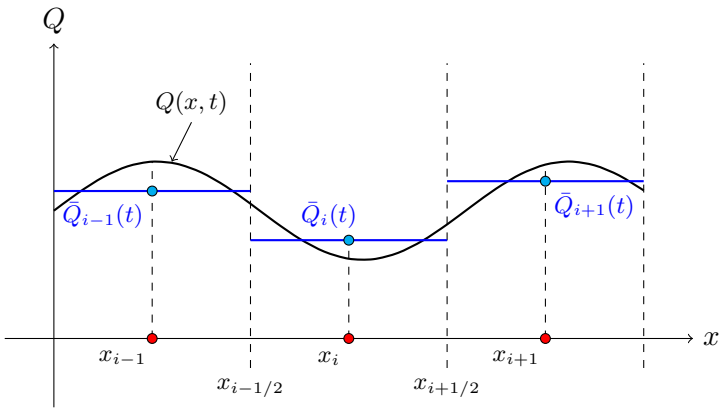


Figure 4.1.: Schematic of the FVM principle. The exact solution $Q(x,t)$ (black curve) is approximated by piecewise constant cell-averaged values $\bar{Q}_i(t)$ (blue horizontal lines). Cell centers x_i (red dots) and faces $x_{i\pm1/2}$ (dashed lines) define the FVM discretization.

The numerical fluxes on the interfaces are resolved via the Rusanov flux (3.41). The NCP terms are discretized using the path-conservative approach (3.68), integrated within the numerical flux framework as in the first-order ADER-DG corrector (3.64).

Solution Projection Strategy Between ADER-DG and FVM Layers

In the presence of non-admissible solution, such as nonphysical values or excessive oscillations, the a-posteriori sub-cell limiting procedure activates a localized fallback strategy. This strategy relies on a hierarchical recomputation scheme that ensures both stability and minimal disruption to unaffected regions.

Upon detection of a troubled cell, the sub-cell limiting process involves three successive stages. First, the solution in the troubled cell and its immediate neighbors is rejected on the new time layer. This rejection includes both the cell where admissibility criteria fail and its first-level neighbors, which may be affected by flux interactions. Next, the troubled cell, along with its first- and second-level neighbors (see Figure 4.2), is projected from the high-order ADER-DG representation to a Finite Volume (FV) sub-cell layer. This extended projection step ensures that sufficient data is available for conservative flux exchange and Riemann problem resolution across all interfaces of the recomputed region. Finally, only the troubled cell and its immediate neighbors are recomputed using a robust FVM. The second-level neighbors remain untouched except for their projection, serving solely as boundary data providers.

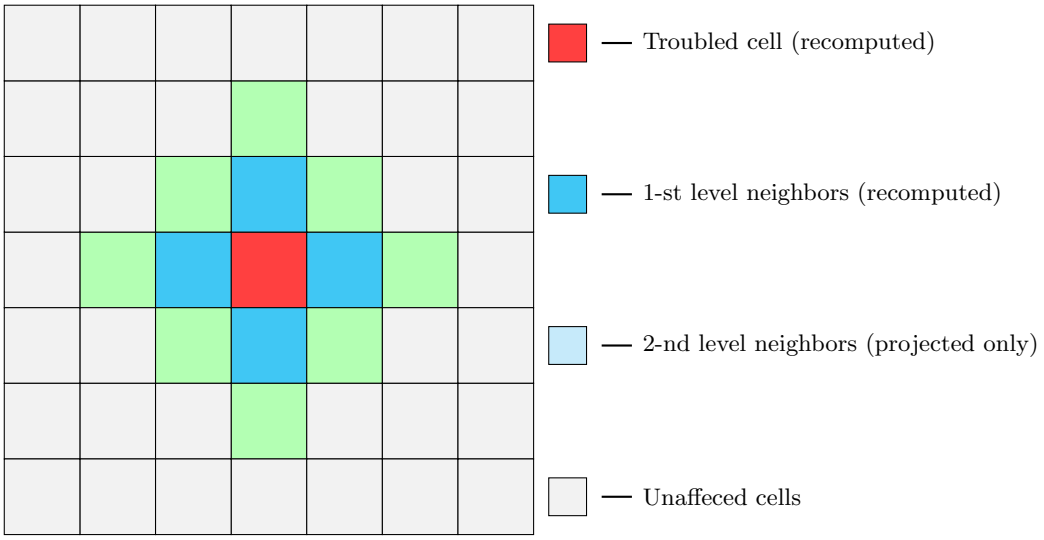


Figure 4.2.: Schematic representation of data exchange in the sub-cell limiting framework. The troubled cell (red) and its first-level neighbors (blue) are recomputed. The second-level neighbors (green) are projected to the Finite Volume layer to provide boundary data. All other cells remain unaffected (gray).

This selective and tiered projection strategy offers several advantages. By projecting second-level neighbors without triggering full recomputation, the method ensures consistency in the data passed to the FV layer, particularly at sub-cell interfaces. The localization of updates reduces computational overhead by confining modifications to a minimal neighborhood.

Moreover, isolating non-admissible regions stabilizes the overall scheme while preserving high-order accuracy in smooth domains.

Sub-Cell Resolution and Time-Step Consistency

To ensure robust updates in troubled regions, the sub-cell limiting framework introduces a subdivision of each affected element (e) into smaller sub-elements (Ω). The number of sub-cells per coordinate direction is guided by the Nyquist-Shannon sampling theorem (also known as the Kotelnikov theorem), which states that a signal containing frequency components up to $f_{\max}$ must be sampled as a rate of at least $2f_{\max}$ to prevent aliasing. In the context of the ADER-DG method, the signal is represented by the polynomial expansion (3.5) of degree $N = n_p - 1$, and the highest resolvable mode corresponds to this degree. Consequently, to capture the full spatial content of the ADER-DG solution, the FV layer must consist of at least $N_s = 2N + 1$ sub-cells per spatial dimension.

This requirement is illustrated in Figure 4.3, where the Gauss-Legendre quadrature nodes used in the ADER-DG formulation are compared to the centers of the FV sub-cells. Such a choice guarantees that the projection from ADER-DG to FV is information-preserving, ensuring that essential solution features are retained and spurious oscillations are avoided.

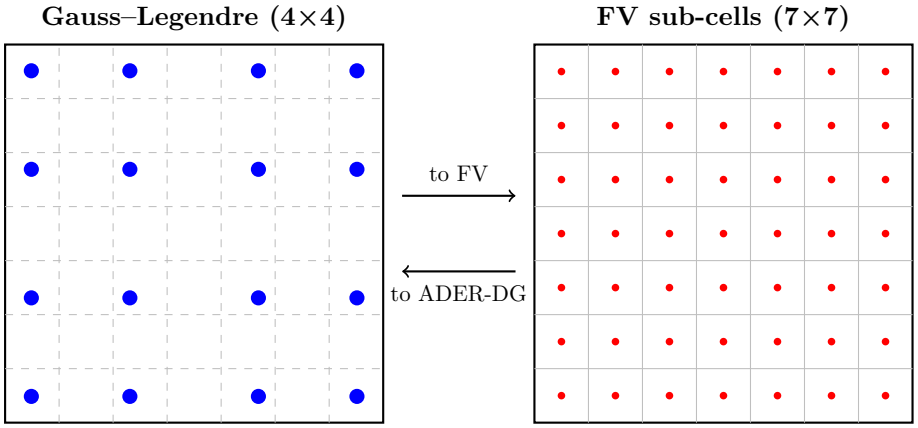


Figure 4.3.: Left: Gauss–Legendre quadrature nodes for $n_p = 4$ ($N = 3$). Right: centers of 7×7 FV sub-cells used for projection ($N_s = 2N + 1 = 7$).

An additional advantage of choosing $N_s = 2N + 1$ lies in maintaining time-step size compatibility between the ADER-DG and FV layers. The ADER-DG time-step size is governed by the CFL-type constraint (3.69). For the FV layer, the sub-cell size becomes $\Delta x_{\text{sub}} = \Delta x / N_s$, and the corresponding CFL condition reads:

$$\Delta t_{\text{FV}} = \text{CFL} \frac{\min(\Delta x_d)}{N_s \max(|\lambda|)}. \tag{4.6}$$

Since $N_s = 1/\beta(N)$, both schemes yield the same time-step. This alignment ensures consistent time integration across both solution layers, facilitating seamless switching between ADER-DG and FV representations without the need for additional time-step size adjustments.

Projection from ADER-DG to Finite Volume Sub-Cells

To ensure numerical stability during shock-capturing procedures, the high-order ADER-DG solution must be projected onto sub-cell averages compatible with the FV framework. This projection is performed in the normalized reference coordinate domain $r \in [0,1]^{\dim}$, which standardizes the spatial relationships between elements and facilitates consistent data transfer between computational layers.

For clarity, the following derivation considers the one-dimensional case. Let (Ω_i) denote the i-th sub-cell ($i \in \{0, 1, \ldots, N_s - 1\}$). The projection onto the FV layer $Q^{\Omega_i}(t_k)$ is obtained by averaging the ADER-DG corrector expansion (3.5), which is done via integration over the sub-cell's spatial extent. To align with the reference coordinate system, a local coordinate $\xi \in [0,1]$ is introduced, mapped to the global reference domain via the linear transformation $r = \Delta\xi(i + \xi)$, where $\Delta\xi = 1/N_s$ defines the sub-cell width in the reference domain.

The projection operation involves evaluating the integral:

$$Q^{(\Omega_i)}(t_k) = \sum_{j=0}^{n_p-1} \left(\ell_j^{(e)} \left(\frac{\xi - i\Delta\xi}{\Delta\xi} \right) d\xi \right) q_j^{(e)}(t_k), \tag{4.7}$$

which can be discretized using the Gauss-Legendre quadrature (3.9) within each sub-cell (Ω):

$$Q^{(\Omega_i)}(t_k) = \sum_{j=0}^{n_p-1} \left(\sum_{m=0}^{n_p-1} w_m \ell_j^{(e)} \left(\frac{\xi_m - i\Delta\xi}{\Delta\xi} \right) \right) q_j^{(e)}(t_k). \tag{4.8}$$

Here, ξ_m are the quadrature nodes within each sub-cell, and w_m are the associated weights.

The projection process is compactly represented in matrix-vector form:

$$\mathbf{Q}^{(\Omega)}(t_k) = \mathbf{A} \cdot \mathbf{q}^{(e)}(t_k), \tag{4.9}$$

where $\mathbf{Q}^{(\Omega)}(t_k) \in \mathbb{R}^{N_s}$ contains the sub-cell averages of cell (e), $\mathbf{q}^{(e)}(t_k) \in \mathbb{R}^{N+1}$ stores the ADER-DG corrector coefficients at t_k, and $\mathbf{A} \in \mathbb{R}^{N_s \times N+1}$ is the precomputable projection matrix. The matrix entries are defined as:

$$A_{ij} = \sum_{m=0}^{n_p-1} w_m \ell_j^{(e)} \left(\frac{\xi_m - i\Delta\xi}{\Delta\xi} \right), \tag{4.10}$$

where i indexes sub-cells and j indexes ADER-DG basis functions.

Reconstruction of ADER-DG Solution from Finite Volume Sub-Cells

After recomputing a robust sub-cell solution $\mathbf{Q}^{(\Omega)}(t_{k+1})$ in the troubled element and its first-level neighbors, the high-order ADER-DG representation $\mathbf{q}^{(e)}(t_{k+1})$ must be reconstructed to resume global time integration. This critical step bridges the low-order FV solution back to the ADER-DG framework.

Since the FV sub-cell layer contains $N_s = 2N + 1$ values per coordinate direction, while the ADER-DG solution is characterized by only $N + 1$, the reconstruction problem is inherently overdetermined. To address this, a constrained least-squares approach, as proposed in [6], is employed.

The objective is to determine the corrector coefficients $\mathbf{q}^{(e)}(t_{k+1})$ that minimizes the discrepancy between the FV sub-cell values and their projection from the ADER-DG basis, subject to the condition that the reconstructed polynomial conserves the total mass within element (e). This leads to the following optimization problem:

$$\min_{\mathbf{q}^{(e)}} \left\| \mathbf{Q}^{(\Omega)}(t_{k+1}) - \mathbf{A} \cdot \mathbf{q}^{(e)}) \right\|^2 \quad \text{subject to} \quad \Delta\xi^{\mathrm{T}} \mathbf{Q}^{(\Omega)}(t_{k+1}) = \mathbf{w}^{\mathrm{T}} \mathbf{q}^{(e)}(t_{k+1}), \tag{4.11}$$

where:

- $\mathbf{A}$ is the projection matrix defined in (4.10);
- $\Delta\xi$ is a vector of sub-cell sizes;
- $\mathbf{w}$ is a vector of the Gauss-Legendre quadrature weights.

To incorporate the mass conservation constraint, a Lagrange multiplier $\beta \in \mathbb{R}$ is introduced. The corresponding augmented Lagrangian takes the form:

$$\mathcal{L}(\mathbf{q}^{(e)}(t_{k+1}), \beta) = \left\| \mathbf{Q}^{(\Omega)}(t_{k+1}) - \mathbf{A} \cdot \mathbf{q}^{(e)}(t_{k+1}) \right\|^2 + \beta \left(\mathbf{Q}^{(\Omega)}(t_{k+1}) - \mathbf{w}^{\mathrm{T}} \mathbf{q}^{(e)}(t_{k+1}) \right). \tag{4.12}$$

The first-order optimality conditions yield the following linear block system:

$$\begin{bmatrix} 2\mathbf{A}^{\mathrm{T}}\mathbf{A} & -\mathbf{w} \\ \mathbf{w}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}^{(e)}(t_{k+1}) \\ \beta \end{bmatrix} = \begin{bmatrix} 2\mathbf{A}^{\mathrm{T}} \\ \Delta\xi^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^{(\Omega)}(t_{k+1}) \\ \mathbf{Q}^{(\Omega)}(t_{k+1}) \end{bmatrix}, \tag{4.13}$$

which defines the reconstruction operator $\mathbf{C}$ for recovering the high-order polynomial coefficients from sub-cell data. This operator can be precomputed to reduce computational overhead:

$$\mathbf{C} = \begin{bmatrix} 2\mathbf{A}^{\mathrm{T}}\mathbf{A} & -\mathbf{w} \\ \mathbf{w}^{\mathrm{T}} & 0 \end{bmatrix}^{-1}. \tag{4.14}$$

It is important to note that neither the forward projection (4.10) nor the reconstruction (4.13) inherently preserves the positivity of physical quantities such as the granular material thickness h, especially in regions where h approaches zero. Since positivity is a critical condition for admissibility (4.1), a solution adjustment step is applied during the forward projection phase:

$$\mathbf{Q}^{(\Omega)}_{\text{adjusted}}(t_k) = \begin{cases} \mathbf{0}, & h(x, t_k) \geq h_{\text{Threshold}} \quad \forall x \in (\Omega), \\ \mathbf{Q}^{(\Omega)}(t_k), & \text{otherwise}, \end{cases} \tag{4.15}$$

where $h_{\text{Threshold}}$ is the prescribed threshold used in (4.1). This ensures that nonphysical negative values do not propagate through the reconstruction process or subsequent time integration.

4.2. Numerical Implementation Aspects Specific to the Landslide Model

While the core ADER-DG scheme provides a general high-order framework for solving hyperbolic systems, certain challenges arise from the specific structure of the landslide

model (2.20). In particular, the presence of stiff source terms and the need for wetting-drying treatment near the moving front. This section outlines the tailored modifications introduced to address these difficulties and ensure the robustness and stability of the simulation in landslide scenarios.

4.2.1. Wetting/Drying Treatment in Avalanche Simulations

Accurate numerical handling of marginal thickness regions ($h \to 0$) represents a critical challenge in landslide flow simulations. These dry-state transition zones occur naturally at propagating flow fronts and along lateral margins, where improper numerical treatment can induce non-physical negative thickness values $h < 0$, violating model hyperbolicity, and solver divergence through unconstrained velocity growth.

Threshold-Activated Flux Limiting

The adopted stabilization strategy implements conditional suppression of momentum exchange mechanisms when local thickness falls below a critical threshold h_{dry} (which can be the same as $h_{Threshold}$ used in (4.1)). Specifically:

$$\text{Flux}_{\text{interface}} = \begin{cases} 0, & \text{if} \quad \min(h_L, h_R) < h_{\text{dry}} \\ \text{Rusanov flux}, & \text{otherwise.} \end{cases} \tag{4.16}$$

By effectively deactivating these contributions in near-dry regions, the method prevents the generation of non-physical states, such as negative depths, and ensures stability.

Threshold Parametrization

The activation threshold h_{dry} balances physical fidelity against numerical robustness:

- Set $h_{\text{dry}} \in [10^{-6}, 10^{-3}]\,\text{m}$ - orders below characteristic flow depth (1 m);
- Calibrated to prevent truncation of physically relevant thin flows;
- Provides a safety margin against round-off error accumulation.

For more advanced wetting and drying techniques, the reader is referred to [23].

4.2.2. Handling Stiff Source Terms in Landslide Dynamics

A source term $S(Q)$ is classified as stiff if small perturbations in the state variables Q induce disproportionately large changes in the source contribution, significantly influencing the system dynamics. Mathematically, stiffness arises when the Jacobian of the source term dominates that of the flux function:

$$\left\| \frac{\partial S}{\partial Q} \right\| \gg \left\| \frac{\partial F}{\partial Q} \right\|, \tag{4.17}$$

where the norm is typically the spectral norm.

Stiff source terms induce rapid temporal variations in the solution, which can destabilize numerical schemes, force prohibitively small time-steps, or fail to resolve critical transients. High-order time integration methods, such as the ADER-DG framework, mitigate these issues by achieving unified high-order accuracy in space and time within a single update step [24].

Source Term Decomposition for Granular Flows

In the context of landslide simulations, the source term in (3.2) is naturally split into two components:

$$S(Q) = -\underbrace{\begin{pmatrix} 0 \\ g\dfrac{hu}{\sqrt{(hu)^2+(hv)^2}}\mu h\cos\varphi \\ g\dfrac{hv}{\sqrt{(hu)^2+(hv)^2}}\mu h\cos\varphi \end{pmatrix}}_{S^{\text{Smooth}}(Q)} - \underbrace{\begin{pmatrix} 0 \\ \dfrac{g}{\xi}\dfrac{hu}{h^2}\sqrt{(hu)^2+(hv)^2} \\ \dfrac{g}{\xi}\dfrac{hv}{h^2}\sqrt{(hu)^2+(hv)^2} \end{pmatrix}}_{S^{\text{Stiff}}(Q)}, \tag{4.18}$$

where $S^{\text{Smooth}}(Q)$ corresponds to Coulomb-type friction and varies smoothly, while $S^{\text{Stiff}}(Q)$ models a turbulent-like drag effect and may exhibit stiffness.

Stiffness Characterization

The Jacobian of the stiff component is given by:

$$\frac{\partial S^{\text{Stiff}}}{\partial Q} = \frac{g}{\xi}\begin{pmatrix} 0 & 0 & 0 \\ 2\dfrac{hu}{h^3}\sqrt{(hu)^2+(hv)^2} & \dfrac{2(hu)^2+(hv)^2}{h^2\sqrt{(hu)^2+(hv)^2}} & \dfrac{(hu)(hv)}{h^2\sqrt{(hu)^2+(hv)^2}} \\ 2\dfrac{hu}{h^3}\sqrt{(hu)^2+(hv)^2} & \dfrac{(hu)(hv)}{h^2\sqrt{(hu)^2+(hv)^2}} & \dfrac{(hu)^2+2(hv)^2}{h^2\sqrt{(hu)^2+(hv)^2}} \end{pmatrix}, \tag{4.19}$$

with eigenvalues:

$$s_1 = 0, \quad s_2 = \frac{g}{\xi}\frac{\sqrt{(hu)^2+(hv)^2}}{h^2}, \quad s_3 = 2\frac{g}{\xi}\frac{\sqrt{(hu)^2+(hv)^2}}{h^2}. \tag{4.20}$$

The dominant eigenvalue $s = s_3$ defines the most restrictive time scale introduced by the stiff source term. To account for this constraint in the time-step computation, the standard CFL condition (3.69) is modified by incorporating the maximum source eigenvalue. The effective wave speed is then defined as:

$$\lambda_{\text{eff}} = \max(\max(|\lambda|), \Delta x_n \cdot \max(|s_3|)), \tag{4.21}$$

where the maxima are taken over all quadrature nodes in all elements, and Δx_n - the face size - is needed to match the dimensions of these physical quantities. This modification, derived in this work specifically to model (2.20), ensures stability of the high-order ADER-DG scheme while avoiding unnecessarily small time-steps.

Finite Volume Stabilization

The modified CFL condition alone is not sufficient to stabilize the stiff source term within the first-order FVM used in the a-posteriori sub-cell limiting. To enhance robustness, a stabilization strategy inspired by [7] is adopted, in which the stiff source term is evaluated at sub-cell interfaces rather than at cell centers.

Specifically, the FVM update includes an interface-based source contribution

$$F^{(*)} = \sum_{f=0}^{N_{\text{faces}}-1} \mathbf{n}_f \cdot \left[\int_{\partial_f [0,1]^{\text{dim}}} \tilde{F}_f^{(*)}(r) \varphi_m^{(*)}(r) dS \right] + \Delta x_f \cdot \left(\text{NCP}_f + S_f^{\text{Stiff}}(Q) \right). \tag{4.22}$$

The stiff source term at face f, denoted by $S_f^{\text{Stiff}}(Q)$, is computed using averaging of the adjacent sub-cell values:

$$S_f^{\text{Stiff}}(Q) = \frac{1}{2} \left(S^{\text{Stiff}}(Q^L) + S^{\text{Stiff}}(Q^R) \right). \tag{4.23}$$

This interface-based treatment helps to smooth local variations and prevents the onset of instabilities caused by abrupt source term gradients.

