Digital Animation of Powder-Snow Avalanches

FILIPE NASCIMENTO, ICMC-USP, Brazil
FABRICIO S. SOUSA, ICMC-USP, Brazil
AFONSO PAIVA, ICMC-USP, Brazil

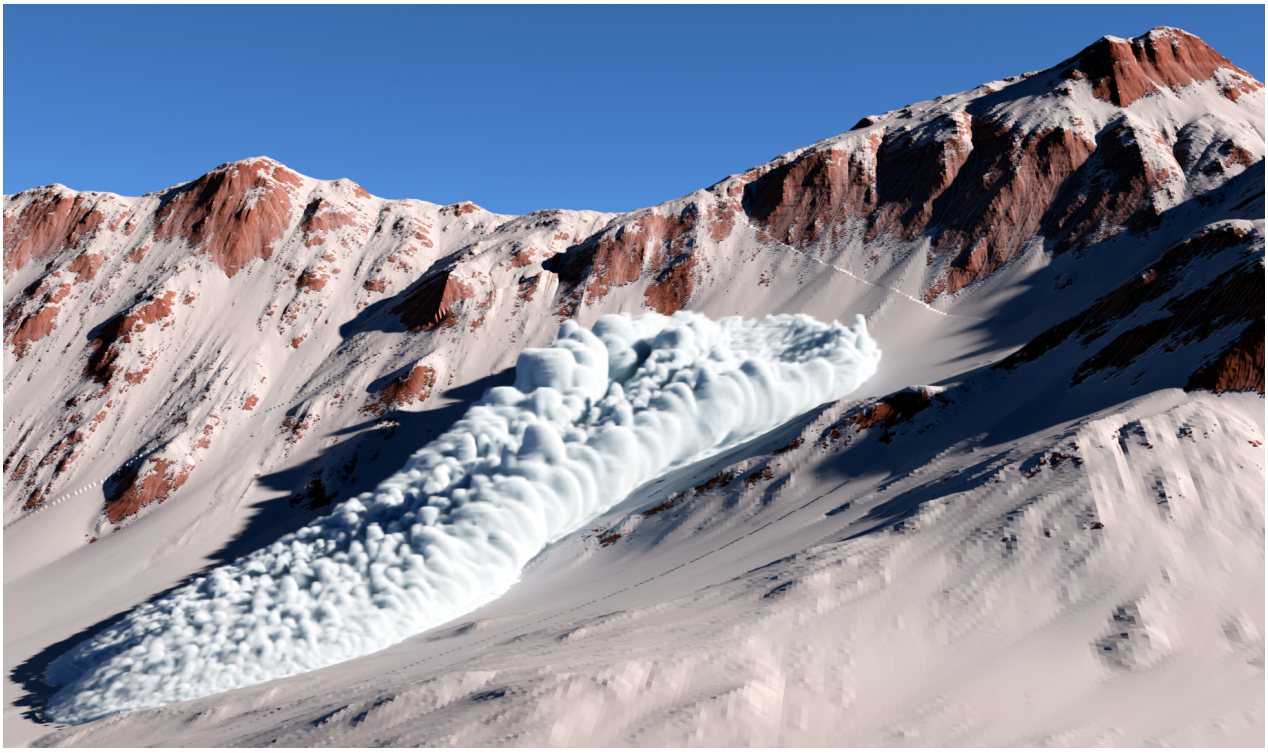


Fig. 1. A large-scale simulation of a powder-snow avalanche using the Finite Volume Method in a mesh with 17.8M cells and a total volume of 0.43 km^3. A turbulent powder-snow cloud forms as the avalanche sweeps down a procedural mountain along a traveled distance of 1.5 km. The plume billow structures arise due to a novel procedural snow entrainment process modeled by our method.