5. Implementation Framework: ExaHyPE 2 Customization

The numerical simulations presented in this work are implemented within ExaHyPE 2[1] (Exascale Hyperbolic PDE Engine), a high-performance computational framework designed for solving systems of hyperbolic PDEs. As a solver construction platform, ExaHyPE 2 provides a flexible infrastructure for modeling linear and nonlinear problems with high accuracy while supporting scalable execution across diverse computing architectures, from laptops to supercomputers. A defining feature of the engine is its requirement for users to explicitly specify problem-dependent components, such as flux functions, source terms, and NCP, ensuring full customization to the physical system under study.

ExaHyPE 2 employs a code-generation architecture, where tailored C++ solver code is automatically synthesized based on user-defined problem specifications. The framework leverages a Python-based interface to configure and drive this code-generation process, eliminating the need for generic abstractions common in universal solvers. For instance, this approach enables memory optimizations, loop unrolling for fixed data sizes, and the removal of redundant runtime condition checks, thereby enhancing computational efficiency. The generated code integrates with the Peano framework, which provides dynamically adaptive Cartesian meshes and a space-filling Peano curve traversal scheme to minimize communication overhead between computational patches.

To configure a simulation, users must define key parameters, including the solver type, computational domain, initial and boundary conditions, and numerical functions such as fluxes. A Python script automates the generation of solver-specific C++ code, assembles the build system, and compiles the executable. While low-level modifications at the C++ stage remain possible, most configurations can be completed entirely within the Python layer. This design abstract the underlying algorithmic complexity, allowing users to focus on the physical and numerical aspects of their problem without exposure to low-level implementation details.

For the landslide modeling undertaken in this study, an a-posteriori sub-cell limiting solver architecture is used. This architecture integrates three core components

1. A high-order regular solver (ADER-DG);
2. A robust first-order FVM acting as a limiter solver;
3. A supervisor solver is responsible for synchronizing the two solvers and enforcing solution validity.

To address the specific challenges of landslide dynamics, the functionality of ExaHyPE 2 was extended. The subsequent sections provide a detailed exposition of each solver component, their interplay within ExaHyPE 2 infrastructure, and the modifications implemented to support landslide-specific physics.

[1] See: `https://gitlab.lrz.de/hpcsoftware/Peano/-/tags/2025ExaHyPELandslideStiffSourceTerm`

5.1. General Workflow for Solver Configuration and Project Generation via Python API in ExaHyPE 2

The ExaHyPE 2 framework provides a high-level Python interface for the structure and modular configuration, generation, and compilation of simulation applications. This approach abstracts low-level implementation details while enabling the rapid development of problem-specific solvers. The workflow consists of four main stages: solver definition, project initialization, simulation configuration, and code generation. The following overview outline the essential steps required to generate most applications, without attempting to exhaustively document all available capabilities.

5.1.1. Solver Definition

The initial stage involves selecting and configuring the numerical solvers that represent the underlying physical system. The key steps include:

- **Solver selection:** Choosing appropriate numerical methods, such as high-order ADER-DG or first-order FVM;
- **Parameter specification:** Defining solver-specific parameters, including the unknown variables (either the number of them or the dictionary structure with names), order, time relaxation parameter (CFL condition), and many more;
- **Mathematical formulation:** Specifying problem-dependent functions, including physical fluxes, source and NCP terms, and admissibility criteria.

These elements collectively define the mathematical model and encode the physical behavior of the target system.

5.1.2. Project Initialization

The second stage establishes the structural foundation of the simulation application:

- **Namespace declaration:** A unique namespace is assigned to the generated C++ code to ensure modularity and avoid naming conflicts;
- **Output directory setup:** The target directory for code generation and result storage is defined;
- **Executable naming:** The name of the final compiled binary is specified.

This phase creates the scaffolding for subsequent solver integration and parameterization.

5.1.3. Simulation Configuration

The third stage links solvers to the project and defines global simulations parameters:

- **Solver registration:** Associating configured solvers (e.g. ADER-DG, FVM, limiter) with the project;
- **Global simulation parameters:** Setting start/end times, output intervals for solution snapshots, specifying spatial domain characteristics such as size and offset, and so on;

- **Parallel execution:** Configuring load-balancing strategies and distributed-memory parallelism parameters.

These settings ensure alignment with physical requirements and computational efficiency goals.

5.1.4. Code Generation and Compilation

The final stage automates the transition from configuration to executable:

- **Framework integration:** Linking to the ExaHyPE 2 and Peano libraries by specifying their root directories;
- **Build parameters selection:** For example, choosing between optimized `Release` or debuggable `Debug` compilation modes;
- **Code synthesis:** Generating tailored C++ solver code from the Python configuration, followed by compilation into a binary.

This end-to-end workflow minimizes manual intervention, enabling rapid iteration and adaptation of solvers to diverse physical systems.

5.2. ADER-DG Solver in ExaHyPE 2

This section presents the configuration and internal execution model of the ADER-DG solver in the ExaHyPE 2 framework. The discussion is structured into two parts: the high-level configuration of the solver using the Python API, and the solver's operational logic governed by a sequence of actions. This overview focuses on essential aspects relevant for practical usage and does not aim to provide an exhaustive description.

5.2.1. ADER-DG Solver Configuration via Python API

The ADER-DG solver is instantiated using the Python wrapper

`exahype2.solvers.aderdg.GlobalAdaptiveTimeStep`.

This class provides an abstraction layer for defining numerical settings, physical parameters, and computational constraints in a concise and modular fashion. It translates high-level input into optimized C++ code, abstracting implementation details such as memory layout and numerical flux integration.

Key Configuration Parameters

The constructor of the `GlobalAdaptiveTimeStep` class accepts several key arguments:

1. `name`:
   A unique identifier for the solver, which determines the corresponding C++ class name in the generated code. For example, setting

   `name="ADERDGSolver"`

   results in a C++ class with the name "ADERDGSolver". This naming convention aids code clarity but does not influence the logic of the solver.

2. `order`:
   Specifies the polynomial degree of the ADER-DG basis (3.6), (3.49), i.e. $N = n_p - 1$, where n_p is the number of degrees of freedom per spatial direction. Increasing the order enhances numerical accuracy at the cost of additional computational effort.
3. `unknowns`:
   A dictionary describing the model's variables. For example, For instance,

   $$\texttt{unknowns} = \{"h" : 1, "hu" : 2\}$$

   declares a scalar h (stored at index 0) and a vector hu with two components (indices 1 and 2), corresponding to three total unknowns in the system. The keys determine variable names, while the values define their dimensions.
4. `min_cell_h` and `max_cell_h`:
   The minimum and maximum allowable cell sizes for adaptive mesh refinement (AMR). Uniform grid resolution is enforced when these values are equal, which is used in the present work.
5. `time_step_relaxation`:
   A relaxation coefficient related to the CFL condition (see (3.69)) that controls the time-step size for stability.
6. **Problem-specific functions**:
   User-defined implementations for physical fluxes, eigenvalues, source terms, boundary conditions, and similar terms are passed either as inline C++ code snippets from the Python layer or indicated with

   `PDETerms.User_Defined_Implementation`,

   when the implementation is provided directly in the C++ generated code.

Workflow Example

The minimal configuration for simulating the landslide (2.20) may be expressed as follows:

```
aderdg_solver = exahype2.solvers.aderdg.GlobalAdaptiveTimeStep(
    name                    ="ADERDGSolver",
    order                   =4,
    unknowns                ={"h": 1, "hu": 2, "z": 1},
    auxiliary_variables     ={},
    min_cell_h              =0.1,
    max_cell_h              =0.1,
    time_step_relaxation    =0.5,
    flux                    =PDETerms.User_Defined_Implementation,
    eigenvalues             =PDETerms.User_Defined_Implementation,
    initial_conditions      =PDETerms.User_Defined_Implementation,
    boundary_conditions     =PDETerms.User_Defined_Implementation,
    ncp                     =PDETerms.User_Defined_Implementation,
    diffusive_source_term   =PDETerms.User_Defined_Implementation
)
aderdg_solver.add_kernel_optimisations(
```

```
    riemann_solver_implementation=PDETerms.
        User_Defined_Implementation
)
```

Listing 5.1: ADER-DG Solver Configuration via Python API in ExaHyPE 2.

In this example, all core numerical terms are marked with

`PDETerms.User_Defined_Implementation`,

indicating that the corresponding C++ routines must be manually provided after the generation. The method

`add_kernel_optimisations`

is used to inject a custom Riemann solver into the generated code.

It is important to note that ExaHyPE 2 automatically pregenerates several auxiliary matrices, whose dimensions depend on the selected order. These include:

- The temporal basis matrix at initial time $\mathbf{\Phi}_t^{(e)}(0)$ (3.29);
- Quadrature based weight matrices $\mathbf{W}_{\text{sp}}$, $\mathbf{W}_{\text{t}}$ defined in (3.55) and (3.25), respectively;
- The spatial derivative matrix $\mathbf{K}_{\text{x}}$ (3.27);
- The temporal matrix $\mathbf{K}_{\text{t}}$ (3.26);
- The basis differentiation matrix in each coordinate direction $\mathbf{D}_{\text{x}_\text{d}}^{(e)}$ (3.33).

These matrices are integral to the predictor and corrector phases of the ADER-DG method, enabling efficient evaluation of integrals and derivatives within the local reference space.

Flux Implementation for Landslide Dynamics

The `flux` function in Listing 5.1 implements the computation of physical fluxes for mass and momentum in the context of landslide simulations governed by the depth-averaged SWE with the Voellmy-Salm friction model (2.20). This routine is essential for evaluating the divergence terms in the governing equations (3.1), and must account for both the advection of momentum and the nonlinear nature of granular flow.

The implementation shown in Algorithm 1 includes an important dry cell handling strategy to ensure stability near the wetting/drying interface. Specifically, if the material thickness $Q[h]$ drops below a user-defined threshold h_{dry}, all flux components are set to zero, effectively freezing the state of the cell and avoiding the appearance of spurious velocities or negative thickness values.

When the cell is considered wet, the function computes the physical fluxes in the normal direction n. The normal velocity V_n is obtained by projecting the momentum onto the interface normal, and flux components are derived accordingly. This includes both convective terms and the pressure-like source term in the normal momentum flux $F[hu + n]$, modeled via the hydrostatic approximation $\frac{1}{2}gh^2$. Notably, elevation $Q[z]$ is considered static and thus excluded from the flux terms.

A key strength of ExaHyPE 2 is that users need not manually manipulate the sign of face normals or apply scaling factors related to mesh spacing. The framework internally

integrates geometric transformations and surface integrals into the numerical kernel. This allows users to express fluxes directly in physical space, greatly simplifying implementation and reducing the risk of coordinate handling errors.

The approach encapsulated in this function forms the backbone of both ADER-DG and FV schemes in ExaHyPE 2, as flux computations directly contribute to the predictor and corrector steps responsible for time evolution.

Algorithm 1: Flux computation for landslide dynamics (3.2) with dry cell handling (4.16).

Input:

- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- Spatial coordinate x, grid spacing dx, current time t, time-step dt
- Normal direction $n \in \{0, 1\}$ (0: x-axis, 1: y-axis)

Output: Flux vector F (components: $F[h], F[hu], F[hu+1], F[z]$)

```
// Compute physical fluxes in direction n
Function flux(Q, x, dx, t, dt, n, F):
    if Q[h] > h_dry then
        // Compute normal velocity
        V_n ← Q[hu + n]/Q[h]
        // Compute flux components
        F[h] ← Q[hu]
        F[hu] ← V_n · Q[hu]
        F[hu + 1] ← V_n · Q[hu + 1]
        F[hu + n] ← 0.5 · g · Q[h]^2
        // No flux for elevation
        F[z] ← 0.0
    else
        // Dry cell:  zero flux
        for i ← 0 to 3 do
            F[i] ← 0.0
```

Non-Conservative Product Term for Landslide Dynamics

The NCP term $\mathcal{B}(Q) \cdot \nabla Q$ appears in the mathematical formulation of landslide dynamics to account for momentum exchange between the granular flow and the underlying topography. Capturing this interaction is essential for physically accurate simulations, particularly in regions with strong elevation gradients. The implementation provided in Algorithm 2 follows the formulation given in (3.2), with special treatment of dry cells according to (4.16) to ensure numerical robustness.

For $n_p = 1$ (i.e., first-order discretization), the NCP term is computed during the corrector phase at cell interfaces, as described in (3.43). In this case, *deltaQ* in Algorithm 2 represents a central difference approximation computed from the left and right states, while Q is taken as their average. ExaHyPE 2 automatically scales *deltaQ* by the grid spacing dx, together

with the flux term (see (3.64)), as part of the discrete divergence operator. Thus, no explicit division by dx is required in the implementation.

For the higher-order ADER-DG method with $n_p > 1$, the NCP term is evaluated during the predictor phase within each cell. In this case, *deltaQ* corresponds to the local spatial derivative reconstructed from the predictor expansion (3.61), and Q denotes the nodal predictor solution values at quadrature points. The NCP term is computed independently in each spatial direction n and accumulated into the volumetric source term. Since *deltaQ* already represents a derivative in this setting, no scaling by dx is needed; ExaHyPE 2 handles this automatically depending on the computation context.

Algorithm 2: NCP computation for landslide dynamics (3.2) with dry cell handling (4.16).

Input:
- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- *deltaQ* represents spatial derivative along the face-normal direction (components $deltaQ[h], deltaQ[hu], deltaQ[hu+1], deltaQ[z]$);
- Spatial coordinate x, grid spacing dx, current time t, time-step dt
- Normal direction $n \in \{0, 1\}$ (0: x-axis, 1: y-axis)

Output: NCP vector $BTimesDeltaQ$ (components: $BTimesDeltaQ[h], BTimesDeltaQ[hu], BTimesDeltaQ[hu+1], BTimesDeltaQ[z]$)

```
Function nonconservativeProduct(Q, deltaQ, x, dx, t, dt, n, BTimesDeltaQ):
    // Initialize all components
    for i ← 0 to 3 do
        BTimesDeltaQ[i] ← 0.0
    if Q[h] > h_dry then
        // Apply NCP term in normal direction
        BTimesDeltaQ[hu + n] ← g · Q[h] · deltaQ[z]
```

Source Term for Landslide Dynamics

The source term $S(Q)$ in landslide dynamics models momentum dissipation resulting from Coulomb friction (2.12) and turbulent-like drag (2.17) - both of which are essential for the landslide model implemented (2.20) in this work. The standard implementation in ExaHyPE 2 typically uses the interface `source_term`, which operates solely on the state vector Q. However, this functionality was extended in the present work through the introduction of a customized function interface, `diffusive_source_term`, enabling access to spatial derivatives of the solution. This extension was specifically developed in this work to support the elevation of the local slope angle α Figure 2.3b, which is necessary for evaluating the Coulomb friction term (2.12).

On the C++ layer, this user-defined source function is implemented as `diffusive AlgebraicSource` in Algorithm 3.

Algorithm 3: Source computation for landslide dynamics (3.2) with dry cell handling (4.16).

Input:
- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- Gradient *deltaQ*: Full gradient tensor stored as $[\partial_x h, \partial_x hu, \partial_x hv, \partial_x z, \partial_y h, \partial_y hu, \partial_y hv, \partial_y z]$;
- Spatial coordinate x, grid spacing dx, current time t, time-step dt

Output: Source term vector S (components: $S[h], S[hu], S[hu+1], S[z]$)

Function `diffusiveAlgebraicSource(`Q, *deltaQ*, x, dx, t, dt, n, S`):`
  `// Initialize all components`
  for $i \leftarrow 0$ **to** 3 **do**
    $S[i] \leftarrow 0.0$
  if $Q[h] > h_{dry}$ **then**
    `// Compute momentum magnitude and slope components`
    $momentumSQR \leftarrow Q[hu]^2 + Q[hu+1]^2$
    $momentum \leftarrow momentumSQR^{1/2}$
    `//` $\partial z/\partial x$
    $gradZX \leftarrow deltaQ[z]$
    `//` $\partial z/\partial y$
    $gradZY \leftarrow deltaQ[z+4]$
    `// Local slope angle`
    $cF \leftarrow (gradZX^2 + gradZY^2)^{-1/2}$
    if $momentum/Q[h] > 0.0$ **then**
      `// Coulomb friction and turbulent drag`
      $coulombFriction \leftarrow -g \cdot \mu \cdot cF \cdot Q[h]$
      $turbulentDrag \leftarrow -g \cdot momentumSQR/(\xi \cdot Q[h]^2)$
      `// Project forces onto velocity direction`
      $S[hu] \leftarrow Q[hu] \cdot (coulombFriction + turbulentDrag)/momentum$
      $S[hu+1] \leftarrow Q[hu+1] \cdot (coulombFriction + turbulentDrag)/momentum$

The elevation gradient $\nabla z = (\partial z/\partial x, \partial z/\partial y)$ is derived from the full solution gradient *deltaQ*, computed using the ADER-DG predictor expansion (3.61). This enables precise evaluation of local slopes α encoded via the factor cF in Algorithm 3. By leveraging spatial derivative information in the source term, this implementation allows consistent and physically informed frictional damping even on non-uniform topography - an essential feature for realistic landslide modeling on complex terrains.

Eigenvalues Computation for Landslide Dynamics

The eigenvalue evaluation implemented in this work determines the effective wave propagation speed λ_{eff} as defined in (4.21). This value governs the adaptive time-step size through the CFL condition (3.69). The corresponding implementation is shown in Algorithm 4.

Unlike classical hyperbolic solvers where eigenvalues for the time-step size update (3.69)

are computed solely from flux Jacobians, this formulation incorporates both advective and source-related wave speeds. This extension is essential in landslide modeling, where stiff source terms (2.17) can dominate local dynamics. By including a source-based contribution, the computed eigenvalue stabilizes the scheme in stiff regions while maintaining high efficiency in smooth flow zones.

The Algorithm 4 first checks for a dry state by evaluating whether the thickness $Q[h]$ exceeds the dry threshold. If so, it computes the local advective velocity V_n and wave speed c in the direction normal to the interface, combining them to estimate the classical flux-based eigenvalue $s_{\max}$. Meanwhile, it calculates the contribution from the turbulent drag term (stiff source), yielding a source-based speed s_{source}. The final effective eigenvalue is chosen as the maximum of the two. In dry cells, the eigenvalue is set to zero, ensuring numerical stability.

Algorithm 4: Maximum absolute eigenvalue computation for landslide dynamics (3.2) with dry cell handling (4.16).

Input:

- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- Spatial coordinate x, grid spacing dx, current time t, time-step dt
- Normal direction $n \in \{0, 1\}$ (0: x-axis, 1: y-axis)

Output: Maximum effective eigenvalue λ_{eff}

Function `maxEigenvalue`(Q, *deltaQ*, x, dx, t, dt, n)**:**
  if $Q[h] > h_{dry}$ **then**
    `// Compute normal velocity`
    $V_n \leftarrow Q[hu+n]/Q[h]$
    $c \leftarrow (g \cdot Q[h])^{1/2}$
    $smax \leftarrow \max(|V_n - c|, |V_n + c|)$
    `// Compute "source" eigenvalue`
    $momentumSQR \leftarrow Q[hu]^2 + Q[hu+1]^2$
    $momentum \leftarrow momentumSQR^{1/2}$
    $sSource \leftarrow 2.0 \cdot g \cdot momentum/(\xi \cdot Q[h] \cdot Q[h])$
    `// Return "effective" eigenvalue`
    return $\max(smax, h[n] \cdot sSource)$
  return 0.0

The Custom Riemann Solver for Landslide Dynamics

The Riemann solver plays a critical role in resolving discontinuities at cell interfaces within Finite Volume and ADER-DG. For landslide dynamics, a custom Riemann solver is implemented to address two key limitations of ExaHyPE 2 default infrastructure:

1. **Auxiliary variable handling:** The elevation term z is treated as a non-evolving variable, requiring explicit suppression of its flux contributions.
2. **Eigenvalue consistency:** The solver must use eigenvalues derived solely from the flux Jacobian, independent of source-term-induced modifications as in Algorithm 4, to

ensure proper hyperbolic wave propagation.

Therefore, the implementation Algorithm 5 addresses these issues for $n_p > 1$.

Algorithm 5: Numerical Riemann solver (3.41) for (2.20) with local wave speed estimate. Implements Rusanov flux for $n_p > 1$. Flux arrays are modified in place.

Input:
- Q_L, Q_R: Left and right states at quadrature nodes;
- F_L, F_R: Local physical fluxes for Q_L and Q_R (input/output);
- t: Current time, dt: Time-step size;
- x: Coordinate of face center, h: Cell size;
- Normal direction $n \in \{0, 1\}$ (0: x-axis, 1: y-axis)

Output: The solution of Riemann problem F_L, F_R (Rusanov flux (3.41))

Function `riemannSolver`(F_L, F_R, Q_L, Q_R, t, dt, x, h, n)**:**
  `// Compute averaged left and right states`
  for $i \leftarrow 0$ **to** 3 **do**
    for $xy \leftarrow 0$ **to** n_p **do**
      $Q_{\text{avg},L}[i] \leftarrow Q_{\text{avg},L}[i] + w[xy] \cdot Q_L[xy \cdot 4 + i]$
      $Q_{\text{avg},R}[i] \leftarrow Q_{\text{avg},L}[i] + w[xy] \cdot Q_R[xy \cdot 4 + i]$
  `// Wave speed estimation based on the flux Jacobian`
  $s_L \leftarrow 0.0$
  if $Q_{avg,L}[h] > h_{dry}$ **then**
    $u_L \leftarrow Q_{\text{avg},L}[hu + n]/Q_{\text{avg},L}[h]$
    $c_L \leftarrow (g \cdot Q_{\text{avg},L}[h])^{1/2}$
    $s_L \leftarrow \max(|u_L - c_L|,\ |u_L + c_L|)$
  $s_R \leftarrow 0.0$
  if $Q_{avg,R}[h] > h_{dry}$ **then**
    $u_R \leftarrow Q_{\text{avg},R}[hu + n]/Q_{\text{avg},R}[h]$
    $c_R \leftarrow (g \cdot Q_{\text{avg},R}[h])^{1/2}$
    $s_R \leftarrow \max(|u_R - c_R|,\ |u_R + c_R|)$
  $s_{\max} \leftarrow \max(s_L, s_R)$
  `// Solve Riemann problem (modifies inputs)`
  for $xy \leftarrow 0$ **to** 4 **do**
    `// Rusanov flux update`
    for $j \leftarrow 0$ **to** 2 **do**
      $i \leftarrow xy \cdot 4 + j$
      $F_L[i] \leftarrow 0.5 \cdot (F_L[i] + F_R[i] + s_{\max} \cdot (Q_L[i] - Q_R[i]))$
      $F_R[i] \leftarrow F_L[i]$
    `// Ignore flux flow for elevation`
    $i \leftarrow xy \cdot 4 + z$
    $F_L[i] \leftarrow 0.0$
    $F_R[i] \leftarrow 0.0$

5.2.2. ADER-DG Solver Execution Logic in ExaHyPE 2

The time-stepping procedure in ExaHyPE 2 is structured as a sequence of multiple sweeps over the computational grid per time-step. During each sweep, the solver operates in a specific internal state that governs which computational action is executed next. These states form a discrete control structure, with transitions defined by internal logic in the generated C++ solver code.

Each grid sweep is initialized by a call to the `startTimeStep()` method of the generated solver class. This method determines the next solver state and thereby the next action that will be performed by the code. Among the sweeps, one is marked as the final sweep for the current time-step. This is indicated by the method `isLastSweepOfGrid()`, which returns `true` when the current sweep concludes the time-stepping cycle. In this case, the solver invokes the `finishTimeStep()` method, which computes a new time-step size for the next time layer based on the Courant-Friedrichs-Lewy (3.69) condition and the maximum effective wave speed observed during the step.

Crucially, this control logic is encapsulated within the generated code and remains transparent to the user. From a user's perspective, the solver setup is typically limited to high-level configuration via the Python API. The underlying mechanism of state transitions, action sequencing, and time-step adjustment is managed automatically by the framework.

This section describes the principal ADER-DG solver states, the corresponding actions generated during solver setup, and the rules that govern their execution within the time-stepping cycle.

Primary ADER-DG Solver States

Among the internal states that govern ADER-DG solver behavior in ExaHyPE 2, two are of main importance:

- `InitialPrediction`;
- `TimeStep`.

These states correspond to the core computational phases of the ADER-DG algorithm and collectively define one full time-step of the ADER-DG. The method `isLastSweepOfGrid()` returns `true` when the ADER-DG solver is in the `TimeStep` state, indicating that the current sweep is the final one of the time-step cycle. Upon completing this sweep, the solver calls `finishTimeStep()`, which updates the global time-step size according to the CFL condition (3.69) and the global simulation time.

Solver states alternate deterministically: When the solver is in the `TimeStep` state and the method `startTimeStep()` is invoked, the state transitions to `InitialPrediction`. Conversely, starting from `InitialPrediction`, the next call to `startTimeStep()` transitions the state to `TimeStep`.

While the ADER-DG solver supports additional internal states (e.g., for output of the solution, grid initialization and etc.), the two described above are sufficient to capture the fundamental time-stepping loop as used in this work.

The implementation of this state machine, including all transitions and their effect on solver behavior, is fully encapsulated within the generated C++ code. The design abstracts low-level control flow and simplifies user-side configuration.

ADER-DG Solver States: `InitialPrediction`

The `InitialPrediction` state initiates the ADER-DG time-stepping cycle, triggering the `Predictor` action. This phase prepares the local space-time predictor solution and assembles volumetric (local) contributions for the subsequent corrector step. The following operations are performed cell-locally:

1. **State preservation:** The current solution vector Q (resulting from the previous corrector step) is stored in Q_{old}, ensuring data consistency during the predictor computation.
2. **Predictor initialization:** An initial guess for the predictor solution is computed using the static approximation strategy described in (3.24).
3. **Local space-time predictor solution:** The nonlinear system governing the predictor (3.62) is solved iteratively using a simple fixed-point scheme proposed in [12]. Each iteration involves:
   - Spatial derivatives of the state vector are computed locally using (3.61);
   - Numerical fluxes are evaluated within each element according to (3.14), following the concrete implementation in Algorithm 1;
   - Source terms are computed locally using the algorithm described in Algorithm 3;
   - The NCP terms are assembled based on local spatial gradients, as outlined in Algorithm 2, and are absorbed into the source term.

   This process is entirely element-local and requires no communication with neighboring cells.
4. **Temporal averaging of flux and source predictors:**
   Once the local predictor converged, a time-averaged flux and source predictors are computed according to (3.45). This averaged state is used in the corrector to integrate the solution over the time-step and reduce the number of Riemann solver calls to one for each face.
5. **Boundary flux evaluation:** Flux predictors are evaluated at the cell boundaries by projecting the predictor solution onto the face nodes. The multidimensional extension of (3.14) is used where the spatial coordinates of the face nodes are substituted to obtain directional flux values.
6. **Volume integral assembly:** All volumetric terms from the corrector (3.63) are evaluated and assembled. The only term deferred to the `TimeStep` state is the surface integral, corresponding to Riemann-based flux corrections across cell interfaces.

This state concludes the local predictor-corrector preparation and sets the stage for inter-element coupling in the next solver state.

ADER-DG Solver States: `TimeStep`

In the `TimeStep` state, the ADER-DG scheme completes the corrector phase by executing the `Correction` action. This action consists of two parts:

1. **Interface corrections:** On cell interfaces, the Riemann problem is solved using left and right extrapolated values obtained from the predictor. This yields numerical fluxes that finalize the surface integral contributions in (3.63), as implemented in Algorithm 5.
2. **Stability enforcement:** Within each cell, the maximum effective wave speed λ_{eff} is computed at each node. This value is essential for the calculation of the time-step size that maintains numerical stability and is evaluated according to the procedure in Algorithm 4.

Once all corrections have been applied, the solver invokes `finishTimeStep()`. This method utilizes the maximum value of λ_{eff} across the domain to compute the new global time-step size based on the CFL condition (3.69), and advances the simulation time accordingly. The `TimeStep` state finalizes all numerical contributions of the current iteration and sets the stage for transitioning back to `InitialPrediction` in the next time-step.

5.3. Finite Volume Solver in ExaHyPE 2

This section outlines the configuration and execution model of the FV solver within the ExaHyPE 2 framework. The solver generation process closely resembles that of the ADER-DG solver discussed previously, relying on the same Python-based abstraction layer for specifying solver parameters and generating C++ code. Accordingly, this section emphasizes aspects specific to the FV solver, including its time-stepping behavior, numerical flux handling, and update mechanism. The discussion is limited to core implementation features relevant for practical applications, rather than providing a complete theoretical or architectural description.

5.3.1. Finite Volume Solver Configuration via Python API

The first-order FV solver is configured via the Python wrapper class:

`exahype2.solvers.fv.godunov.GlobalAdaptiveTimeStep`.

While the general workflow mirrors the ADER-DG solver configuration, key distinctions arise from the FV method's structure. The FV solver replaces the ADER-DG `order` parameter with `patch_size`, specifying the number of sub-cells per spatial direction within each Finite Volume cell. This controls the spatial resolution, but the solver remains first-order accurate and does not permit adjustment of the numerical order through configuration.

The `min_volume_h` and `max_volume_h` arguments mirror the ADER-DG `min_cell_h` and `max_cell_h`, defining bounds for adaptive mesh refinement (AMR). In this work, AMR is not utilized; hence, these values are assigned identically to enforce a uniform grid.

Furthermore, the FV solver uses `max_eigenvalue` instead of the `eigenvalues` parameter in ADER-DG. While this change does not alter behavior from the code generation perspective, it reflects a conceptual distinction: in the FV solver, `max_eigenvalue` computes only the characteristic wave speed derived from the flux Jacobian, since it is only used inside the Riemann solver. The effective wave speed, including contributions from source terms, is instead evaluated within the Riemann solver, which is passed directly as a configuration argument - unlike ADER-DG, where custom Riemann solvers are introduced via an explicit optimization step.

Coulomb Source Term for Landslide Dynamics

In the FV solver, only the Coulomb friction (2.12) component is included in the source term (see (4.22)), unlike in the ADER-DG scheme, where both friction (2.12) and turbulent-like drag (2.17) are considered. The standard ExaHyPE 2 source interface was extended as well for the FV solver via the introduction of the `diffusive_source_term` argument in the Python configuration. This extension - implemented specifically in the context of this work - enables access to spatial gradients of the elevation field, which are required to evaluate the local slope angle α (see Figure 2.3b) and compute the Coulomb term.

At the C++ layer, the FV diffusive source term function has the name `diffusiveSourceTerm`, which is different from the analogous function of ADER-DG. A key advantage of the ExaHyPE 2 Python API is that users need only specify the argument `diffusive_source_term` in the configuration; the appropriate method signature and C++ bindings are automatically generated, regardless of the solver type.

Algorithm 6: Coulomb friction source computation in FV solver for landslide dynamics (2.20).

Input:
- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- Gradient $deltaQ$: Full gradient tensor stored as $[\partial_x h, \partial_x hu, \partial_x hv, \partial_x z, \partial_y h, \partial_y hu, \partial_y hv, \partial_y z]$;
- Spatial coordinate x, grid spacing dx, current time t, time-step dt

Output: Source term vector S (components: $S[h], S[hu], S[hu+1], S[z]$)

Function `diffusiveSourceTerm`(Q, $deltaQ$, x, dx, t, dt, n, S)**:**
  `// Initialize all components`
  for $i \leftarrow 0$ **to** 3 **do**
    $S[i] \leftarrow 0.0$
  if $Q[h] > h_{dry}$ **then**
    `// Compute momentum magnitude and slope components`
    $momentum \leftarrow (Q[hu]^2 + Q[hu+1]^2)^{1/2}$
    `//` $\partial z/\partial x$
    $gradZX \leftarrow deltaQ[z]$
    `//` $\partial z/\partial y$
    $gradZY \leftarrow deltaQ[z+4]$
    `// Local slope angle`
    $cF \leftarrow (gradZX^2 + gradZY^2)^{-1/2}$
    if $momentum/Q[h] > 0.0$ **then**
      `// Coulomb friction`
      $coulombFriction \leftarrow -g \cdot \mu \cdot cF \cdot Q[h]$
      `// Project forces onto velocity direction`
      $S[hu] \leftarrow Q[hu] \cdot coulombFriction/momentum$
      $S[hu+1] \leftarrow Q[hu+1] \cdot coulombFriction/momentum$

Non-Conservative Product Term for Landslide Dynamics

To ensure stability of the first-order method, the stiff source term should be computed at cell interfaces, as described in (4.22). However, ExaHyPE 2 does not provide a dedicated interface for source term computation at interfaces. To address this issue, the turbulent-like drag contribution (2.17) is integrated directly into the NCP function, which is executed at interfaces.

To ensure the correct physical scaling, this stiff term is explicitly multiplied by the interface size. Outside the user implementation, ExaHyPE 2 automatically rescales the result so that the *deltaQ* behaves as a finite difference across the interface. The final implementation appears in Algorithm 7.

Algorithm 7: NCP computation in FV solver including turbulent-like drag.

Input:

- State vector Q: $Q[h]$ (granular material thickness), $Q[hu]$ (x-momentum), $Q[hu+1]$ (y-momentum), $Q[z]$ (elevation)
- *deltaQ* represents spatial derivative along the face-normal direction (components $deltaQ[h], deltaQ[hu], deltaQ[hu+1], deltaQ[z]$);
- Spatial coordinate x, grid spacing dx, current time t, time-step dt
- Normal direction $n \in \{0, 1\}$ (0: x-axis, 1: y-axis)

Output: NCP vector $BTimesDeltaQ$ (components: $BTimesDeltaQ[h], BTimesDeltaQ[hu], BTimesDeltaQ[hu+1], BTimesDeltaQ[z]$)

Function `nonconservativeProduct(`Q, $deltaQ$, x, dx, t, dt, n, $BTimesDeltaQ$`):`

```
// Initialize all components
for i ← 0 to 3 do
    BTimesDeltaQ[i] ← 0.0
if Q[h] > h_dry then
    // Compute momentum magnitude and slope components
    momentumSQR ← Q[hu]^2 + Q[hu+1]^2
    momentum ← momentumSQR^(1/2)
    turbulentDrag ← 0.0
    if momentum/Q[h] > 0.0 then
        // Turbulent drag
        V_n ← Q[hu+n]/momentum
        turbulentDrag ← g · dx[n] · V_n · momentumSQR/(ξ · Q[h]^2)
    BTimesDeltaQ[hu+n] ← g · Q[h] · deltaQ[z] + turbulentDrag
```

Custom Riemann Solver Implementation for Landslide Dynamics

In the FV solver, the Riemann solver plays a central role in both computing numerical fluxes and evaluating stiff source and NCP contributions at cell interfaces. The algorithm used in this work is shown in Algorithm 8. This design aligns with the first-order approximation to the source term (4.22).

Algorithm 8: FV Riemann Solver with NCP

Input:
- Left and right states $Q_L, Q_R \in \mathbb{R}^4$
- Cell center coordinate x, mesh size dx, time t, time-step dt
- Normal direction $n \in \{0, 1\}$

Output: Updated fluxes F_L, F_R; maximum effective wave speed λ_{eff}

Function `solveRiemannProblem`(Q_L, Q_R, x, dx, t, dt, n, F_L, F_R)**:**
  `// Compute max wave speed from both sides`
  $s_{\max} \leftarrow \max(\texttt{maxEigenvalue}(Q_L), \texttt{maxEigenvalue}(Q_R))$
  `// Evaluate local fluxes`
  `flux`(Q^L, x, dx, t, dt, n, F^L)
  `flux`(Q^R, x, dx, t, dt, n, F^R);
  `// Compute averaged state and Rusanov flux`
  for $i \leftarrow 0$ **to** 3 **do**
    $Q_{\text{avg}}[i] \leftarrow 0.5 \cdot (Q_L[i] + Q_R[i])$
    $\Delta Q[i] \leftarrow Q_R[i] - Q_L[i]$
    $F_L[i] \leftarrow 0.5 \cdot (F_L[i] + F_R[i] - s_{\max} \cdot \Delta Q[i])$
    $F_R[i] \leftarrow F_L[i]$
  `// NCP contribution`
  `nonconservativeProduct`(Q_{avg}, ΔQ, x, dx, t, dt, n, ncp)
  for $i \leftarrow 0$ **to** 3 **do**
    $F_L[i] \leftarrow F_L[i] + 0.5 \cdot ncp[i]$
    $F_R[i] \leftarrow F_R[i] - 0.5 \cdot ncp[i]$
  `// Ignore elevation flux`
  $F_R[z] \leftarrow 0.0$
  $F_L[z] \leftarrow 0.0$
  `// Compute effective wave speed due to drag`
  $s_L, s_R \leftarrow 0.0$
  if $Q_L[h] > h_{dry}$ **then**
    $h_L \leftarrow \max(Q_L[h], h_{\text{threshold}})$
    $momentum_L \leftarrow (Q_L[hu]^2 + Q_L[hu+1]^2)^{1/2}$
    $s_L \leftarrow (2.0 \cdot g \cdot momentum_L)/(\xi \cdot h_L^2)$
  if $Q_R[h] > h_{dry}$ **then**
    $h_R \leftarrow \max(Q_R[h], h_{\text{threshold}})$
    $momentum_R \leftarrow (Q_R[hu]^2 + Q_R[hu+1]^2)^{1/2}$
    $s_R \leftarrow (2.0 \cdot g \cdot momentum_R)/(\xi \cdot h_R^2)$
  return $\max(s_{\max}, h[n] \cdot \max(s_L, s_R))$

A key feature of the FV solver is that the NCP term is evaluated directly at interfaces This implementation closely mirrors the correction phase of the ADER-DG scheme in the case of first-order discretization ($n_p = 1$), where the solution is sought for piecewise constant functions and the Riemann problem is solved between averaged states at cell interfaces.

The FV Riemann solver also computes the maximum effective wave speed λ_{eff}, which is

used to constrain the time-step size via the CFL condition (3.69). Together, this implementation combines classical Rusanov-type flux computation (3.41) with NCP-based stiff source handling, providing a consistent and physically meaningful treatment of granular landslide dynamics within the FV method.

5.3.2. Finite Volume Solver Execution Logic

The FV solver in ExaHyPE 2 adopts the same core abstraction framework as ADER-DG: it is structured around the concepts of grid sweeps, solver states, and actions. However, due to the inherently simpler structure of the FVM, the execution logic is significantly easier.

In particular, the FV solver operates entirely within a single solver state: `TimeStep`. During each grid sweep, the action `UpdateCell` is executed to carry out all necessary computations for advancing the solution in time. These include the evaluation of fluxes (Algorithm 1, Algorithm 8), source terms (Algorithm 6), NCP (Algorithm 7), and the update of conservative variables.

As a consequence, the method `isLastGridSweep()` always returns `true` for the first-order FV solver described in this section. The full time-step is computed in a single traversal of the grid.

5.4. A-Posteriori Sub-Cell Limiting in ExaHyPE 2

ExaHyPE 2 implements a robust a-posteriori limiting strategy through the coupling of two independently configured solvers, a high-order regular solver, responsible for achieving high accuracy in smooth regions, and a low-order limiting solver, which ensures robustness and stability in regions exhibiting non-smooth behavior such as shocks or discontinuities.

These two solvers are coordinated via an additional component referred to in this work as the supervisor solver. The supervisor solver synchronizes execution between the regular and limiting solvers, manages transitions between them, and encapsulates the limiting logic. It also implements a user-defined admissibility criterion to detect troubled cells and handles the project of solution data between solver layers when transitioning from regular to limited solution modes and vice versa.

This architecture separates high-order accuracy and shock-capturing robustness into modular components while allowing flexible, runtime-controllable coupling within the unified ExaHyPE 2 framework.

5.4.1. The A-Posteriori Sub-Cell Limiting Configuration via Python API

The configuration of the a-posteriori sub-cell limiting solver in ExaHyPE 2 differs fundamentally from single-solver setups, requiring the explicit construction of three interconnected solvers within a unified Python script.

First, an ADER-DG solver with a high-order of accuracy is created. Second, a Finite Volume (FV) solver is defined, with the number of patches `patch_size` must be set to $2 \cdot \texttt{order} + 1$. Furthermore, the size of the cell and CFL number provided in the ADER-DG solver must match the volume size and CFL number of the FV solver to ensure consistency during projection and recomputation. The supervisor solver is then initialized with the ADER-DG and FV solvers as its `regular_solver` and `limiting_solver`, respectively.

A key difference from the standard solver configuration is that all three solvers must be added to the project. Here, the ExaHyPE 2 concept of a solver repository becomes essential. During grid sweeps, repository-level methods such as `startTimeStep()` or `finishTimeStep()` are invoked, which in turn sequentially trigger the corresponding methods of all registered solvers. In addition, ExaHyPE 2 provides mechanisms to directly access individual solvers via the repository during execution. This enables fine-grained control and configuration, allowing user-defined actions to target specific solver behavior when necessary. It is important to ensure that the supervisor solver is added to the project last, so that it can properly manage and coordinate states of the regular and limiting solvers.

Key Configuration Parameters of the Supervisor Solver

The supervisor solver in ExaHyPE 2 is configured via the Python wrapper class

`exahype2.solvers.limiting.PosterioriLimiting`

with several key parameters that define the behavior of the a-posteriori limiting procedure. Among them are the following:

- `number_of_dmp_observables`,
- `dmp_relaxation_parameter`,
- `dmp_differences_scaling`,
- `physical_admissibility_criterion`.

The first three parameters implement the Discrete Maximum Principle (DMP) (4.2), which is applied to the first `number_of_dmp_observables` variables. This principle helps to detect non-physical oscillations by comparing current values to a relaxed range of previous extrema. The parameter `dmp_differences_scaling` corresponds to the scaling factor k in (4.3), while `dmp_relaxation_parameter` represents the threshold value b in the same expression.

It is worth noting that a more fine-grained tuning of the DMP behavior is available via the C++ layer, which is beyond the scope of this work.

The parameter `physical_admissibility_criterion` defines a user-provided function that implements additional, problem-specific conditions for solution admissibility. In this work, it corresponds to the physical admissibility criteria described in (4.1).

In addition to managing admissibility checks, the supervisor solver is also responsible for inter-solver projection. During its generation, the operators A and C - responsible for projecting from ADER-DG to FV and from FV to ADER-DG, respectively - are generated as well, as described in (4.10) and (4.14).

However, the `adjust_solution` callback that adjusts the projected ADER-DG solution on the FV layer, in accordance with (4.15), must be provided as a parameter `adjust_solution` to the FV solver during its configuration as in Algorithm 9.

Algorithm 9: Adjustment of the ADER-DG projection to FV layer.

Input: Spatial coordinate x, cell size dx, current time t, time-step size dt, solution vector Q

Output: Modified solution vector Q

Function `adjustSolution`(x, dx, t, dt, Q)**:**
  if $Q[h] < h_{Threshold}$ **then**
    $Q[h] \leftarrow \max(Q[h], 0.0)$
    $Q[hu] \leftarrow 0.0$
    $Q[hu+1] \leftarrow 0.0$

5.4.2. A-Posteriori Sub-Cell Limiting Execution Logic in ExaHyPE 2

The a-posteriori sub-cell limiting in ExaHyPE 2 is orchestrated by the supervisor solver, which dynamically switches between solvers based on local solution quality and admissibility criteria. This mechanism is centered around a finite set of internal states that reflect the solver configuration in each cell at a given time.

The supervisor solver operates under four primary states:

- `RegularSolver`,
- `LimiterStatusSpreadingToNeighbors`,
- `LimiterStatusSpreadingToSecondNeighbors`,
- `LimiterSolver`.

Both the regular and limiting solvers may enter a special `Suspended` state. When a solver is suspended, it is effectively excluded from participating in the grid sweep for the current time-step. This selective activation is coordinated by the supervisor solver and enables the correct coupling the of the solver results.

The `isLastGridSweep()` method, which normally indicated the completion of grid traversal for a given solver, is modified accordingly: if a solver is in the `Suspended` state, this method returns `false`. This adjustment is particularly important for the FV solver, as it otherwise always reports `true` by default. The correct behavior of `isLastGridSweep()` is essential to prevent premature updates or inconsistencies in the execution flow.