Powder-snow avalanches are natural phenomena that result from an instability in the snow cover on a mountain relief. It begins with a dense avalanche core moving fast down the mountain. During its evolution, the snow particles in the avalanche front mix with the air, forming a suspended turbulent cloud of snow dust surrounding the dense snow avalanche. This paper introduces a physically-based framework using the Finite Volume Method to simulate powder-snow avalanches under complex terrains. Specifically, the primary goal is to simulate the turbulent snow cloud dynamics within the avalanche in a visually realistic manner. Our approach relies on a multi-layer model that splits the avalanche into two main layers: dense and powder-snow. The dense-snow layer flow is simulated by solving a type of Shallow Water Equations suited for intricate basal surfaces, known as the Savage-Hutter model. The powder-snow layer flow is modeled as a two-phase mixture of miscible fluids and simulated using Navier-Stokes equations. Moreover, we propose a novel model for the transition layer, which is responsible for coupling the avalanche main layers, including the snow mass injected into the powder-snow cloud from the snow entrainment processes and its injection velocity. In brief, our framework comprehensively simulates powder-snow avalanches, allowing us to render convincing animations of one of the most complex gravity-driven flows.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Powder-Snow Avalanches, Finite Volume Method, Savage-Hutter Model, Fluid Simulation, Computer Animation

ACM Reference Format:
Filipe Nascimento, Fabricio S. Sousa, and Afonso Paiva. 2025. Digital Animation of Powder-Snow Avalanches. *ACM Trans. Graph.* 44, 4 (August 2025), 20 pages. https://doi.org/10.1145/3730862

Authors' addresses: Filipe Nascimento, filipedecn@gmail.com, ICMC-USP, São Carlos, SP, Brazil; Fabricio S. Sousa, fsimeoni@icmc.usp.br, ICMC-USP, São Carlos, SP, Brazil; Afonso Paiva, apneto@icmc.usp.br, ICMC-USP, São Carlos, SP, Brazil.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3730862.

1 INTRODUCTION

Natural hazard simulations, such as earthquakes [Chourasia et al. 2007], tsunamis [Wu et al. 2018], landslides [Li et al. 2024], storms [Hädrich et al. 2020], volcano eruptions [Pretorius et al. 2024], hurricanes [Herrera et al. 2024], and other events of Mother Earth's wrath, have also attracted significant attention in the computer animation field and digital VFX industry. In particular, snow avalanches are incredibly complex large-scale phenomena to model and simulate. One of the main challenges of the feature film *"Black Widow"* from Marvel Studios, as reported by Wētā FX, was to simulate a visually realistic snow avalanche over a kilometer long [Failes 2021].

Avalanches arise from an instability in the snow cover on a mountain relief, resulting from the accumulation of several snowfalls that occur over time. The slope angle is the most crucial terrain factor influencing avalanche release. For instance, avalanches can start on more than 25° slopes. Beyond the terrain's inclination, other factors can influence the release process, such as ground surface roughness, snow-pack structure, and weather conditions [Ancey 2001]. Moreover, snow avalanches can be classified according to their motion [Dutykh et al. 2011]:

- **Flowing avalanche** has a high-density core made of snow and heavy ice at the bottom, leading to a gravity-driven laminar flow, where the mountain relief influences its trajectory. This dense snow layer accelerates quickly, increasing in mass and volume as more snow becomes entrained;
- **Powder-snow avalanche** begins with a flowing snow avalanche moving down a relief. During its development, the snow particles mix with the air, forming a suspended cloud of powder-snow, i.e., a particle-laden *gravity current* [Simpson 1999]. Meaning the gravitational field is constrained to flow horizontally, driven by a fluid density difference. Furthermore, the gravity current creates fluid turbulence that suspends fine snow dust in the air, similar to the enormous clouds made up of particles and gases launching into the atmosphere during a volcano eruption.

We introduce a numerical framework based on the Finite Volume Method to simulate the dynamics of a powder-snow avalanche under complex terrains. Our framework separates the avalanche into two main layers: the dense and powder-snow layers. The dense-snow layer flow is simulated by solving a type of Shallow Water Equations suited for complex terrains, known as the *Savage-Hutter model*. The powder-snow layer flow is modeled as a miscible mixture of two fluids and simulated using the incompressible Navier-Stokes equations discretized with second-order accuracy in space. In addition, we introduce a numerical model for coupling the avalanche layers, encompassing the snow mass injected into the powder-snow cloud from the snow entrainment processes and its injection velocity. Fig. 1 shows our method in action.