Supervisor Solver State: `RegularSolver`

In the `RegularSolver` state, the supervisor solver executes two grid sweeps. During the first sweep, the ADER-DG (regular) solver operates in the `InitialPrediction` mode, while the FV (limiting) solver is suspended and therefore does not participate in the computations.

The sweep begins with the `SaveNewCellData` action executed by the supervisor solver. This action serves two purposes. First, it synchronizes the time-step size between the regular and limiting solvers. If no troubled cells were detected in the previous time-step, the time-step size from ADER-DG solver is propagated to the FV solver. Conversely, if limiting is active, it is assumed that the FV solver holds valid state information for troubled cells and their first-level neighbors, and its time-step size is adopted for the next time layer calculation. This ensures numerical stability in the presence of local instabilities. Second,

`SaveNewCellData` also computes the local maxima and minima of the solution variables for each cell. These extrema are used later to evaluate the DMP criteria (4.2) during the limiter status check phase. After `SaveNewCellData`, the `Prediction` action is performed. It has been considered in details before for ADER-DG solver execution logic.

During the second grid sweep in this state, the ADER-DG solver performs the `Correction` step under the `TimeStep` state. Since this is the final operation of the ADER-DG solver within the current time-step, the ADER-DG time-step size is recomputed for the next time-step. Following the `Correction`, the supervisor solver executes the `VerifyTroubleness` action. This action determines whether any cells violate the admissibility criteria (4.1) and (4.2). The transition to the next state depends on the outcome of this verification:

- If no troubled cells are found, the supervisor solver concludes the time-step and re-enters the `RegularSolver` state for the next time-step.
- If troubled cells are detected, the supervisor switches to the `LimiterStatusSpreading ToNeighbors` state, indicating that the time-step is not yet complete and further processing by the limiting solver is required.

Supervisor Solver State: `LimiterStatusSpreadingToNeighbors` and `LimiterStatusSpreadingToSecondNeighbors`

The states `LimiterStatusSpreadingToNeighbors` and `LimiterStatusSpreadingToSecond Neighbors` are transitional phases during which both the regular and limiting solvers are suspended. As a result, only the supervisor solver is active in these stages, coordinating the propagation of troubled cell information and preparing the FV layer for local recomputation.

In the `LimiterStatusSpreadingToNeighbors` state, the supervisor executes the `Spread LimiterStatus` action. This routine scans all non-troubled cells and checks whether they are immediate (first-level) neighbors of troubled cells. If so, these cells are marked for further consideration and processing by the limiter.

On the neighbor status has been determined, the solver transitions into the `Limiter StatusSpreadingToSecondNeighbors` state. Here, the `CopyAndConvertPatch` action is triggered. It traverses all cell interfaces to identify and mark second-level neighbors - those adjacent to first-level neighbors of troubled cells. After marking, this same routine projects the solution data of all troubled, first-level, and second-level neighbor cells from the ADER-DG representation to the FV representation using the pregenerated operator A (4.10). Subsequently, the `adjustSolution` operation of the FV solver (see Algorithm 9) is applied.

Supervisor Solver State: `LimiterSolver`

In the `LimiterSolver` state, the regular solver is suspended, while the limiting solver transitions its active `TimeStep` phase. During this stage, the FVM method is used to recompute the solution in all cells previously marked as troubled, as well as their immediate neighbors. The core action in this state is `UpdateState`, which performs the numerical update using the FV scheme. This step ensures robust and stable computation in regions where the high-order ADER-DG method was deemed unreliable.

Once the FV update is completed, the supervisor invokes the `CopyAndConverPatch` operation again. This time, it traverses all previously troubled ADER-DG cells and their

first-level neighbors to project the updated solution back from the FV layer to the ADER-DG representation using the operator C (4.14). The recovered solution serves as the input for the next time-step.

Additionally, all ADER-DG cells are marked as troubled, while the number of genuinely troubled cells is reset to zero. This ensures that, during the following `RegularSolver` phase, the `SaveNewCellData` step correctly detects that limiting was applied in the previous time step, and therefore uses the time-step size from the finite volume (FV) solver.

6. Evaluation

To evaluate the accuracy, stability, and physical relevance of the proposed numerical framework, two complementary case studies are examined: a synthetic granular flow on an idealized 35° inclined slope and an avalanche simulation on Norway's Tafjord fjord topography. These scenarios serve distinct yet interconnected purposes - model verification under controlled conditions and validation in a complex, field-representative environment.

The synthetic test case is specifically designed to isolate and analyze the influence of key rheological parameters, including the Coulomb basal friction angle θ (see (2.14)) and the turbulent drag coefficient ξ, as defined in (2.15) and (2.17), respectively. By systematically varying these parameters, the physical consistency of the model is assessed, and characteristic flow behaviors are explored. Additionally, grid convergence studies are performed by comparing solutions across multiple mesh resolutions (coarse, standard, fine) and ADER-DG orders (4th, 6th, and 8th) to distinguish genuine physical effects from numerical artifacts.

The Tafjord case represents a real-world application of the complete model, incorporating realistic topography, frictional effects, and drag mechanisms derived from digital elevation data and field-based assumptions. This scenario evaluates the predictive capabilities of the solver in replicating large-scale avalanche motion under natural conditions. It also includes mesh and order sensitivity studies to ensure solution robustness in operational contexts.

Together, these cases enable a comprehensive evaluation of the numerical approach. The results presented in the following sections include both qualitative flow field visualizations and quantitative metrics, aimed at validating model performance, identifying resolution-dependent behavior, and establishing parameter settings appropriate for hazard analysis across synthetic and practical domains.

6.1. Granular Flow Dynamics on a 35° Inclined Plane

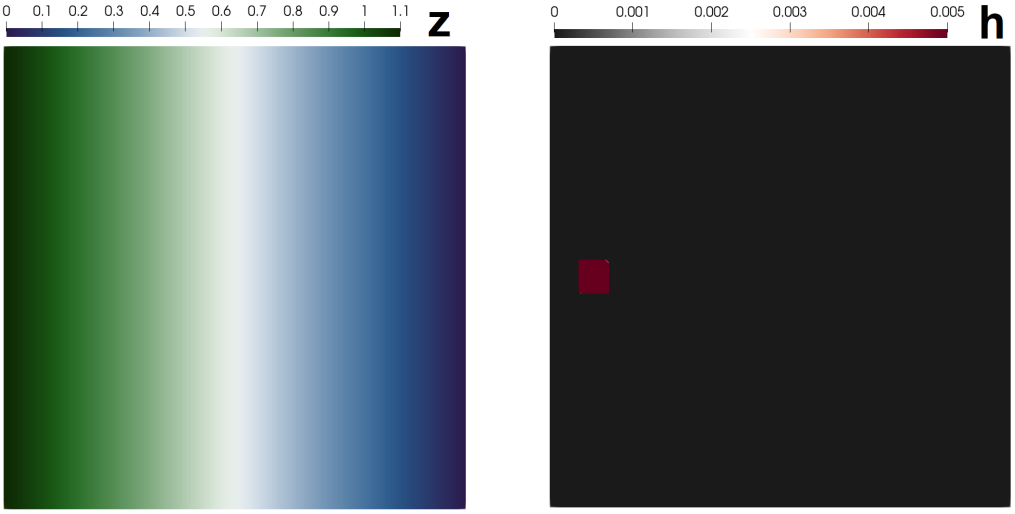
A synthetic test case is designed to evaluate the numerical framework's accuracy and stability under controlled conditions. The scenario simulates the gravitational-driven motion of a dry granular column released on a smooth, inclined plane with a constant slope angle of $\zeta = 35°$. This configuration enables systematic validation of flow dynamics and numerical robustness.

The computation domain consists of a $1.58\,\mathrm{m} \times 1.58\,\mathrm{m}$ square region discretized using structured quadrilateral elements. Topographic elevation is analytically prescribed as:

$$z(x, y) = (L - x) \cdot \tan(\zeta), \tag{6.1}$$

where $L = 1.58\,\mathrm{m}$ represents the domain length, and x denotes the horizontal coordinate, as depicted in Figure 6.1a. This formulation generates a planar slope oriented along the x-axis. The granular material is initialized as a square column with dimensions $0.1\,\mathrm{m} \times 0.1\,\mathrm{m}$, centered at $(0.15\,\mathrm{m}, 0.79\,\mathrm{m})$ near the slope crest. The initial material thickness field h

is uniformly set to 0.1 m with the column and zero elsewhere, as shown in Figure 6.1b. All velocity components are initialized to zero to simulate a stationary release. Although simulations are conducted in two dimensions, results are visualized using a three-dimensional perspective as well to emphasize topographic gradients and flow morphology. The initial configuration, including elevation and depth fields, is illustrated in Figure 6.2.



(a) Slope elevation $z(x, y)$. (b) Initial material thickness distribution h.

Figure 6.1.: Initial configuration for the 35° inclined slope case.

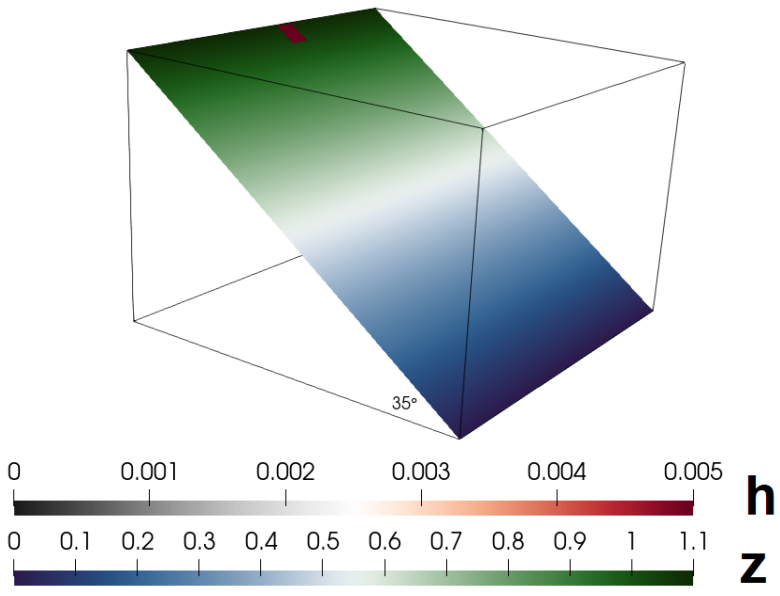


Figure 6.2.: 3D perspective of the 35° inclined slope case. The granular column (red) is positioned at the slope crest, with elevation gradients highlighted by shading.

To mitigate numerical instabilities at wet-dry interfaces, the dry-state threshold h_{dry},

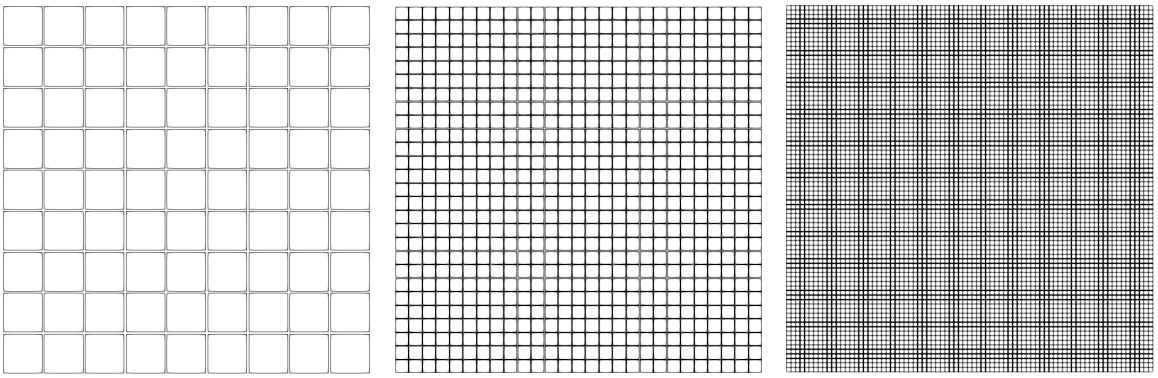
introduced in (4.16), and the solution-adjustment threshold $h_{\text{threshold}}$ from the Algorithm 9 are both assigned values $10^{-3}\,\text{m}$. Reflective conditions are applied across all domain boundaries, enforcing momentum inversion relative to boundary normals and ensuring no-flux behavior.

6.1.1. Grid Convergence and Solver Order Analysis

A systematic convergence study is carried out to evaluate the sensitivity of the simulated dynamics to discretization parameters, aiming to identify a threshold beyond which numerical artifacts become negligible. The analysis focuses on two key aspects: mesh resolution and the order of numerical approximation. The objective is to verify consistency across solver orders and resolutions, thereby isolating rheology-driven effects from discretization-induced biases. Throughout this section, simulations are conducted with fixed friction parameters: $\xi = 200\,\text{m/s}^2$ and $\theta = 25°$.

Grid Independence Study

To investigate the influence of spatial resolution on the simulation outcome, a grid convergence study is performed using three structured meshes of increasing refinement. The fixed 6th-order numerical scheme is used in all cases, while only the cell size is varied. The coarsest mesh contains 2916 elements, the standard mesh of 26244 elements, and the finest mesh 236196 elements, as depicted in Figure 6.3.



(a) Coarse mesh (b) Standard mesh (c) Fine mesh

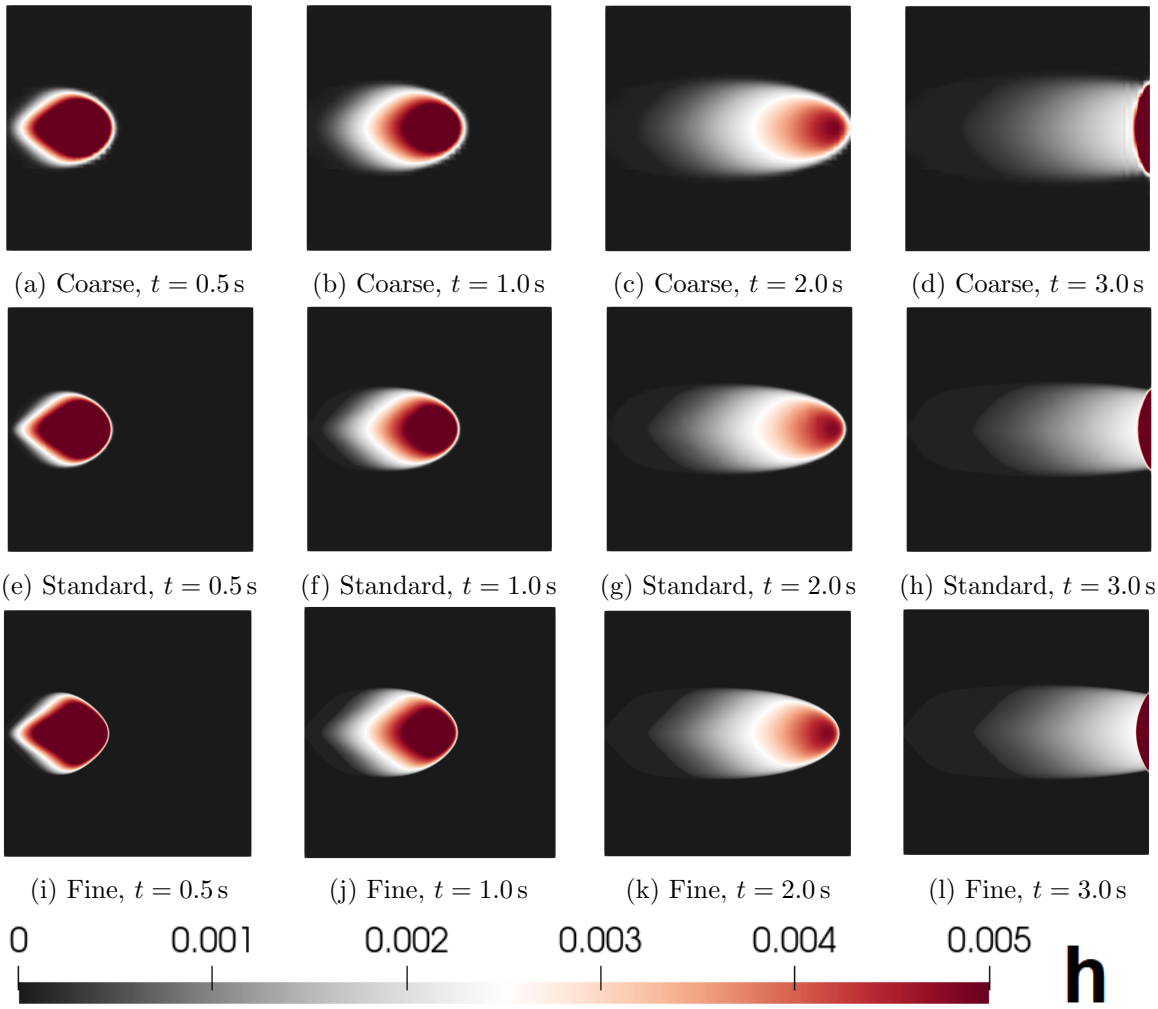
Figure 6.3.: Coarse (2916 elements), Standard (26244 elements), and Fine (236196 elements) grids for the 35° inclined slope case.

To ensure numerical stability and enforce the Discrete Maximum Principle (4.2), the parameters b and k in the DMP criterion (4.3) are adapted to the grid resolution. Specifically, the following values are used:

- **Coarse:** $b = 2 \cdot 10^{-3}$, $k = 0.4$;
- **Standard:** $b = 1 \cdot 10^{-3}$, $k = 0.2$;
- **Fine:** $b = 2 \cdot 1^{-4}$, $k = 0.02$.

The CFL relaxation parameter (3.69) is set to 0.45 for coarse and standard meshes, and reduced to 0.2 for the fine mesh to maintain stability under refined spatial discretization.

Figure 6.4 shows the temporal evolution of the material thickness h at $t = \{0.5\,\text{s}, 1.0\,\text{s}, 2.0\,\text{s}, 3.0\,\text{s}\}$. As the resolution increases, numerical diffusion is reduced and small-scale features become more coherent. The coarse mesh displays visible artifacts. In contrast, the standard and fine meshes produce smoother and more physically plausible flow fields, with negligible differences between them.



(a) Coarse, $t = 0.5\,\text{s}$ (b) Coarse, $t = 1.0\,\text{s}$ (c) Coarse, $t = 2.0\,\text{s}$ (d) Coarse, $t = 3.0\,\text{s}$

(e) Standard, $t = 0.5\,\text{s}$ (f) Standard, $t = 1.0\,\text{s}$ (g) Standard, $t = 2.0\,\text{s}$ (h) Standard, $t = 3.0\,\text{s}$

(i) Fine, $t = 0.5\,\text{s}$ (j) Fine, $t = 1.0\,\text{s}$ (k) Fine, $t = 2.0\,\text{s}$ (l) Fine, $t = 3.0\,\text{s}$

Figure 6.4.: h for three mesh resolutions. As resolution increases, numerical artifacts are reduced and the solution becomes smoother.

In addition to the visual comparison, the flow depth is monitored at a fixed point near the slope toe, located at $(1.57\,\text{m}, 0.79\,\text{m})$. Figure 6.5 presents the temporal evolution of the h at this location. On the coarse mesh, the material accumulates more slowly and to a lesser extent. The medium and fine meshes, however, yield nearly identical curves, especially after $t \approx 2.2\,\text{s}$, suggesting that the grid convergence has been achieved. Based on these results, the standard-resolution mesh is considered sufficient for capturing the key flow features with

reasonable accuracy, while offering significant computational savings compared to the fine mesh.