Contributions. In summary, our key contributions are:

- We present a comprehensive two-layer framework for physically-based simulation of powder-snow avalanches, which includes the interaction of the dense-snow avalanche with the terrain and the turbulent motion of the powder-snow cloud in suspension (Section 6);

Fig. 2. Avalanche in the mountains (adapted from photo by [Künnap 1985]).

- A transition layer model based on a novel procedural snow entrainment mechanism that enables the coupling between the dense-snow layer and the powder-snow layer, allowing the transference of mass and momentum to the powder-snow layer (Sections 4.3 and 6.2);
- Since the frontal region of the avalanche feeds an energetic turbulent powder-snow cloud, we introduce a fast algorithm for computing the avalanche front distance field, which impacts the formation of the plume structures (Section 7.1).
- We present an accurate partition of unity interpolation scheme for mesh data conversion to render the powder-snow clouds on large terrains. (Section 7.2).

2 POWDER-SNOW AVALANCHE IN A NUTSHELL

In literature, a powder snow-avalanche (PSA), a.k.a. *mixed-motion avalanche*, usually consists of two main layers [Sovilla et al. 2015; Turnbull and Bartelt 2003]: the *dense-snow layer* (DSL) and the *powder-snow layer* (PSL), representing the flowing avalanche and the powder avalanche, respectively. In addition, two extra layers are used to explain the changes in mass observed in the DSL and PSL. The first layer, referred to here as the *ground layer*, consists of the snow cover available in the terrain. The second layer, referred to as the *transition layer* (TL), appears between the DSL and the PSL. This layer is responsible for the exchange of mass and momentum between the two main layers. Table 1 lists the four mentioned layers and their main characteristics.

Table 1. Layers of a powder-snow avalanche.

layer	description	flow type
PSL	suspension layer	turbulent particle-laden flow
TL	two-phase viscous wall layer	two-phase flow
DSL	flowing avalanche layer	laminar flow
Ground	stagnant snow layer	snow at rest

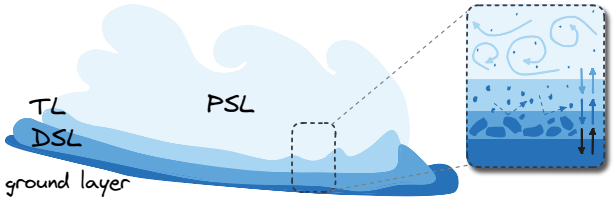


Fig. 3. PSAs can be dissected into four main layers of different flow types. The DSL comprises big snow packs that break into smaller pieces as the flow develops. Particles at the fluidized DSL surface are stirred up, forming the TL. Small particles may be suspended by turbulent air and form the PSL, which is self-accelerated by gravity and turbulence. The DSL exchanges mass with the static snow cover that forms the ground layer by entrainment and deposition processes.

The sequential events of a PSA happen along three distinct phases that roughly split the avalanche path. The *starting zone* is the top of the avalanche path, where the block of snow detaches from the mountain surface and starts to slide down the slope. The *track* is the middle part of the path where the avalanche slides, accelerating and increasing in size by entraining snow from the ground through an erosion process of the snow cover. The amount of entrained snow will influence how fast and long the flow will advance. Finally, the *runout zone* is the bottom of the avalanche path, where the avalanche starts to slow down until it stops depositing snow. The maximum traveled distance by the avalanche is known as *runout distance*. Figure 2 shows a photo of a PSA track in the mountains.

During a PSA, the DSL may gain mass from the ground layer by snow entrainment and lose mass from snow deposition processes. However, there are also exchanges between the DSL and the PSL. As collisions fracture large chunks of snow, the DSL surface becomes fluidized. The increasing wind causes particles to leave the DSL surface in ballistic motion, creating a transition region. Some particles go up to the PSL, where collisions are less frequent and get into strong aerodynamic turbulence. Figure 3 depicts the PSA anatomy.