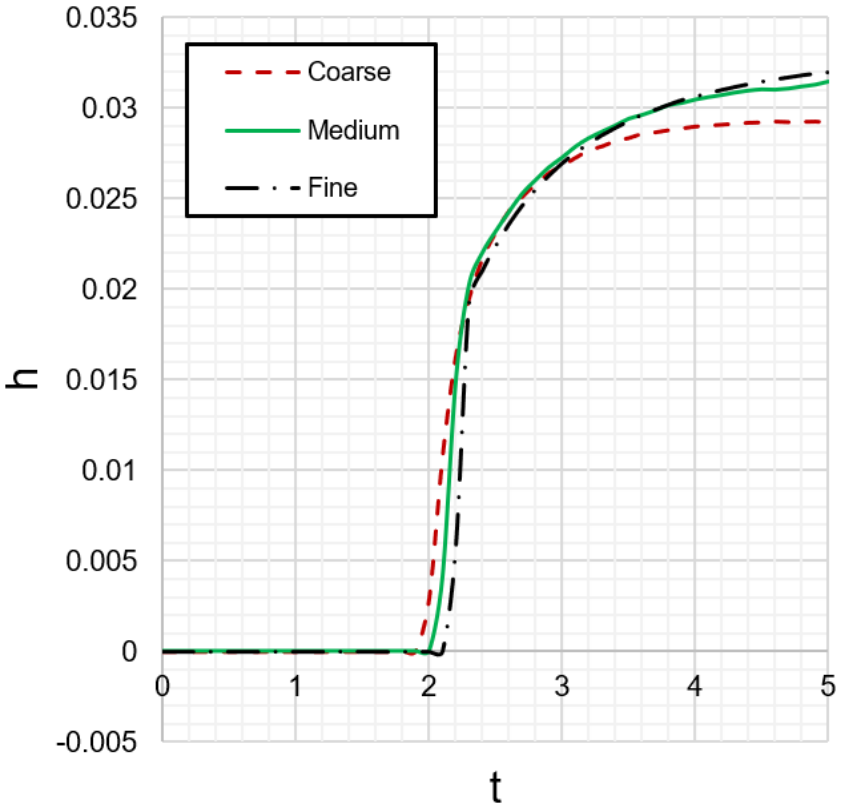


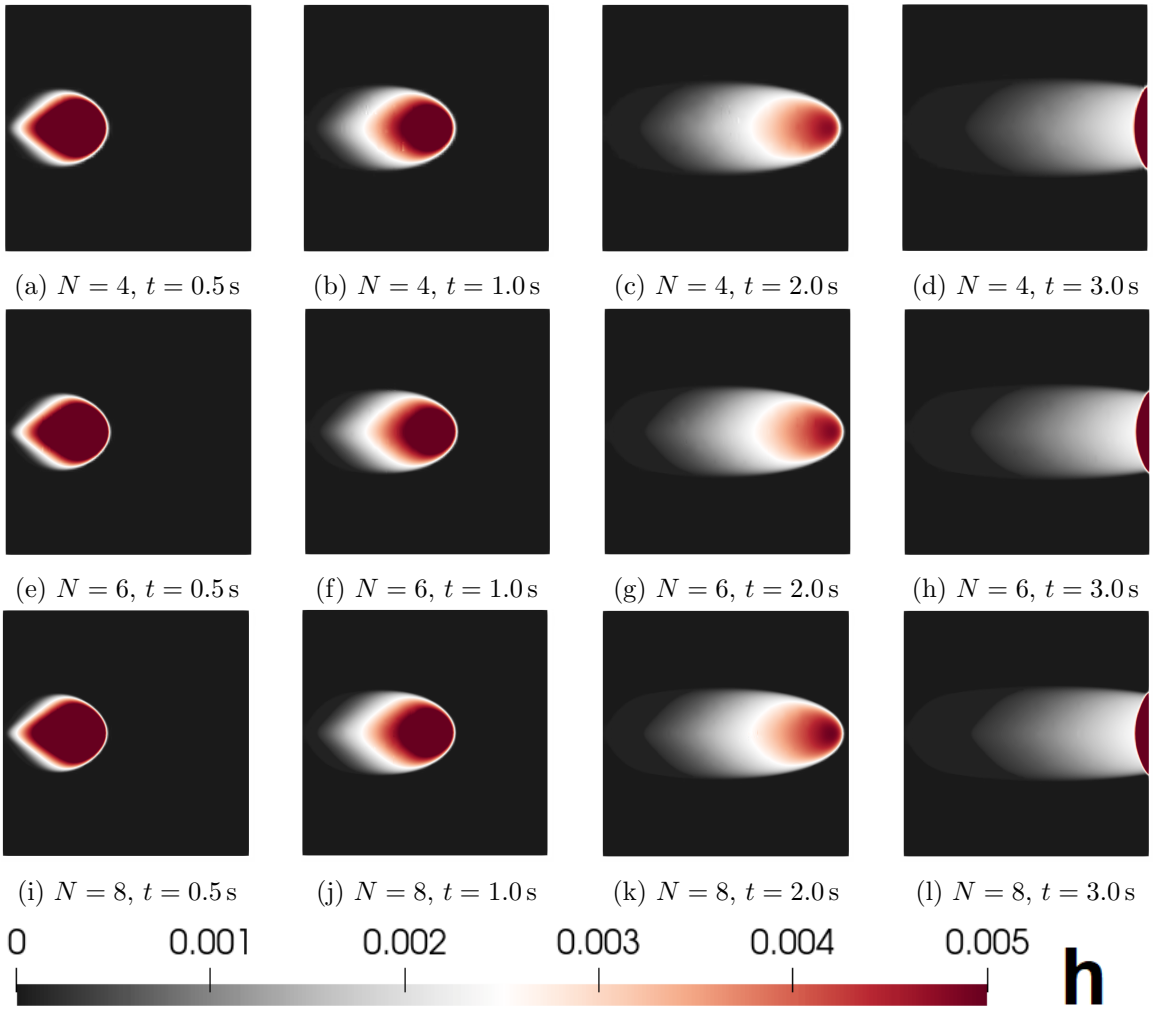
Figure 6.5.: Temporal evolution of h at the monitoring point $(1.57\,\mathrm{m}, 0.79\,\mathrm{m})$ for different mesh resolutions. The standard and fine meshes exhibit close agreement.

ADER-DG Order Sensitivity

To scrutinize the influence of the numerical approximation order on solution quality, simulations are performed using 4th-, 6th-, and 8th-order ADER-DG schemes on the standard mesh. To maintain stability, the DMP (4.3) parameters b and k in criterion (4.2) are adjusted according to the order. Specifically, for the 4th- and 6th-orders $b = 1 \cdot 10^{-3}$ and $k = 0.2$ are used, while the 8th-order employed $b = 5 \cdot 10^{-4}$ and $k = 0.1$. A CFL number (3.69) of 0.45 is applied across all considered orders.

Figure 6.6 illustrates the temporal evolution of the material thickness h at $t = \{0.5\,\mathrm{s}$, $1.0\,\mathrm{s}$, $2.0\,\mathrm{s}$, $3.0\,\mathrm{s}\}$, for each ADER-DG order. The flow initiates from a localized squared column, as depicted in Figure 6.1b. Across all methods, the overall propagation direction and deposition trends remain qualitatively similar. However, visual inspection reveals subtle but important differences. At $t = 0.5\,\mathrm{s}$ the granular front is circular and relatively compact. As time progresses to $t = 1.0\,\mathrm{s}$ and $t = 2.0\,\mathrm{s}$, material spreads rapidly along the slope, and the influence of numerical diffusion becomes more pronounced. The 4th-order method shows slightly smeared fronts compared to the higher-order counterparts. The 6th-order scheme captures sharper transitions and preserves the shape of the leading edge more effectively.

The 8th-order method performs similarly, through its advantage is less significant compared to the 6th-order at the selected resolution. At the final snapshot $t = 3.0\,\mathrm{s}$, all schemes show the granular flow settling, but the higher-order methods retain a more compact shape.



(a) $N = 4$, $t = 0.5\,\mathrm{s}$ (b) $N = 4$, $t = 1.0\,\mathrm{s}$ (c) $N = 4$, $t = 2.0\,\mathrm{s}$ (d) $N = 4$, $t = 3.0\,\mathrm{s}$

(e) $N = 6$, $t = 0.5\,\mathrm{s}$ (f) $N = 6$, $t = 1.0\,\mathrm{s}$ (g) $N = 6$, $t = 2.0\,\mathrm{s}$ (h) $N = 6$, $t = 3.0\,\mathrm{s}$

(i) $N = 8$, $t = 0.5\,\mathrm{s}$ (j) $N = 8$, $t = 1.0\,\mathrm{s}$ (k) $N = 8$, $t = 2.0\,\mathrm{s}$ (l) $N = 8$, $t = 3.0\,\mathrm{s}$

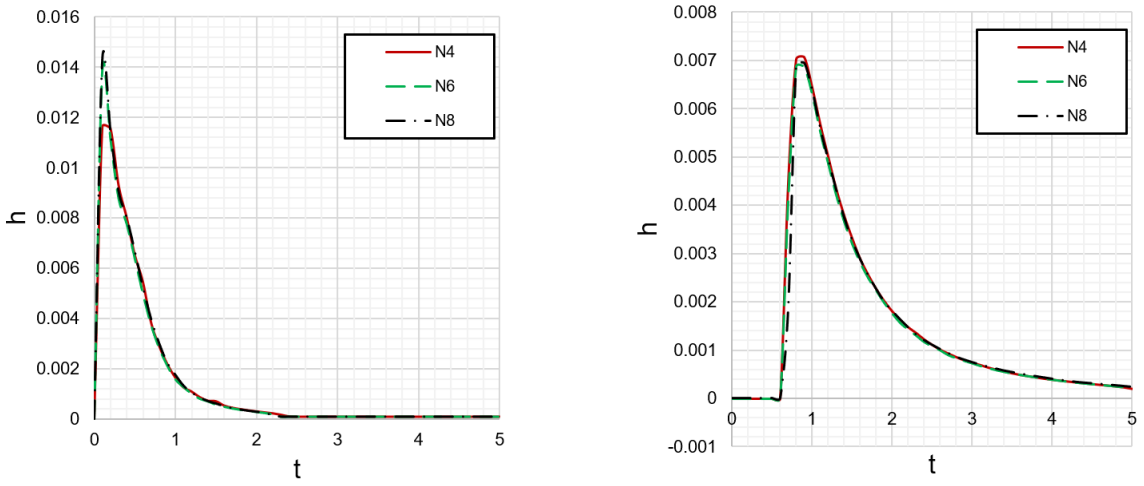
Figure 6.6.: The material thickness distribution h for 4th, 6th, and 8th ADER-DG orders for the 35° inclined slope.

To provide a quantitative comparison, the flow depth is monitored at two representative locations along the slope: $(0.3\,\mathrm{m}, 0.79\,\mathrm{m})$ near the release zone, and $(0.79\,\mathrm{m}, 0.79\,\mathrm{m})$ in the mid-slope region. These positions are selected to capture different phases of the avalanche - the initial release and the active movement. The corresponding temporal profiles of the material thickness h are shown in Figure 6.7a and Figure 6.7b.

At the first monitoring point Figure 6.7a, which lies close to the initial release region, the material thickness exhibits a steep initial rise followed by rapid decay, reflecting the local depletion of mass as the flow begins to spread downslope. All three schemes (4th, 6th, and 8th order) produce nearly identical curves, although higher-order schemes exhibit a slightly more aligned initial peak decay due to reduced numerical dissipation. At the second

point Figure 6.7b, located farther down the slope, the profiles capture the passage of the main flow front. A distinct peak forms around $t = 0.9\,\mathrm{s}$, corresponding to the arrival of the avalanche mass. The 8th-order scheme captures the sharpness of the peak slightly better, while the 4th-order scheme produces a slightly smoothed version. Nevertheless, the temporal evolution remains consistent across all orders, and long-term differences are negligible.

These results confirm the reliability of the ADER-DG solver across high polynomial orders, with the 6th-order scheme offering an effective trade-off between solution accuracy and computational cost. It maintains sufficient resolution of steep gradients without the higher memory and time demands associated with 8th-order discretizations.



(a) Monitoring point at $(0.30\,\mathrm{m}, 0.79\,\mathrm{m})$

(b) Monitoring point at $(0.79\,\mathrm{m}, 0.79\,\mathrm{m})$

Figure 6.7.: Temporal evolution of material thickness h at two monitoring locations for different ADER-DG orders. All schemes yield nearly identical results.

Landslide Evolution on the Standard Configuration

To illustrate the temporal evolution of the granular flow under fixed numerical settings, simulations are performed using the 6th-order ADER-DG scheme on the standard mesh. The configuration corresponds to previously tested parameters with CFL=0.45 and DMP coefficients $b = 1 \cdot 10^{-3}$ and $k = 0.2$.

Figure 6.8 shows snapshots of the material flow thickness h at selected time $t = 0.5\,\mathrm{s}$, $t = 1.0\,\mathrm{s}$, $t = 2.0\,\mathrm{s}$, and $t = 3.0\,\mathrm{s}$. The material initially accelerates and rapidly deforms, producing a characteristic downslope spreading pattern. At later times, the front continues to propagate while the rear portion thins and decelerates.

The 2D plots in the first row of Figure 6.8 show the material spreading outward and downslope, forming an elongated front. The central region of great thickness gradually flattens out as the mass redistributes. In the 3D images in the second row, you can see how the granular layer follows the topography, with steeper gradients enhancing the forward motion. It can also be seen how the thickness peak shifts forward in time, while the trailing edge slows down due to frictional dissipation.

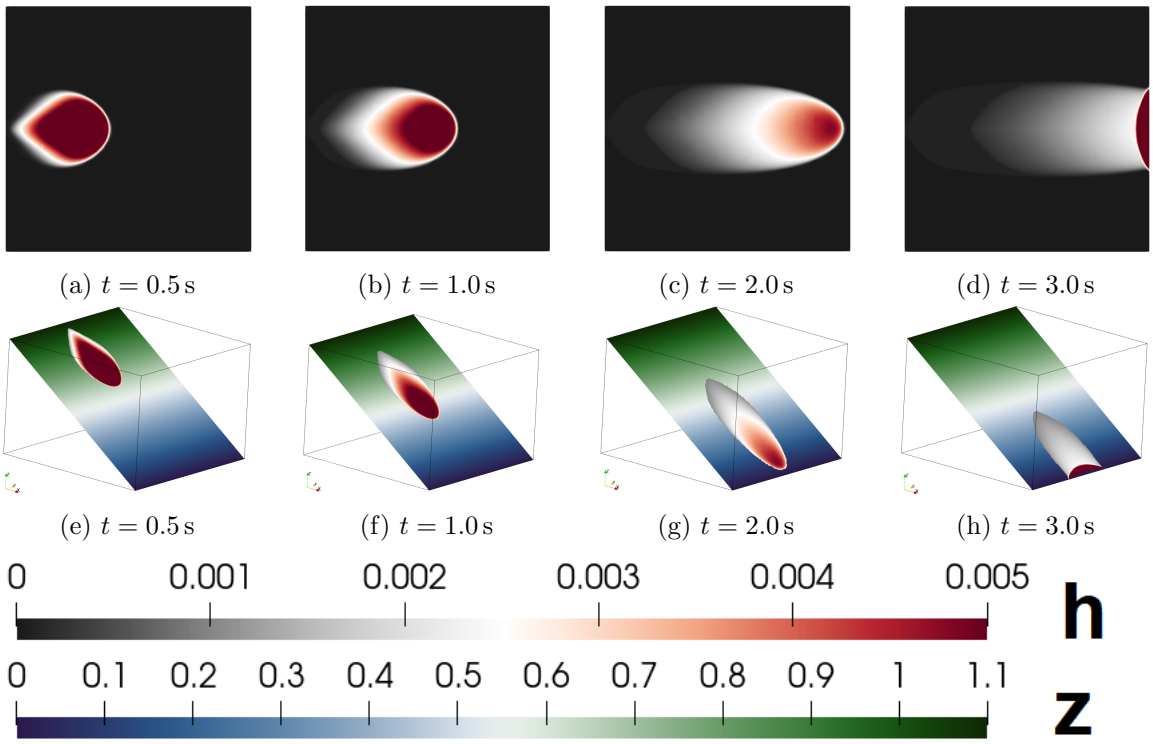


Figure 6.8.: Temporal evolution of the material thickness h using the 6th-order scheme on the standard mesh for the 35° inclined slope case.

6.1.2. Influence of Frictional Parameters on Flow Behavior

To evaluate the physical consistency of the numerical model, the influence of key rheological parameters is investigated. In particular, the Coulomb friction angle θ (2.14) and the turbulent drag coefficient ξ (2.17) are varied independently to observe their impact on the flow propagation. These tests aim to verify that the model responds in physically interpretable ways to changes in both basal and velocity-dependent friction mechanisms.

All simulations are performed using the 6th-order ADER-DG scheme on the standard mesh configuration. The CFL relaxation parameter (3.69) is fixed at 0.45, and the DMP stabilization coefficients (4.3) are set to $b = 5 \cdot 10^{-4}$ and $k = 0.1$.

Variation of Coulomb Friction Angle θ

A parametric study evaluates the influence of Coulomb-type basal friction (2.12), governed by the angle $\theta \in \{0°, 12.5°, 25°, 50°\}$, on granular flow dynamics. The turbulent drag is deactivated to isolate frictional effects. The angle θ directly affects basal friction coefficient μ, which is defined as $\mu = \tan(\theta)$. Thus, with higher values, the increased friction is expected.

Figure 6.9 demonstrates the temporal evolution of material thickness h at four snapshots: $\{0.25\,\mathrm{s}, 0.5\,\mathrm{s}, 0.75\,\mathrm{s}, 1.0\,\mathrm{s}\}$. A clear trend is observed: larger friction angles result in reduced downslope propagation, delayed fronts, and limited material spreading. As friction increases, the granular column retains a more compact shape during the motion. In the extreme case of $\theta = 50°$, no significant motion is observed throughout the simulation, and the friction

was able to fully stop the material at the starting location. This outcome is consistent with theoretical expectations, as the basal friction angle exceeds the slope inclination $\zeta = 35°$, resulting in the driving gravitational force insufficient to overcome basal resistance.

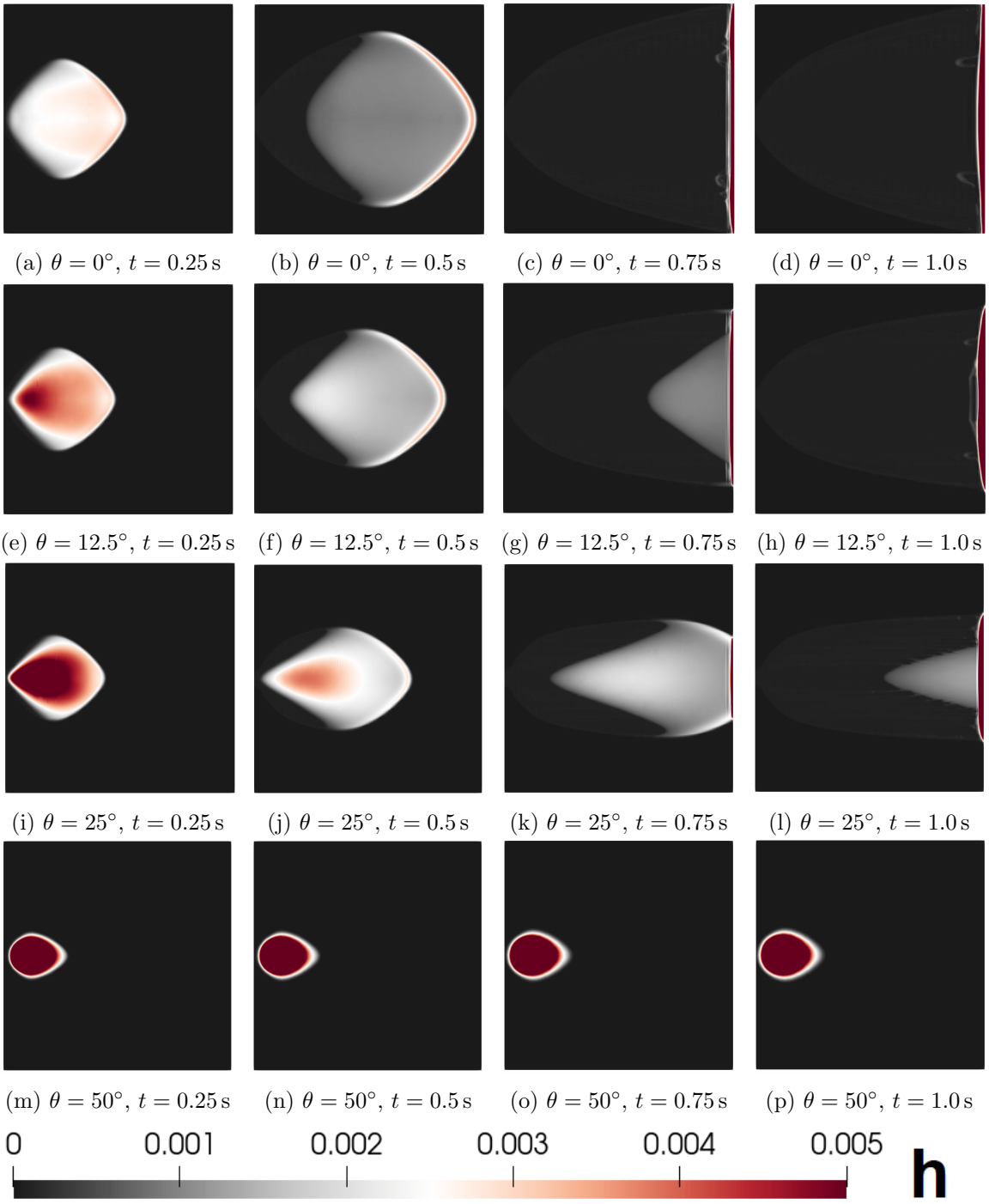


Figure 6.9.: Evolution of the material thickness for h for different Coulomb friction angles θ.

Quantitative comparison is performed at the downslope monitoring point $(1.57\,\mathrm{m}, 0.79\,\mathrm{m})$. The resulting profiles are shown in Figure 6.10. The arrival time of the from is inferred from the timing of the peak value. For $\theta = 0°$, the front reaches the point at $t \approx 0.8\,\mathrm{s}$, with a peak height $h \approx 0.042\,\mathrm{m}$. For $\theta = 12.5°$ the peak is delayed to $t \approx 0.9\,\mathrm{s}$, and the maximum height increases to $h \approx 0.054\,\mathrm{m}$. For $\theta = 25°$ the peak occurs at $t \approx 1.1\,\mathrm{s}$, with $h \approx 0.052\,\mathrm{m}$. These results confirm that moderate friction angles delay and reduce flow propagation, while high friction angles can entirely suppress motion. The observed behavior is consistent with the physical interpretation of Coulomb-type basal resistance.

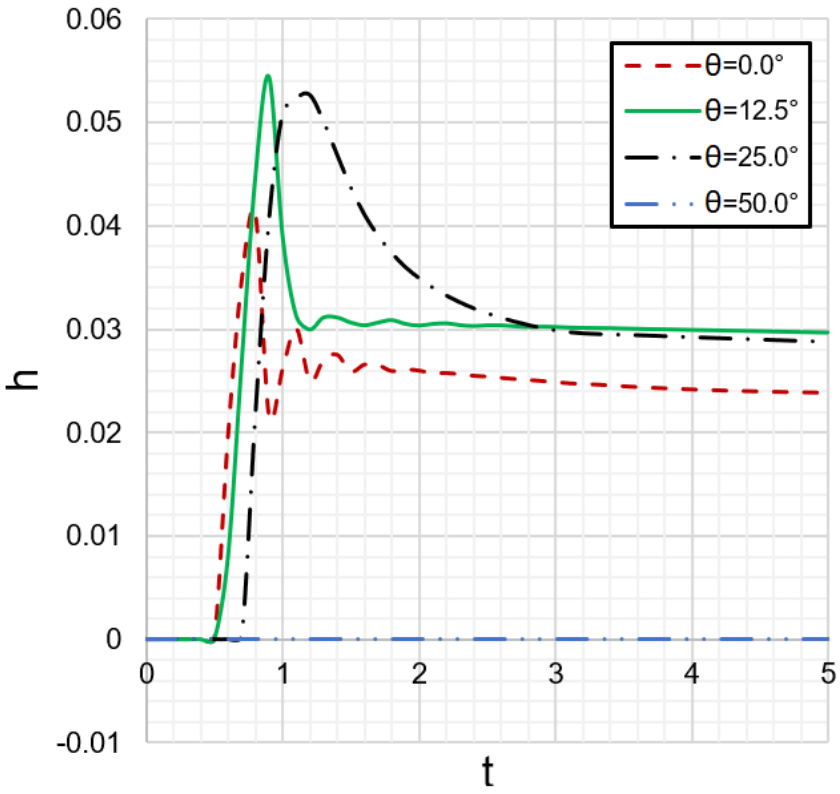


Figure 6.10.: Temporal evolution of the material thickness h at the monitoring point $(1.57\,\mathrm{m}, 0.79\,\mathrm{m})$ for different basal friction angles θ.

Variation of Turbulent Drag Coefficient ξ

The influence of the turbulent drag coefficient ξ on the landslide dynamics has also been studied. To isolate and evaluate the effect of turbulent drag on granular flow behavior, a series of simulations is performed with fixed Coulomb friction angle $\theta = 25°$, while varying the turbulent drag coefficient $\xi \in \{100\,\mathrm{m/s^2}, 200\,\mathrm{m/s^2}, 2000\,\mathrm{m/s^2}, \infty\}$. The case $\xi \to \infty$ effectively corresponds to the absence of turbulent drag.

The turbulent drag force (2.17) is inversely proportional to ξ, such that smaller values of ξ correspond to stronger velocity-dependent resistance. Therefore, increasing ξ reduces the influence of turbulent dissipation on the flow dynamics.

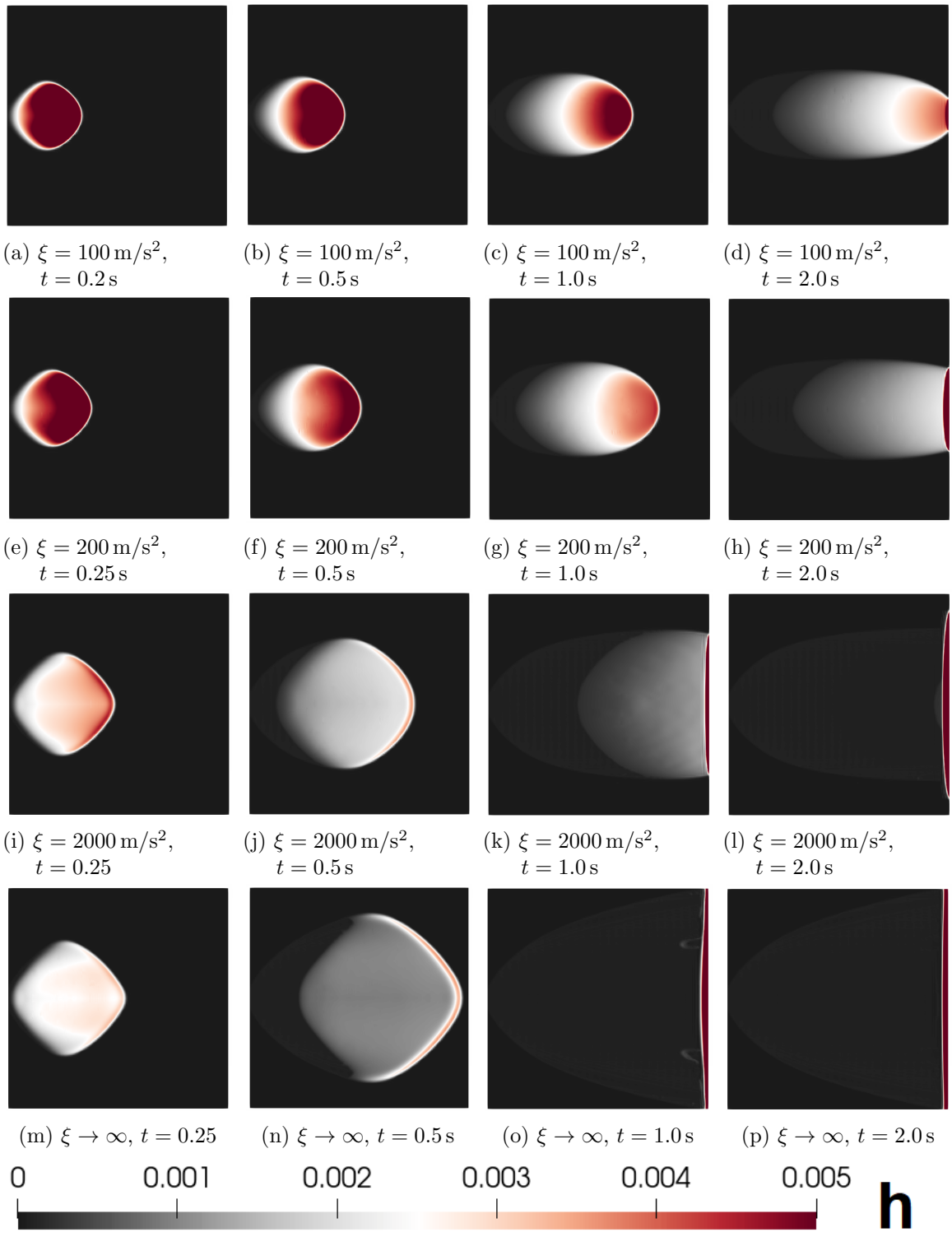


Figure 6.11.: Evolution of the material thickness h for various turbulent drag coefficients ξ.

Figure 6.11 illustrates the spatiotemporal evolution of granular flow thickness h under

varying turbulent drag coefficients ξ. Lower values of ξ, corresponding to stronger velocity-dependent resistance, significantly suppress downslope motion. This results in compact, slow-moving deposits with limited lateral spreading. As ξ increases, the influence of turbulent dissipation diminishes, allowing for faster front propagation and more elongated flow patterns. In the limiting case $\xi \to \infty$, where drag is absent, inertial forces dominate the dynamics, leading to rapid downslope acceleration and the formation of thin, widely dispersed deposits. These results underscore the critical role of ξ in modulating the interplay between inertial transport and dissipative effects, thereby shaping the extent and morphology of granular flows.

The Figure 6.12 displays the temporal evolution of h at the monitoring point $(1.57\,\mathrm{m}, 0.79\,\mathrm{m})$ for different values of the turbulent drag coefficient ξ. The timing and magnitude of the peak indicate when the material reaches the slope toe. For $\xi = 100\,\mathrm{m/s^2}$, the front arrives at approximately $t = 2\,\mathrm{s}$; for $\xi = 200\,\mathrm{m/s^2}$, the arrival occurs earlier at around $t = 1.6\,\mathrm{s}$; and for $\xi = 2000\,\mathrm{m/s^2}$, the peak is observed near $t = 1\,\mathrm{s}$. In the limiting case $\xi \to \infty$, corresponding to zero velocity drag, the front reaches the point as early as $t = 0.5\,\mathrm{s}$.

These results demonstrate the decelerating influence of turbulent drag: lower ξ values (i.e., stronger drag) lead to slower downslope movement and more localized accumulation. Conversely, as ξ increases, resistance weakens, and the material reaches the toe more quickly.

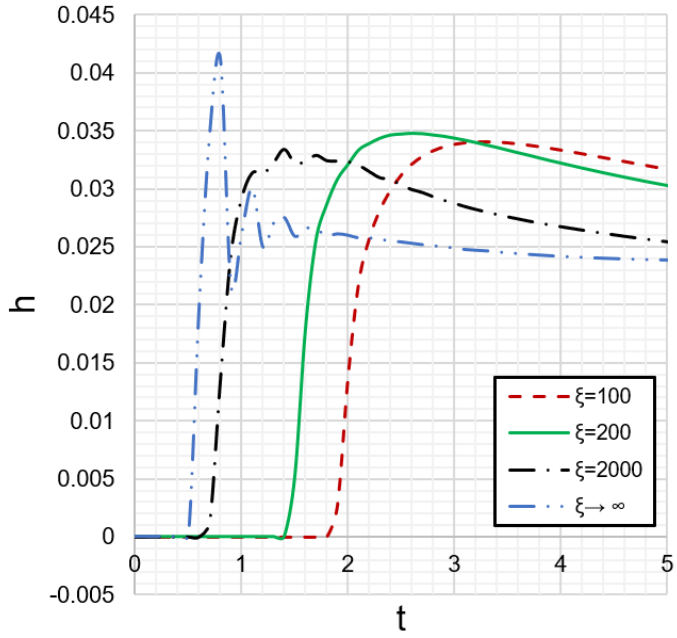
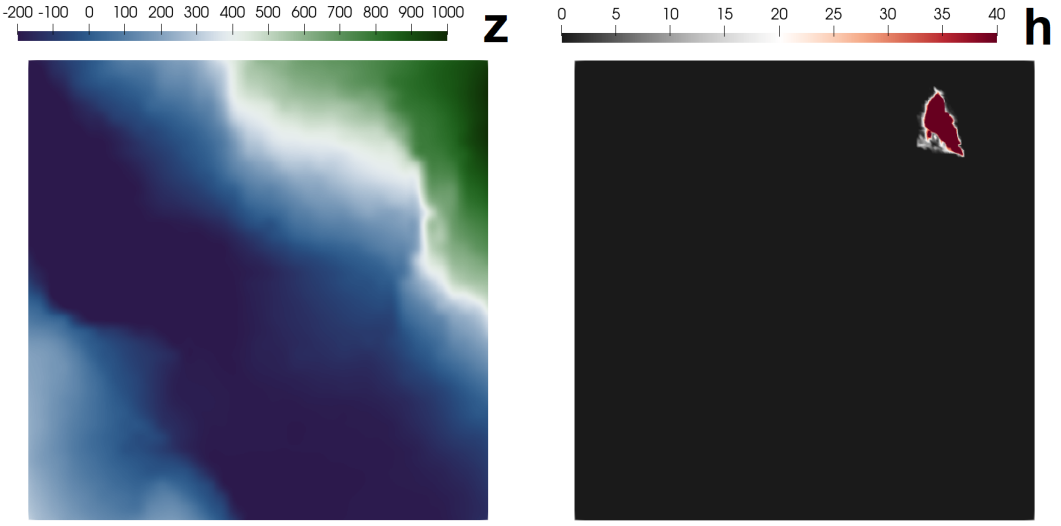


Figure 6.12.: Temporal evolution of the material thickness h at the monitoring point $(1.57\,\mathrm{m}, 0.79\,\mathrm{m})$ for various turbulent drag coefficients ξ.

6.2. Tafjord Topography-Based Simulation

To examine the applicability of the numerical model to complex terrain, a test case based on the topography of the Tafjord region in Norway is considered. The area features steep

mountainous slopes and varied elevation, providing a suitable environment to evaluate the model's capability to handle realistic flow configurations.



(a) The Tafjord case elevation z.

(b) Initial distribution of material thickness h in 2D view.

Figure 6.13.: Initial topography and material configuration for the Tafjord case.

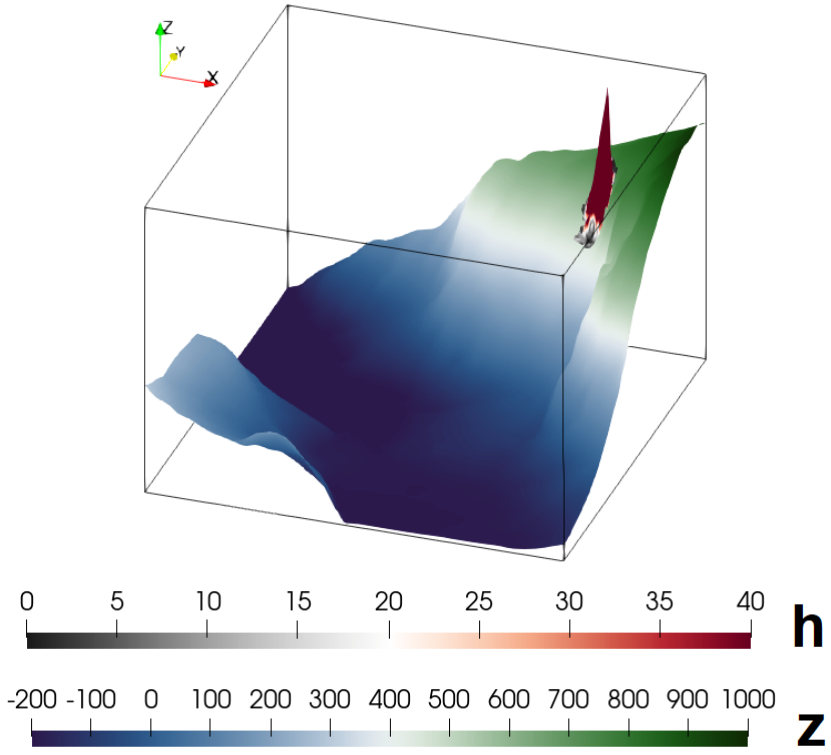


Figure 6.14.: 3D visualization of initial topography and release zone of the Tafjord case.

The computational domain covers a square region of size 1900×1900 measured in meters (as per UTM coordinate system), offset from the origin such that its lower-left corner is located at coordinates $(414895.5\,\mathrm{m}, 6904545.5\,\mathrm{m})$. The bed elevation z is initialized using high-resolution topographic data provided in NetCDF format. The initial material thickness h is prescribed from a separate NetCDF dataset, designed to represent a localized granular release zone. Both momentum components hu and hv are set to zero, corresponding to an initially static state. Figure 6.13 presents the initial conditions in both elevation and thickness fields. A complementary three-dimensional visualization is shown in Figure 6.14, offering a clearer impression of the topographic variation and the placement of the initial release area.

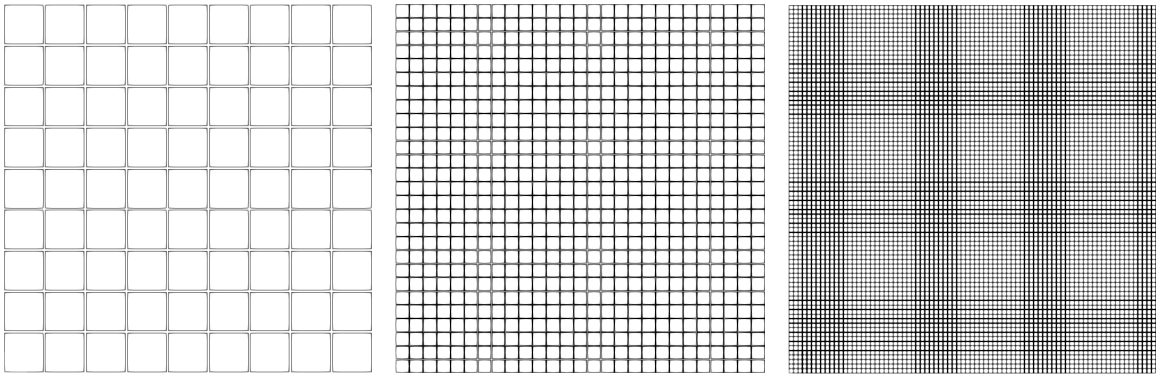
The dry-state treatment is controlled by setting $h_{\mathrm{dry}} = h_{\mathrm{threshold}} = 10^{-1}\,\mathrm{m}$ to prevent numerical instabilities in regions with near-vanishing material thickness h.

6.2.1. Grid Convergence and Solver Order Analysis

Prior to conducting the full simulation and analyzing the granular flow dynamics over Tafjord topography, a preliminary study is carried out to determine suitable discretization parameters. This investigation focuses on the influence of mesh resolution and numerical solver order, aiming to identify a configuration that balances computational cost with sufficient accuracy. By evaluating the convergence of the solution with respect to grid refinement and increasing ADER-DG order, the study ensures that the numerical results are stable, reliable, and largely independent of the underlying discretization. These optimal parameters are then adopted for the subsequent simulation phase to capture the flow behavior over the real topographic surface with minimal numerical artifacts. During all numerical experiments in this section parameters are set: $\xi = 200\,\mathrm{m/s^2}$ and $\theta = 25°$.

Grid Independence Study

To assess the sensitivity of the numerical model to spatial resolution, a grid convergence study is performed using three structured meshes of increasing refinement: coarse (2916 elements), standard (26244 elements), and fine (236196 elements), as depicted in Figure 6.15.

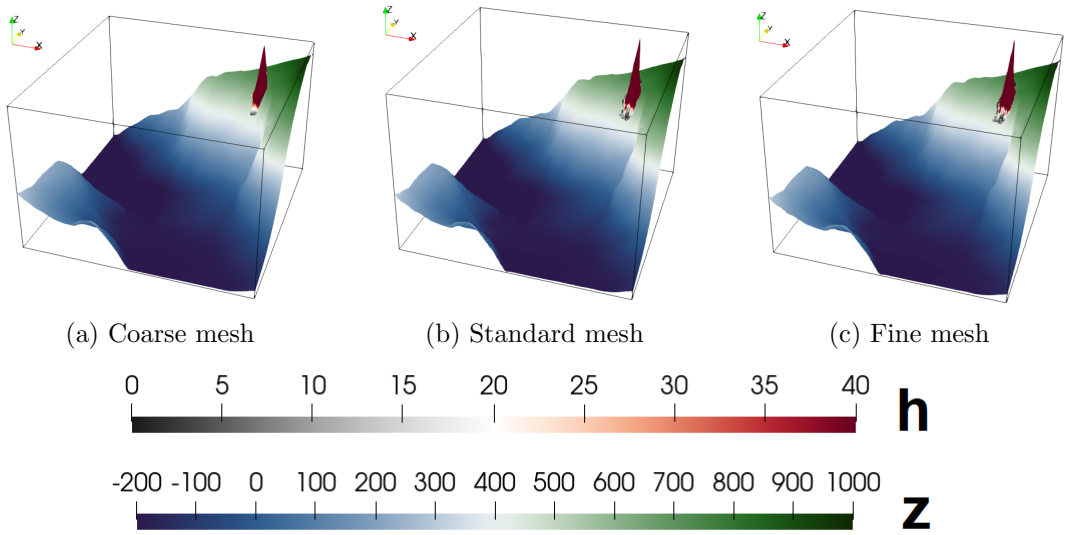


(a) Coarse mesh (b) Standard mesh (c) Fine mesh

Figure 6.15.: Coarse (2916 elements), Standard (26244 elements), and Fine (236196 elements) grids for the Tafjord case.

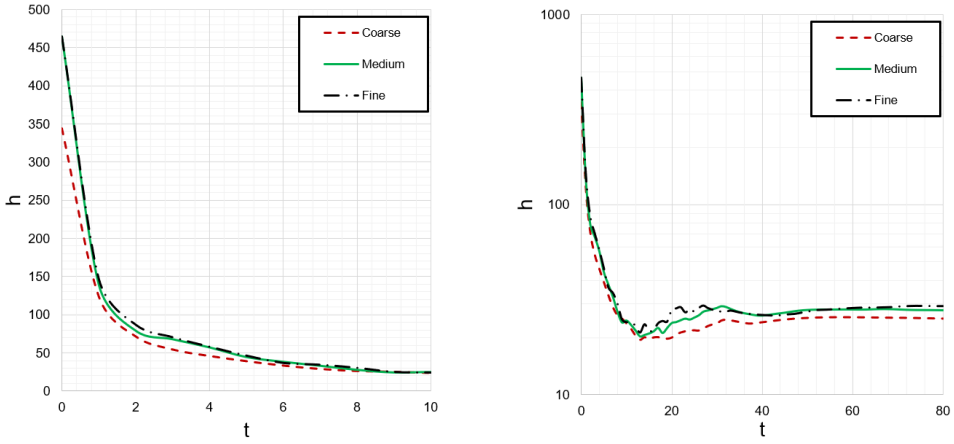
To enforce the DMP (4.2) and suppress unphysical oscillations, the stabilization parameters k and b (4.3) are scaled inversely with mesh resolution. Specifically, $(k = 0.02, b = 2.0)$, $(k = 0.01, b = 1.0)$, and $(k = 0.002, b = 0.2)$ are employed for coarse, standard, and fine meshes, respectively. For the fine grid, CFL=0.2 is used, whereas for other cases, CFL=0.45. The ADER-DG order is 6 for all of these cases.

The initial configuration for each resolution is shown in Figure 6.16 in 3D perspective. The coarse mesh lacks sufficient granularity to represent the peak of the granular deposit, leading to artificial smoothing of the initial condition. In contrast, the standard and fine meshes preserve the initial distribution of h identically.



(a) Coarse mesh (b) Standard mesh (c) Fine mesh

Figure 6.16.: Initial configuration of the material thickness h and topography elevation z for different grid resolutions for the Tafjord case.