The snow entrainment plays a central role in the evolution of an avalanche flow, particularly in a PSA. The entrainment process feeds the PSL, dictating the powder-snow cloud's size. There are some fundamental mechanisms of entrainment in snow avalanches [Issler 2014; Li et al. 2022; Sovilla et al. 2006]:

- The frontal *plowing* occurs when the dense avalanche core incurs into the snow cover, pushing and displacing it. The amount of entrained snow depends on the depth of the snow cover, its strength, and the speed of the avalanche;
- The snow *eruption* caused by the frontal compression force of the avalanche creates a high pore pressure that pushes the interstitial air out. The drag caused by the airflow displaces the snow upwards, creating the typical frontline of PSAs. This process happens quickly, between 0.1 and 2 seconds, with significant entrainment rates up to 350 kg/m^2s;

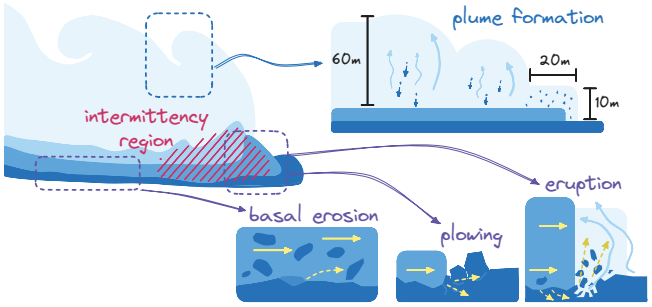


Fig. 4. The various entrainment processes present in a PSA. Away from the front, the continuous friction forces in the ground surface cause scour and ripping. In the leading edge, snow can be plowed frontwards but can also be violently ejected into the air by eruption. Plumes are born from the snow eruption, growing as heavy particles settle down and displace air upwards. The intermittency region includes surges of rapid flows and produces the oscillatory behavior of the PSA front.

- In *basal erosion*, the avalanche rasps mass from the ground layer in proportion to the shearing force that the avalanche exerts on the basal surface. As erodible snow loses cohesion and strength, the chunks of snow break into smaller pieces, getting captured in the avalanche flow. This process occurs away from the front and at low entrainment rates of 10 kg/m^2s.

The plowing and eruption mechanisms are essential for forming plume structures in a powder-snow cloud. Some researchers [Bartelt et al. 2013] observed that after an initial plume quickly achieves $10m$ of height, the growth pauses and resumes as the plume is no longer at the avalanche front (at least $20m$ behind). The air intake in the front, which is velocity-dependent, expands the volume of the cloud. As the ejected heavy particles fall, the entrained air is displaced upwards, suspending the ice dust up to $60m$ of height. Another ingredient for the *plume formation* is the oscillatory behavior in the *intermittency region* [Sovilla et al. 2018], where the frequency of plume generation is approximately $0.4Hz$ (plumes per second). Figure 4 shows the snow entrainment mechanisms in PSAs.

3 RELATED WORK

To better contextualize our approach, we organize the existing methods for snow avalanches simulation into two groups related to Computational Geoscience and Computer Graphics literature, highlighting the strengths and weaknesses of prior methods.

PSA in Computational Geoscience. Most 3D computational frameworks in this field adopt *depth-averaged models* to simulate the avalanche core using a wide variety of numerical methods for Partial Differential Equations (PDEs): Finite Difference Method (FDM) [Wang et al. 2004], Finite Volume Method (FVM) [Christen et al. 2010; Rauter et al. 2018], Smoothed Particle Hydrodynamics (SPH) [Sampl and Granig 2009], and Material Point Method (MPM) [Guillet et al. 2023]. Based on Shallow Water Equations (SWE), depth-averaged methods ignore the spatial variation in flow depth direction to reduce the computational effort. Thus, these methods can efficiently compute the snow deposition, velocity core, and impact pressure.

In particular, one of these frameworks, *Rapid Mass Movements Simulator* (RAMMS) [Christen et al. 2010], is currently the most popular commercial avalanche software. Other numerical approaches applied FDM [Mergili et al. 2017] or MPM [Li et al. 2021] to discretize elastoplastic models to simulate dense snow avalanches. However, elastoplastic and depth-averaged models do not compute the dynamics of the powder cloud. In order to reproduce the behavior of a gravity current in PSL, some 2D frameworks simulated the interaction between powder-snow and air as a mixture of two fluids modeled using Finite Volumes/Elements [Calgaro et al. 2015; Dutykh et al. 2011; Etienne et al. 2006, 2004]. The different regime flows in PSAs can be modeled as a *two-layer model*, i.e., the DSL is simulated using a depth-averaged model, while the PSL is modeled using a liquid mixture model [Issler 1998]. Recently, Gurjar [2023] proposed a 2D framework using an improved RAMMS [Bartelt et al. 2016] for DSL and an FVM-based solver of a quasi-compressible mixture modeled by Fick's law [Dutykh et al. 2011]. The main difficulty in two-layer flows relies on modeling the momentum and mass transference between the layers. As discussed above, 3D simulation of a PSA remains a great challenge in the Geoscience field.

Snow avalanches in Graphics. Despite snow avalanche simulations being visually appealing in VFX footages [Imageworks 2020; Kapler 2003; Kim and Flores 2008] and snow simulation is a well-established research area [Gissler et al. 2020; Stomakhin et al. 2013], only some works in Computer Graphics tackle snow avalanche dynamics. Tsuda et al. [2010] introduced a method to simulate a PSA computing the interaction between the DSL modeled using SPH and the PSL discretized using a stable but dissipative grid-based method [Stam 1999]. Despite its pioneering, this work has some drawbacks: limited to short runout distances, the numerical diffusion of physical quantities caused by the particle-to-grid transferences in the overlapping domain, and the high memory footprint. Their method becomes impractical for high-resolution discretizations of both grid and particles, mainly in large-scale terrains. Cordonnier et al. [2018] proposed a method for generating snow-covered landscapes produced by avalanches. They used a 2D Eulerian height field method based on a hydrostatic pipe model to simulate wet and dry avalanches, not including PSA. Liu et al. [2021] presented a Position-based Dynamics (PBD) framework combined with the Bingham viscoplastic model to simulate wet snow (*slab*) avalanches and a level set-based model to perform collisions between the snow flow and the terrain relief. However, their framework does not simulate the powder-snow cloud dynamics. The proposed method simplifies the PSA dynamics by applying a drag force to the slab avalanche to produce a snow fog effect for rendering purposes during post-processing. The drag force ignores details of the fluid flow, including wake structures and turbulence, which are crucial for plume formation. Furthermore, the prior methods in Computer Graphics represent the powder-snow cloud as a single-phase flow, i.e., these models assume instantaneous mixing, neglecting the intricacies of mass transport.

4 GOVERNING EQUATIONS

Here, we briefly present the governing equations and theoretical background for modeling a PSA. As described in Section 2, a PSA consists of four main layers: a ground layer representing the snow cover encountered in the mountain, the DSL, the TL, and the PSL. In order to guide our choices, we make the following assumptions:

Table 2. Notations used in DSL and PSL models.

layer	notation	meaning	unit
both	t	time	s
	$\mathbf{x}$	position	m
	$\mathbf{g}$	gravitational acceleration	kg/ms^2
DSL	$\bar{\mathbf{u}}$	depth-averaged flow velocity	m/s
	$\bar{p}$	basal pressure	kg/ms^2
	h	avalanche flow height	m
	h_s	snow cover thickness	m
	$\bar{\rho}_s$	snow density	kg/m^3
	q_{er}	snow entrainment rate	kg/ms^2
	$\boldsymbol{\tau}_b$	basal shear stress	kg/ms^2
	υ	Voellmy dry friction	–
	ξ	Voellmy dynamic friction	m/s^2
	E_b	specific erosion energy	m^2/s^2
PSL	$\hat{\mathbf{u}}$	air-snow mixture velocity	m/s
	$\hat{p}$	air-snow mixture pressure	kg/ms^2
	ρ	air-snow mixture density	kg/m^3
	$\hat{\rho}_k$	reference density of phase k	kg/m^3
	μ	air-snow mixture viscosity	kg/ms
	$\hat{\mu}_k$	reference viscosity of phase k	kg/ms
	α	powder-snow volume fraction	–
	α_a	ambient air volume fraction	–
	Γ	diffusion coefficient	m^2/s