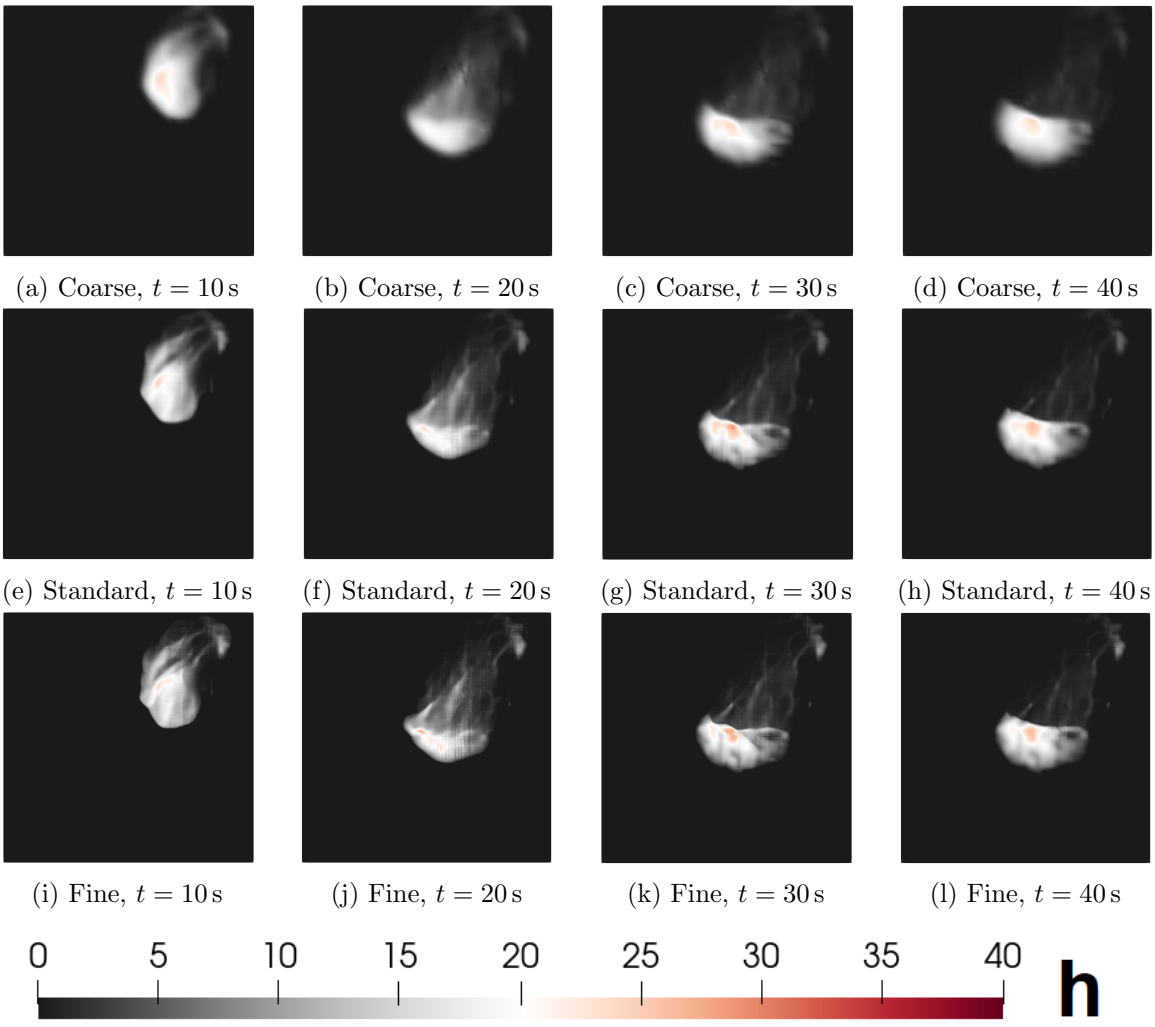


(a) Linear scale (b) Logarithmic scale

Figure 6.17.: Temporal evolution of the material thickness h at the point $(416423\,\mathrm{m}, 6906230\,\mathrm{m})$ for three mesh resolutions.

The analysis focuses on a monitoring point at (416423 m, 6906230 m), corresponding to the initial granular column's peak. Temporal evolution of material thickness h at this location is shown in Figure 6.17 using linear and logarithmic scales. All simulations exhibit rapid thinning of the granular layer due to gravitational acceleration and lateral spreading. Notable discrepancies arise during the initial collapse phase: the coarse mesh underestimates peak height, whereas standard and fine meshes produce nearly identical profiles, with differences in peak magnitude below 2%, confirming convergence.

Figure 6.18 compares flow progression at $t = 10\,\text{s}$, $20\,\text{s}$, $30\,\text{s}$, $40\,\text{s}$ across resolutions. Coarse mesh results exhibit front smearing, while standard and fine resolutions capture sharp flow fronts and secondary deposition lobes.



(a) Coarse, $t = 10\,\text{s}$ (b) Coarse, $t = 20\,\text{s}$ (c) Coarse, $t = 30\,\text{s}$ (d) Coarse, $t = 40\,\text{s}$

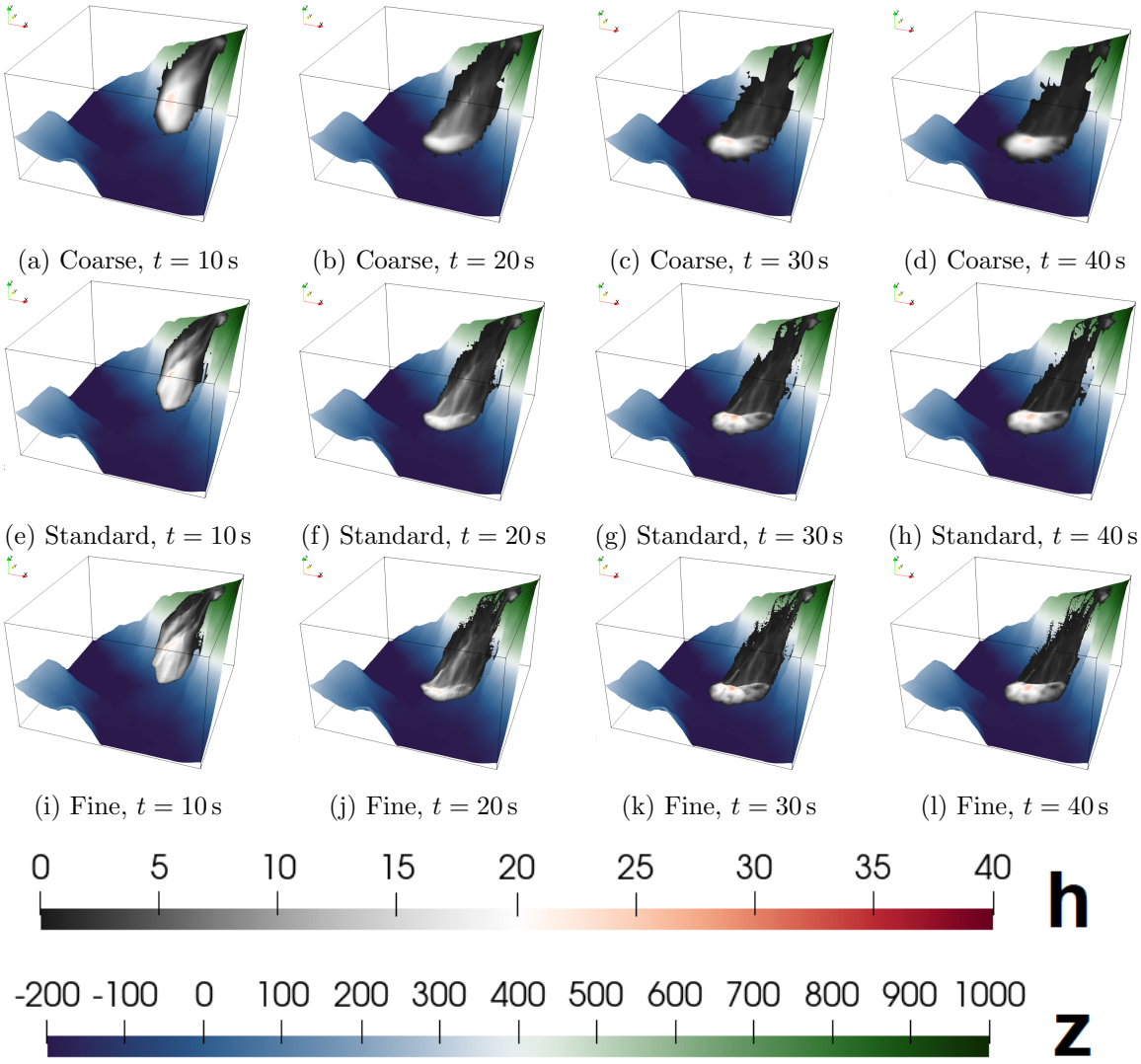
(e) Standard, $t = 10\,\text{s}$ (f) Standard, $t = 20\,\text{s}$ (g) Standard, $t = 30\,\text{s}$ (h) Standard, $t = 40\,\text{s}$

(i) Fine, $t = 10\,\text{s}$ (j) Fine, $t = 20\,\text{s}$ (k) Fine, $t = 30\,\text{s}$ (l) Fine, $t = 40\,\text{s}$

Figure 6.18.: The material thickness h for different mesh resolutions at multiple time-steps for the Tafjord case.

Complementary three-dimensional views of the same data are provided in Figure 6.19, emphasizing the interaction between the granular flow and the underlying topography. On

coarse grids, the terrain features are insufficiently resolved, leading to smoothed and less accurate flow fields, whereas standard and fine resolutions preserve more elevation features.



(a) Coarse, $t = 10\,\mathrm{s}$ (b) Coarse, $t = 20\,\mathrm{s}$ (c) Coarse, $t = 30\,\mathrm{s}$ (d) Coarse, $t = 40\,\mathrm{s}$

(e) Standard, $t = 10\,\mathrm{s}$ (f) Standard, $t = 20\,\mathrm{s}$ (g) Standard, $t = 30\,\mathrm{s}$ (h) Standard, $t = 40\,\mathrm{s}$

(i) Fine, $t = 10\,\mathrm{s}$ (j) Fine, $t = 20\,\mathrm{s}$ (k) Fine, $t = 30\,\mathrm{s}$ (l) Fine, $t = 40\,\mathrm{s}$

Figure 6.19.: Three-dimensional visualizations of the material thickness h and elevation z for different grid resolutions for the Tafjord case.

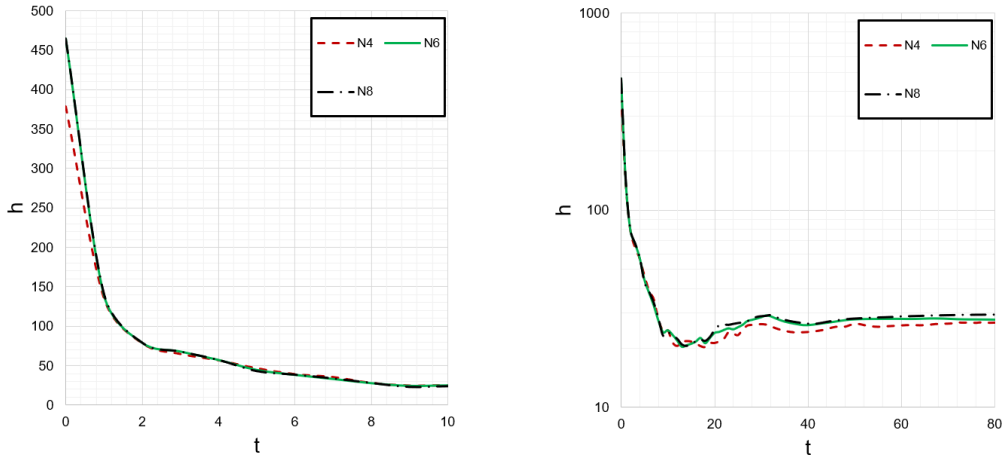
In summary, the convergence study demonstrates that coarse meshes significantly impair the representation of granular flow dynamics, especially during early collapse. The standard mesh achieves a satisfactory balance between accuracy and computational cost, while the fine mesh provides marginal gains in precision. These observations justify the choice of the standard-resolution mesh for the remainder of this study. It provides a robust framework for investigating friction effects, rheological sensitivity, and topographic influences without incurring prohibitive run-time costs. Thus, the standard configuration is adopted as the default configuration for subsequent simulations and parameter studies.

ADER-DG Order Sensitivity

To evaluate the influence of polynomial approximation order on solution quality, a sensitivity analysis with respect to the ADER-DG scheme order is performed. The study aims to ensure that numerical predictions remain stable and consistent across increasing orders of accuracy, thereby confirming that the selected solver configuration is robust and reliable.

Simulations are conducted on the standard-resolution mesh (26244 elements), while varying the ADER-DG approximation order in $N \in \{4, 6, 8\}$. Across all cases, a CFL number of 0.45 is applied. The DMP parameters (4.3) are adapted according to the order: for $N = 4$, and $N = 6$, values $k = 0.01$ and $b = 1.0$ are used, while for $N = 8$ the parameters are set to $k = 0.005$ and $b = 0.5$ to reflect the increased stiffness sensitivity at higher resolutions.

Temporal evolution of the material thickness h is evaluated at the monitoring point $(416423\,\mathrm{m}, 6906230\,\mathrm{m})$, with results shown in Figure 6.20 for both linear and logarithmic scales. The results indicate that all three solver orders capture the dominant dynamics of flow collapse and spreading. Slight deviations are observed for the $N = 4$ case, particularly in the magnitude of the initial peak, which appears underestimated. This discrepancy arises from the insufficient number of nodes in the $N = 4$ scheme to accurately extract the data from the NetCDF file at the given mesh resolution. Despite this limitation, the overall temporal trends closely match those of the $N = 6$ and $N = 8$ schemes, suggesting that the flow evolution is still well captured in an integral sense.



(a) Linear scale

(b) Logarithmic scale

Figure 6.20.: Temporal evolution of the material thickness h at the point $(416423\,\mathrm{m}, 6906230\,\mathrm{m})$ for different ADER-DG orders N for the Tafjord case.

To support these observations, Figure 6.21 presents 2D snapshots of the evolving flow field for all three considered orders. Across all orders, the granular material is observed to descend the slope and decelerate until it fully comes to rest near its base, forming a stable deposition zone. A narrow region of material consistently remains immobile near the initial release location, where the slope is relatively flat and friction dominates over gravity. The lowest-order $N = 4$ solution exhibits visible numerical diffusion and smoothing of the flow front, while higher-order schemes preserve sharper interfaces and better resolve secondary

structures.

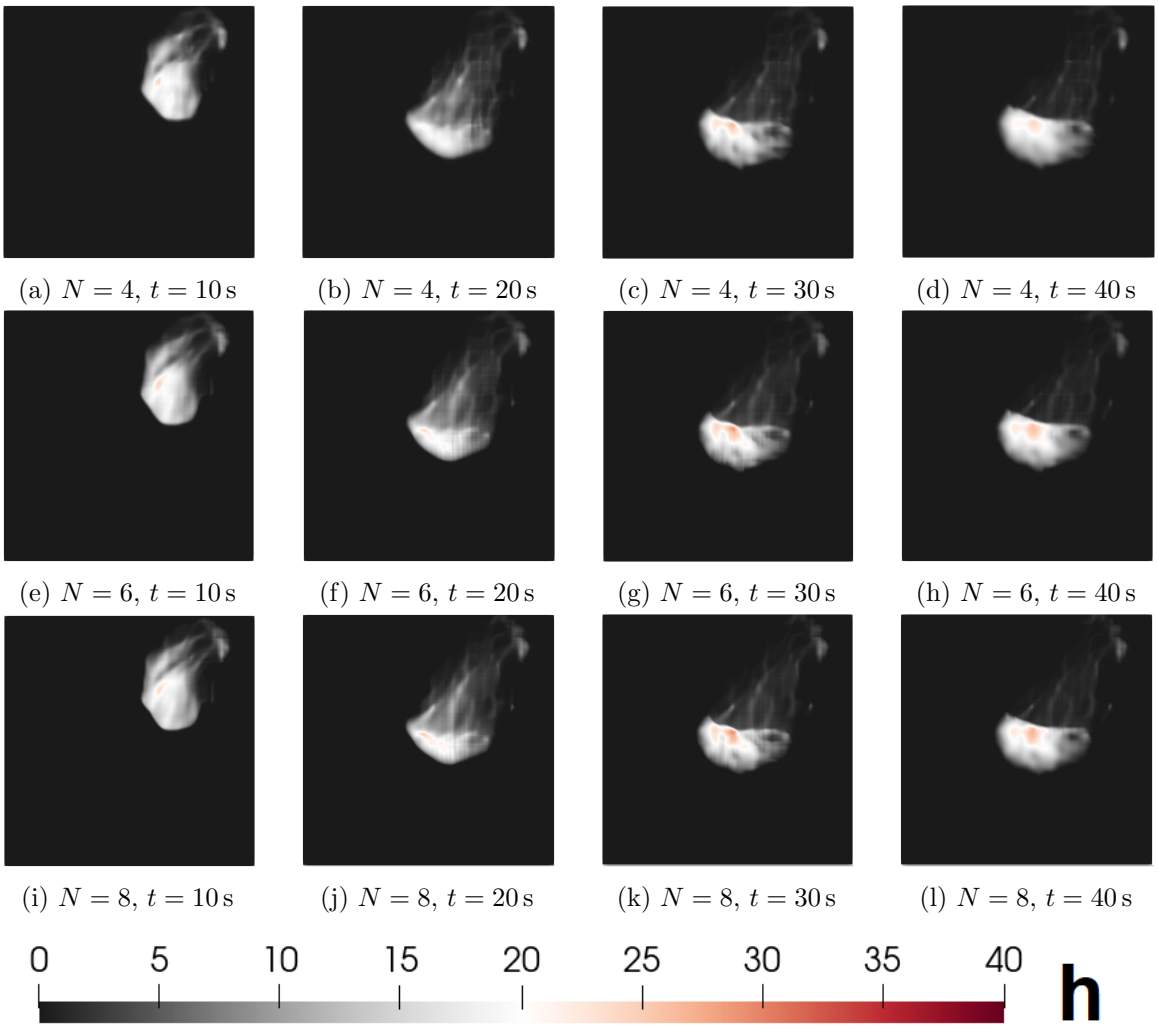
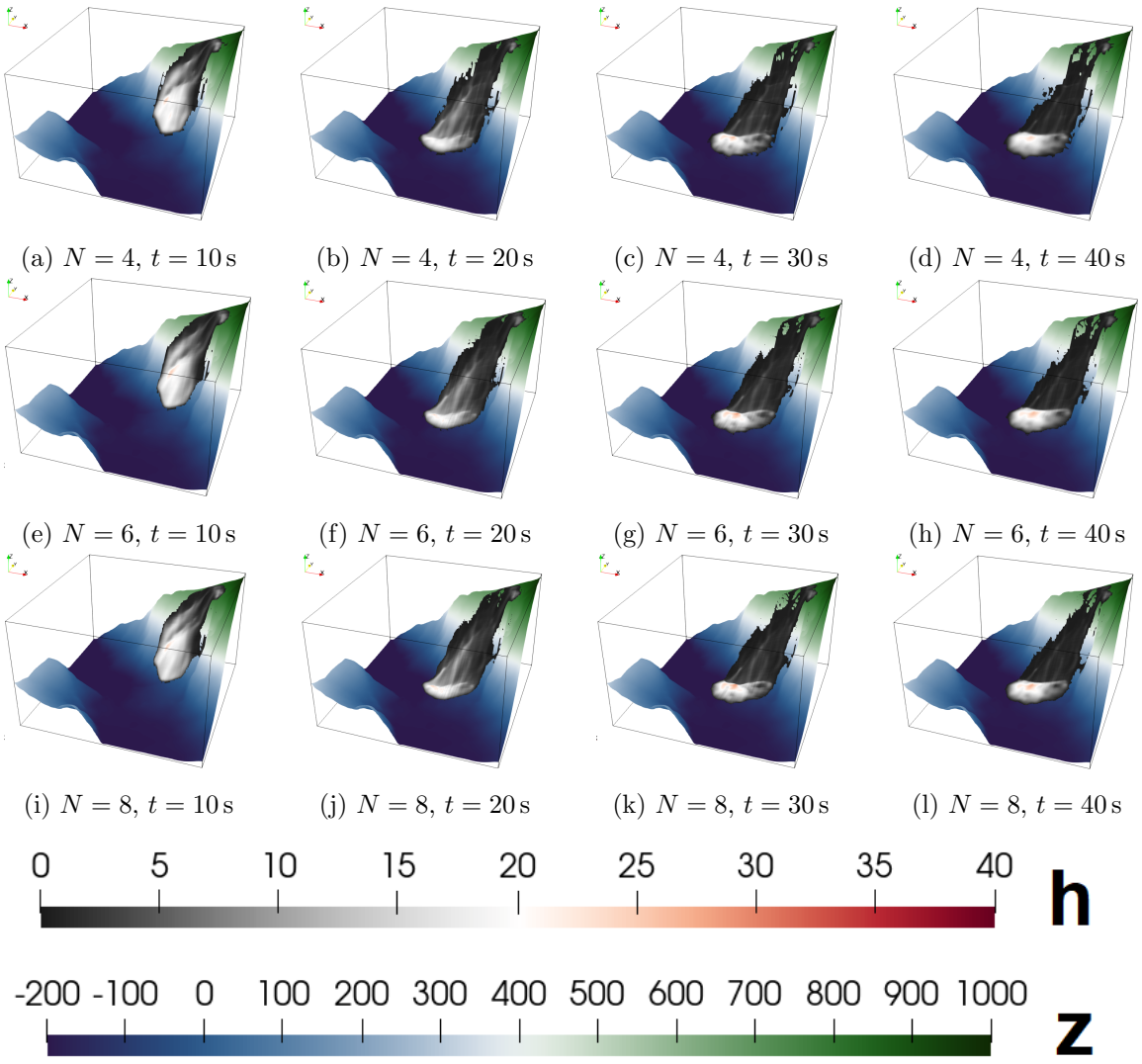


Figure 6.21.: h for 4th, 6th, and 8th orders.

Complementary three-dimensional visualizations are shown in Figure 6.22, illustrating the impact of solver order on both material distribution and terrain interaction. The 3D perspective also provides a clearer understanding of the physical mechanisms that drive the flow. It clearly shows how the granular mass separates from the initial release zone and accelerates downslope, guided by the terrain geometry. The visualizations highlight how the flow follows natural depressions and valleys in the topography before coming to rest at lower elevations. These geometric influences explain the asymmetrical spreading pattern and curved shape of the final deposit, which are less obvious in 2D projections. Higher-order methods produce sharper gradients and more pronounced from propagation, especially in regions with complex topographic curvature.



(a) $N = 4$, $t = 10\,\mathrm{s}$ (b) $N = 4$, $t = 20\,\mathrm{s}$ (c) $N = 4$, $t = 30\,\mathrm{s}$ (d) $N = 4$, $t = 40\,\mathrm{s}$

(e) $N = 6$, $t = 10\,\mathrm{s}$ (f) $N = 6$, $t = 20\,\mathrm{s}$ (g) $N = 6$, $t = 30\,\mathrm{s}$ (h) $N = 6$, $t = 40\,\mathrm{s}$

(i) $N = 8$, $t = 10\,\mathrm{s}$ (j) $N = 8$, $t = 20\,\mathrm{s}$ (k) $N = 8$, $t = 30\,\mathrm{s}$ (l) $N = 8$, $t = 40\,\mathrm{s}$

Figure 6.22.: h for 4th, 6th, and 8th orders.

In summary, this order sensitivity study confirms that ADER-DG orders $N = 6$ and $N = 8$ yield consistent and accurate results. While $N = 4$ provides qualitatively correct behavior, it introduces noticeable numerical diffusion. The sixth-order configuration offers a satisfactory trade-off between computational cost and solution fidelity and is therefore adopted for the remaining research.