A1. *The layers' physical characteristics are sufficiently different to justify different governing equations to explain their respective flow regime.*
A2. *The TL is considered a rough wall that coincides with the surface avalanche of the DSL. The wall moves at the same speed as the DSL.*
A3. *Neither the turbulent air nor the deposition processes of the PSL are significant enough to have any effect on the motion or mass of the DSL. Therefore, the influence of the PSL in the DSL is negligible.*

The assumption **A1** is more of a necessity in the face of the complexity of the whole phenomenon; it allows the application of more appropriate models for each type of flow. On the other hand, assumption **A2** enormously simplifies the method by removing the explicit TL representation, which is particularly challenging to model. In practice, we simulate only the DSL and PSL; the TL is implicitly modeled as boundary conditions for both layers. Finally, assumption **A3** further simplifies the overall setting. The direct consequence of **A3** is that the resulting system becomes a one-way coupling system of simulations, meaning that the output data of the DSL simulation serves as input for the PSL.

The DSL and PSL models are described by PDEs derived from physical laws of conservation of mass and momentum. Table 2 lists physical quantities involved in the models. Next, we will describe the governing equations of each main layer.

4.1 Dense-Snow Layer Model

This section describes the mechanical model for simulating the dense flow avalanche corresponding to the DSL. The key idea is to consider that the DSL behaves as a granular material that undergoes high deformation. The steep slope of the mountain tests the basal friction against the gravitational force. Moreover, internal friction also determines the basal surface's erosion and the motion of granular flow by generating heat and resisting deformations.

Our approach resorts to an improved SWE-based model to describe the granular flow, the Savage-Hutter (SH) model [Savage and Hutter 1991]. The SH extends the SWE by introducing a Coulomb-like basal friction and a yield criterion to handle the internal friction. Additionally, the SH tackles the inherent limitation of the SWE for steep slopes by describing its equations with a local curvilinear coordinate system. However, modeling granular flows on complex basal surfaces is not trivial since the basal pressure field is derived from centrifugal forces, which depend on the surface curvature.

In order to employ a curvature-free description for complex reliefs and to avoid complicated governing equations induced by curvilinear coordinate transformations, we adopt a *surface PDE* (SPDE) version of SH equations [Rauter et al. 2018; Rauter and Tuković 2018]. Let $\mathcal{S}_b \subset \mathbb{R}^3$ be the *basal surface* that represents the mountain terrain, the governing equations defined for all points $\mathbf{x}_b \in \mathcal{S}_b$ can be written in Cartesian coordinates as follows:

SPDE-based Savage-Hutter model

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\bar{\mathbf{u}}) = \frac{q_{\text{er}}}{\bar{\rho}_s} \tag{1}$$

$$\frac{\partial (h\bar{\mathbf{u}})}{\partial t} + \mathbf{P}^{\parallel}_{\mathbf{n}_b} \nabla \cdot (h\,\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) = -\frac{\boldsymbol{\tau}_b}{\bar{\rho}_s} + h\,\mathbf{P}^{\parallel}_{\mathbf{n}_b}\mathbf{g} - \frac{1}{2\bar{\rho}_s}\mathbf{P}^{\parallel}_{\mathbf{n}_b}\nabla(h\,\bar{p}) \tag{2}$$

$$\mathbf{P}^{\perp}_{\mathbf{n}_b} \nabla \cdot (h\,\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) = h\,\mathbf{P}^{\perp}_{\mathbf{n}_b}\mathbf{g} - \frac{1}{2\bar{\rho}_s}\mathbf{P}^{\perp}_{\mathbf{n}_b}\nabla(h\,\bar{p}) - \frac{\bar{p}}{\bar{\rho}_s}\mathbf{n}_b \tag{3}$$

The unknown fields at $\mathcal{S}_b$ are the avalanche height $h \in \mathbb{R}$, the depth-averaged flow velocity $\bar{\mathbf{u}} = (\bar{u}, \bar{v}, \bar{w}) \in \mathbb{R}^3$, and the basal pressure $\bar{p} \in \mathbb{R}$ (see Figure 5).

Equation (1) refers to continuity equation, Equations (2) and (3) are the surface-tangential and surface-normal counterparts of the momentum equation, respectively. The normal projection matrix $\mathbf{P}^{\perp}_{\mathbf{n}_b}$ and the tangential projection matrix $\mathbf{P}^{\parallel}_{\mathbf{n}_b}$ are given by:

$$\mathbf{P}^{\perp}_{\mathbf{n}_b} = \mathbf{n}_b \otimes \mathbf{n}_b \quad \text{and} \quad \mathbf{P}^{\parallel}_{\mathbf{n}_b} = \mathbf{I}_3 - \mathbf{P}^{\perp}_{\mathbf{n}_b},$$

where $\mathbf{n}_b \in \mathbb{R}^3$ is the surface normal vector at $\mathbf{x}_b$, $\mathbf{I}_3$ is the identity matrix of order 3, and $\otimes$ denotes the outer product. The basal shear stress $\boldsymbol{\tau}_b$ follows the *Voellmy friction model* [Christen et al. 2010; Voellmy 1955] defined by:

$$\boldsymbol{\tau}_b = \begin{cases} \frac{\bar{\mathbf{u}}}{\|\bar{\mathbf{u}}\|_2}\left[\upsilon\,\bar{p} + \bar{\rho}_s\, g \frac{\|\bar{\mathbf{u}}\|_2^2}{\xi}\right], & \|\bar{\mathbf{u}}\|_2 \neq 0 \\ \mathbf{0}, & \|\bar{\mathbf{u}}\|_2 = 0 \end{cases}, \tag{4}$$

where $\mathbf{0} \in \mathbb{R}^3$ is the null vector and $g = \|\mathbf{g}\|_2$.

The right-hand side of the modified continuity equation (1) appears due to the mass growth caused by the entrainment process that occurs on potential erodible snow cover h_s in the ground layer [Fischer et al. 2015]. The *entrainment rate* is defined empirically as

$$q_{\text{er}} = \begin{cases} \frac{\boldsymbol{\tau}_b \cdot \bar{\mathbf{u}}}{E_b}, & h_s > 0 \\ 0, & h_s = 0 \end{cases}. \tag{5}$$

For larger values of the erosion energy E_b, the entrainment rate generates a gradual basal erosion, while for small values, the entrainment rate increases considerably, entraining the entire snow cover at the frontal flow as a plowing process.

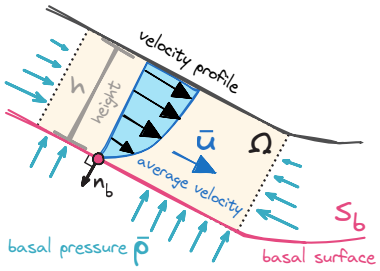


Fig. 5. Depth-averaged velocity $\bar{\mathbf{u}}$, avalanche height h and basal pressure $\bar{p}$ on a control volume Ω. The velocity $\bar{\mathbf{u}}$ is parallel to the basal surface S_b, replacing the velocity profile. The flow height h is measured normal to S_b.

4.2 Powder-Snow Layer Model

We model the PSL as a two-phase mixture of miscible fluids formed by ambient air and powder-snow in suspension (a snow aerosol). The mixture model represents the multiphase flow as a single-phase fluid, i.e., the two fluid phases are assumed to move at the same average velocity achieved by solving a single momentum equation. In mixture flows, a standard approach is to utilize *volume fractions* [Dutykh et al. 2011; Xu et al. 2023].

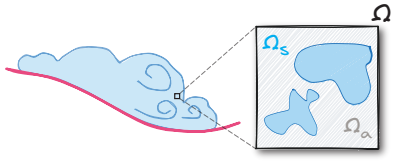


Fig. 6. The powder-snow phase Ω_s and ambient air phase Ω_a can occupy the same volume control Ω in the PSL.