6.2.2. Results

To investigate how different friction mechanisms influence granular flow over realistic terrain, a series of numerical experiments was conducted using the Tafjord topography. The following configurations were considered:

1. **No friction**: A baseline case in which the granular material evolves without any resistance, governed solely by the classical SWE model (2.10);

2. **Coulomb friction only**: Basal friction is included via a Coulomb-type formulation(2.15) with the basal friction angle $\theta = 25°$, and the turbulent drag term is deactivated ($\xi \to \infty$);

3. **Turbulent drag only**: The flow is subjected solely to velocity-squared resistance (2.17), with $\xi = 200\,\mathrm{m/s^2}$ and zero basal friction ($\theta = 0°$);

4. **Full Voellmy-Salm friction**: Both frictional mechanisms are simultaneously activated, with parameters $\theta = 25°$ and $\xi = 200\,\mathrm{m/s^2}$.

The stabilization parameters in the DMP criterion (4.3) are set to $k = 0.01$ and $b = 1.0$ for the full model, and halved ($k = 0.005, b = 0.5$) for all others to reflect the reduced stiffness in the absence of one or both frictional terms. The CFL number (3.69) is set to 0.45.

The resulting flow fields are shown in Figure 6.23 and Figure 6.24, depicting the evolution of the material thickness h at four representative time instances $t = 10\,\mathrm{s}, 20\,\mathrm{s}, 30\,\mathrm{s}, 40\,\mathrm{s}$.

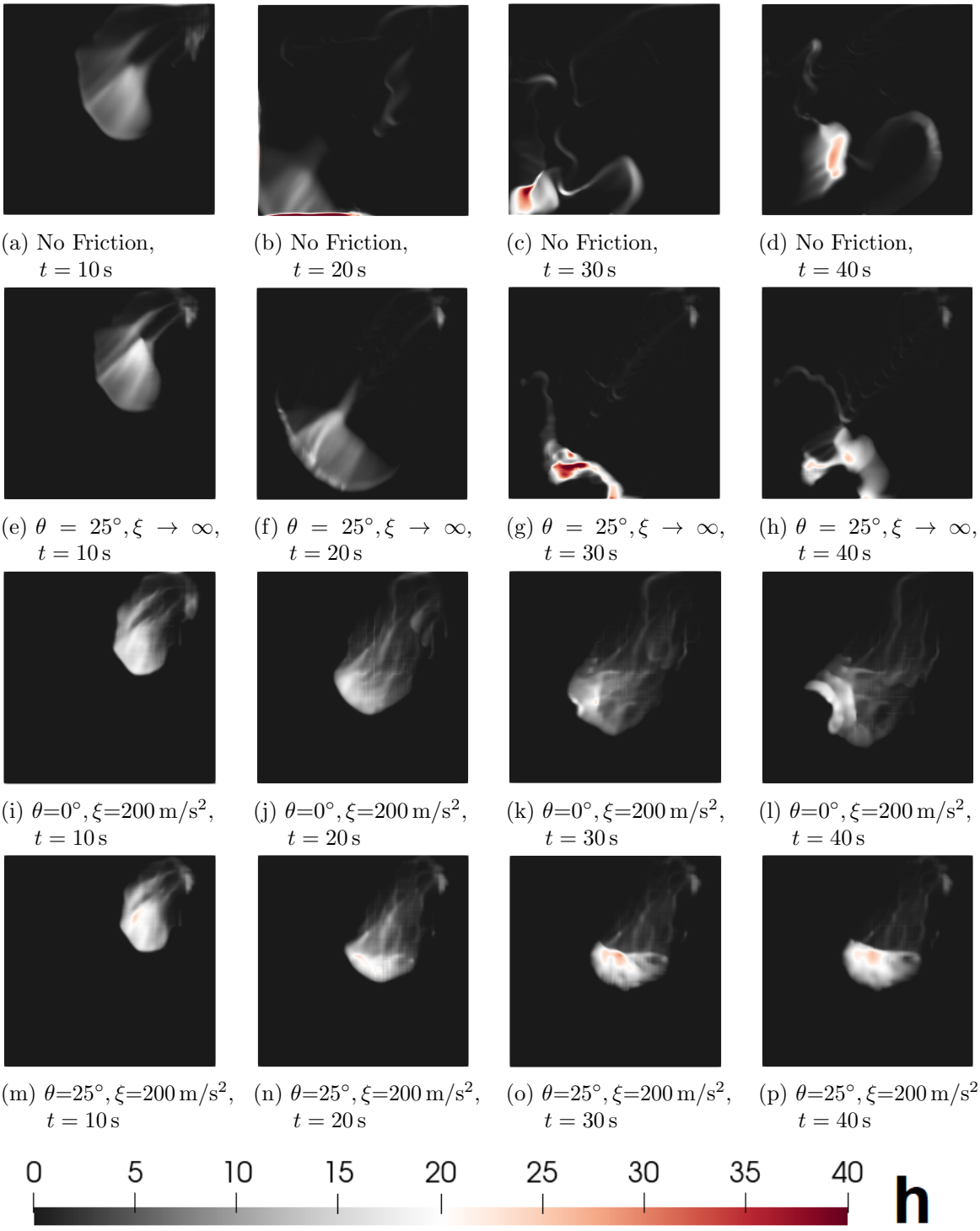
In the absence of any frictional resistance (first row), the granular material accelerates unimpeded downslope, traverses the valley, and ascends the opposing terrain. A portion of the material begins to oscillate across the valley, reflecting excessive kinetic energy and the lack of dissipative mechanisms.

Introducing only Coulomb basal friction (second row) substantially alters the flow pattern. The material no longer reaches the top of the opposite slope. Instead, due to the gravitational force and the friction term, the material starts to descend from the second slope and accumulates at its base. It is also observed that the portion of the mass remains on the initial incline for the whole simulation due to the high basal friction at this location. This results in a more compact and localized deposition.

In the third configuration, turbulent drag is applied in the absence of Coulomb-type friction. The material accelerates freely but experiences progressive dissipation due to velocity-squared resistance. The entire mass descends from the initial slope, yet halts abruptly at the valley base, unable to overcome the gradient of the second slope. The result is a more concentrated accumulation zone compared to the no-friction case, yet broader and less localized than in the Coulomb-only configuration.

Finally, the complete Voellmy-Salm model (fourth) row combines both frictional effects. The granular material dissipates nearly all kinetic energy during descent and comes to rest near the toe of the initial slope. This configuration produces the most compact and physically realistic deposition pattern, with minimal run-out and no material crossing the valley.

These results demonstrate the critical role of both Coulomb and turbulent resistance in regulating granular mobility. While Coulomb friction limits the onset of motion, turbulent drag provides continuous damping throughout the flow. The combined model yields the most restrained and stable evolution, consistent with observations of real-world avalanches and debris flows.



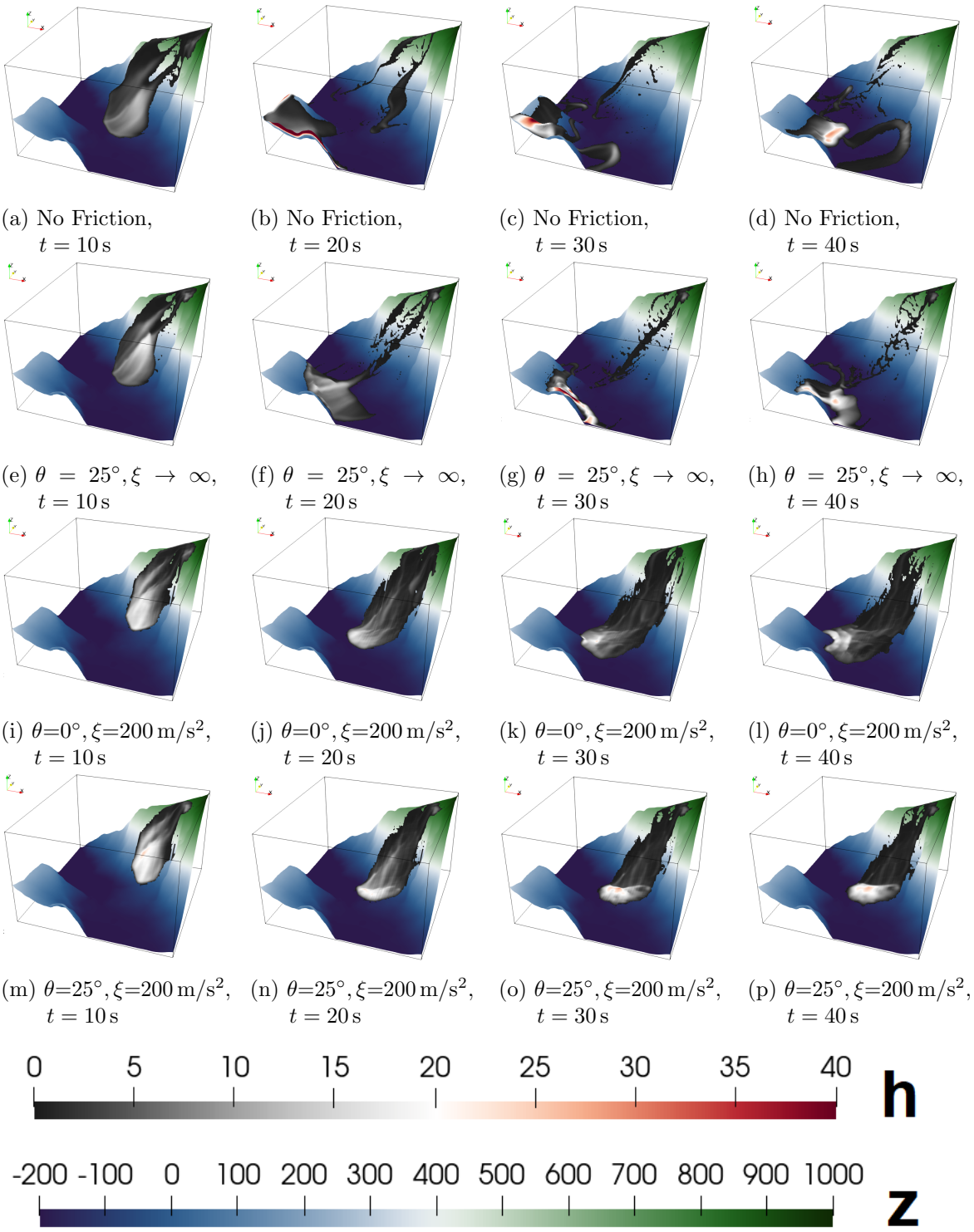
(a) No Friction, $t = 10\,\mathrm{s}$ (b) No Friction, $t = 20\,\mathrm{s}$ (c) No Friction, $t = 30\,\mathrm{s}$ (d) No Friction, $t = 40\,\mathrm{s}$

(e) $\theta = 25°, \xi \to \infty$, $t = 10\,\mathrm{s}$ (f) $\theta = 25°, \xi \to \infty$, $t = 20\,\mathrm{s}$ (g) $\theta = 25°, \xi \to \infty$, $t = 30\,\mathrm{s}$ (h) $\theta = 25°, \xi \to \infty$, $t = 40\,\mathrm{s}$

(i) θ=0°, ξ=200 m/s^2, $t = 10\,\mathrm{s}$ (j) θ=0°, ξ=200 m/s^2, $t = 20\,\mathrm{s}$ (k) θ=0°, ξ=200 m/s^2, $t = 30\,\mathrm{s}$ (l) θ=0°, ξ=200 m/s^2, $t = 40\,\mathrm{s}$

(m) θ=25°, ξ=200 m/s^2, $t = 10\,\mathrm{s}$ (n) θ=25°, ξ=200 m/s^2, $t = 20\,\mathrm{s}$ (o) θ=25°, ξ=200 m/s^2, $t = 30\,\mathrm{s}$ (p) θ=25°, ξ=200 m/s^2, $t = 40\,\mathrm{s}$

Figure 6.23.: Temporal evolution of the material thickness h at four time-steps ($t = 10\,\mathrm{s}, 20\,\mathrm{s}, 30\,\mathrm{s}, 40\,\mathrm{s}$) for different friction models on the Tafjord topography using a 2D view.



(a) No Friction, $t = 10\,\mathrm{s}$
(b) No Friction, $t = 20\,\mathrm{s}$
(c) No Friction, $t = 30\,\mathrm{s}$
(d) No Friction, $t = 40\,\mathrm{s}$
(e) $\theta = 25°, \xi \to \infty$, $t = 10\,\mathrm{s}$
(f) $\theta = 25°, \xi \to \infty$, $t = 20\,\mathrm{s}$
(g) $\theta = 25°, \xi \to \infty$, $t = 30\,\mathrm{s}$
(h) $\theta = 25°, \xi \to \infty$, $t = 40\,\mathrm{s}$
(i) θ=0°, ξ=200 m/s^2, $t = 10\,\mathrm{s}$
(j) θ=0°, ξ=200 m/s^2, $t = 20\,\mathrm{s}$
(k) θ=0°, ξ=200 m/s^2, $t = 30\,\mathrm{s}$
(l) θ=0°, ξ=200 m/s^2, $t = 40\,\mathrm{s}$
(m) θ=25°, ξ=200 m/s^2, $t = 10\,\mathrm{s}$
(n) θ=25°, ξ=200 m/s^2, $t = 20\,\mathrm{s}$
(o) θ=25°, ξ=200 m/s^2, $t = 30\,\mathrm{s}$
(p) θ=25°, ξ=200 m/s^2, $t = 40\,\mathrm{s}$

Figure 6.24.: Three-dimensional visualization of the flow evolution for the same frictional configurations and time-steps as in Figure 6.23, showing both material thickness h and topography elevation z. These visualizations highlight terrain interaction and illustrate how frictional forces influence the extent and elevation of deposited material across the domain.

7. Conclusion

This thesis presented the development, implementation, and validation of a high-order numerical framework for simulating gravity-driven granular flows governed by the Voellmy-Salm friction model. The depth-averaged Shallow Water Equations, extended with Coulomb-type basal friction and turbulent drag, were discretized using the ADER-DG method within the ExaHyPE 2 simulation engine. Several numerical challenges, including stiff source terms, non-conservative topographic interactions, and transient wetting and drying, were addressed using a combination of advanced high-order and limiting techniques.

The ADER-DG scheme enabled high-order accuracy in both space and time, making it suitable for resolving the sharp gradients and discontinuities typical of granular avalanches. A predictor-corrector approach allowed for explicit integration of the stiff friction terms, with stability ensured by a modified CFL condition adapted to the source-term stiffness. To handle non-smooth regions, an a-posteriori sub-cell Finite Volume limiter was employed. In this framework, cells identified as troubled were projected to a first-order Finite Volume scheme on sub-grids, using interface-based source term treatment.

A key contribution of this work lies in the detailed configuration and solver setup using ExaHyPE 2's Python-based interface. The modularity of the engine allowed flexible composition of numerical components and automated code generation. Specific implementation steps - including flux, source, NCP functions, time-step control, and solvers coupling - were outlined and applied to both synthetic and real-world test cases.

Validation studies confirmed the physical consistency and numerical stability of the proposed framework. In a synthetic test case involving granular collapse on a 35° inclined slope, the effects of friction were systematically explored by isolating and combining the Coulomb and turbulent drag components. These simulations revealed distinct flow behaviors - unimpeded spreading in the frictionless case, compact deposition zones under turbulent drag, and partial retention of mass on the slope with the Coulomb term. Notably, when the friction angle θ exceeded the slope inclination, the Coulomb term alone fully stopped the downslope motion, aligning with theoretical expectations.

To asses numerical reliability, a comprehensive grid convergence and order sensitivity study was conducted on the real world Tajord topography. Grid refinement (coarse to fine) demonstrated that standard and fine resolutions produce nearly identical flow profiles, while the coarse grid fails to capture key topographic and dynamic features. An additional study on ADER-DG orders ($N = 4, 6, 8$) indicated that the sixth-order scheme offers the best trade-off between accuracy and computational cost, with higher orders yielding diminishing improvements. These results justified the choice of the standard resolution and sixth-order approximation for subsequent simulations.

The validated solver was then applied to simulate granular flow over the Tafjord region using different friction configurations. In the absence of friction, material traversed the valley and oscillated across slopes, indicating energy overestimation. Introducing Coulomb friction limited downslope mobility and led to partial retention on the initial slope. With

only turbulent drag, the material advanced into the valley but halted at its base. The full Voellmy-Salm model provided the most restrained and localized deposition, confirming the necessity of both friction components for accurate hazard modeling.

Despite these advancements, certain limitations remain. The Discrete Maximum Principle (DMP) used in the sub-cell limiter relies on manually tuned stabilization parameters (k, b), which may lack generality across heterogeneous topography and flow regimes. Future work should investigate adaptive strategies for selecting these parameters dynamically based on local solution features such as gradients or curvature. Additionally, the use of the Rusanov flux at cell interfaces may limit resolution of complex wave structures; exploring more accurate Riemann solvers (e.g., HLLC or Roe-type) could improve fidelity. Enhanced wetting/drying algorithms may also contribute to greater stability in shallow flow regions.

In conclusion, this work bridges the gap between theoretical granular flow modeling and high-performance numerical simulation. The integration of high-order discretization, adaptive limiting, and real-world terrain data provides a scalable and physically consistent framework for modeling dry granular avalanches. This foundation supports further developments toward predictive tools for geophysical hazard analysis and contributes to the broader field of computational geomechanics.

List of Figures

Bibliography

[1] A. Voellmy, "Über die zerstörungskraft von lawinen," *Schweizerische Bauzeitung*, vol. 73, pp. 212–217, 1955.

[2] B. Salm, "On nonuniform, steady flow of avalanching snow," in *International Association of Scientific Hydrology Publication 79 (General Assembly of Bern 1967 — Snow and Ice)*, pp. 19–29, International Association of Scientific Hydrology, 1968.

[3] B. Salm, "Flow, flow transition and runout distances of flowing avalanches," *Annals of Glaciology*, vol. 18, pp. 221–226, 1993.

[4] R. J. LeVeque and H. C. Yee, "A study of numerical methods for hyperbolic conservation laws with stiff source terms," *Journal of Computational Physics*, vol. 86, no. 1, pp. 187–210, 1990.

[5] M. Dumbser, F. Fambri, M. Tavelli, M. Bader, and T. Weinzierl, "Efficient implementation of ader discontinuous galerkin schemes for a scalable hyperbolic pde engine," *Axioms*, vol. 7, p. 63, Sept. 2018.

[6] M. Dumbser and R. Loubère, "A simple robust and accurate a posteriori sub-cell finite volume limiter for the discontinuous galerkin method on unstructured meshes," *Journal of Computational Physics*, vol. 319, pp. 163–199, Aug. 2016. See pp. 1, 16, 17.

[7] F. Bouchut, H. Ounaissa, and B. Perthame, "Upwinding of the source term at interfaces for euler equations with high friction," *Computers & Mathematics with Applications*, vol. 53, pp. 361–375, Feb. 2007.

[8] 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*, 1871.

[9] M. O. Bristeau, A. Mangeney, J. Sainte-Marie, and N. Seguin, "An energy-consistent depth-averaged euler system: derivation and properties," *Journal of Computational Physics*, 2016.

[10] A. Dufresne, A. Bösmeier, and C. Prager, "Sedimentology of rock avalanche deposits – case study and review," *Earth-Science Reviews*, vol. 163, pp. 234–259, 2016.

[11] S. Hergarten, "Scaling between volume and runout of rock avalanches explained by a modified voellmy rheology," *Earth Surface Dynamics*, vol. 12, pp. 219–229, 2024.

[12] M. Dumbser, D. S. Balsara, E. F. Toro, and C.-D. Munz, "A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes," *Journal of Computational Physics*, vol. 227, pp. 8209–8253, Sept. 2008.

[13] M. Dumbser, I. Peshkov, E. Romenski, and O. Zanotti, "High order ader schemes for a unified first order hyperbolic formulation of continuum mechanics: Viscous heat-conducting fluids and elastic solids," *Journal of Computational Physics*, vol. 314, pp. 824–862, June 2016.

[14] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Mathematics of Computation*, vol. 23, no. 106, pp. 221–230, 1969.

[15] G. Dahlquist and Åke Björk, *Numerical Methods*. Prentice-Hall, 1974. Equidistant Interpolation and the Runge Phenomenon.

[16] F. Fraysse and R. Saurel, "Automatic differentiation using operator overloading (adoo) for implicit resolution of hyperbolic single phase and two-phase flow models," *Journal of Computational Physics*, vol. 399, p. 108942, 2019.

[17] M. Dumbser, "Arbitrary high order pnpm schemes on unstructured meshes for the compressible navier–stokes equations," *Computers & Fluids*, vol. 39, no. 1, pp. 60–76, 2010.

[18] G. Gassner, F. Lörcher, and C.-D. Munz, "A discontinuous galerkin scheme based on a space-time expansion ii. viscous flow equations in multi dimensions," *Journal of Scientific Computing*, vol. 34, no. 3, pp. 260–286, 2008.

[19] S. K. Godunov, *Different Methods for Shock Waves*. Ph.d. dissertation, Moscow State University, Moscow, 1954.

[20] S. K. Godunov, "A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations," *Matematicheskii Sbornik*, vol. 47, pp. 271–306, 1959. English translation: US Joint Publ. Res. Service, JPRS 7226, 1969.

[21] D. Gottlieb and C.-W. Shu, "The gibbs phenomenon and its resolution," *SIAM Review*, vol. 39, no. 4, pp. 644–668, 1997.

[22] Y. Nakayama, "Chapter 15 - computational fluid dynamics," in *Introduction to Fluid Mechanics (Second Edition)* (Y. Nakayama, ed.), pp. 293–327, Butterworth-Heinemann, second edition ed., 2018.

[23] A. Kurganov and G. Petrova, "A second-order well-balanced positivity preserving central-upwind scheme for the saint-venant system," *Communications in mathematical sciences*, vol. 5, 03 2007.

[24] M. Dumbser, C. Enaux, and E. F. Toro, "Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws," *Journal of Computational Physics*, vol. 227, no. 8, pp. 3971–4001, 2008.