Let $\Omega \subset \mathbb{R}^3$ be a volume control, $\Omega_s \subset \Omega$ the portion occupied by the powder-snow particles, and $\Omega_a \subset \Omega$ the portion occupied by the air (see Figure 6), such that

$$\text{volume}(\Omega) = \text{volume}(\Omega_s) + \text{volume}(\Omega_a). \tag{6}$$

For all $\mathbf{x} \in \Omega$, the volume fractions $\alpha(\mathbf{x}, t), \alpha_a(\mathbf{x}, t) \in [0, 1]$ of the powder-snow and the air, respectively, are computed as follows:

$$\alpha(\mathbf{x}, t) = \frac{\text{volume}(\Omega_s)}{\text{volume}(\Omega)} \quad \text{and} \quad \alpha_a(\mathbf{x}, t) = \frac{\text{volume}(\Omega_a)}{\text{volume}(\Omega)}.$$

Thus, from Equation (6), we have $\alpha_a(\mathbf{x}, t) = 1 - \alpha(\mathbf{x}, t)$. The following convex combination defines the density and the viscosity of the air-snow mixture:

$$\rho = \alpha\,\hat{\rho}_s + (1 - \alpha)\hat{\rho}_a \quad \text{and} \quad \mu = \alpha\,\hat{\mu}_s + (1 - \alpha)\hat{\mu}_a. \tag{7}$$

For simplicity, we assume that the resulting air-snow mixture is an incompressible Newtonian fluid. Therefore, the governing equation for the mixture flow can be expressed as

Mixture flow model

$$\nabla \cdot \hat{\mathbf{u}} = 0 \tag{8}$$

$$\frac{\partial(\rho\hat{\mathbf{u}})}{\partial t} + \nabla \cdot (\rho\, \hat{\mathbf{u}} \otimes \hat{\mathbf{u}}) = -\nabla p + \mu\Delta\hat{\mathbf{u}} + \rho\mathbf{g} \tag{9}$$

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha\, \hat{\mathbf{u}}) = \Gamma\Delta\alpha \tag{10}$$

Equations (8) and (9) are the incompressible Navier-Stokes in conservative form. The conservation of the volume fraction α gives rise to an additional convection-diffusion equation (10). The right-hand side of Equation (10) models the diffusive behavior within the mixture, i.e., the dilution of the snow dust by the air. This model relies on *Fick's law* [Etienne et al. 2004] and is used to predict how the concentration (volume fraction) of each mixture constituent varies over time, moving from a region of high concentration to a low concentration across a concentration gradient. The diffusion coefficient Γ encompasses the eddy dispersion and the molecular diffusivity, which describes the diffusion velocity of molecules from the snow phase into the air phase.

Pressure gradient. The hydrostatic pressure gradient in the air-snow mixture is greater than a single-phase fluid. Consequently, the buoyancy force acting on a snow particle in suspension is larger. In order to capture the hydrostactic effects produced by the pressure gradient in the momentum equation (9), we use the total mixture pressure p that represents the sum of the dynamic and hydrostatic pressure [Rusche 2002]:

$$p = \hat{p} + \rho\mathbf{g} \cdot \mathbf{x}\,, \tag{11}$$

where the gradient of the hydrostatic pressure is given by:

$$\nabla\rho\mathbf{g} \cdot \mathbf{x} = \rho\mathbf{g} + (\mathbf{g} \cdot \mathbf{x})\nabla\rho\,.$$

Replacing the Equation (11) in Equation (9), we have:

Momentum equation revisited

$$\frac{\partial(\rho\hat{\mathbf{u}})}{\partial t} + \nabla \cdot (\rho\, \hat{\mathbf{u}} \otimes \hat{\mathbf{u}}) = -\nabla\hat{p} + \mu\Delta\hat{\mathbf{u}} - (\mathbf{g} \cdot \mathbf{x})\nabla\rho\,. \tag{12}$$

Poisson Pressure Equation (PPE). The momentum equation (12) couples the unknown fields $\hat{\mathbf{u}} = (\hat{u}, \hat{v}, \hat{w}) \in \mathbb{R}^3$ and $\hat{p} \in \mathbb{R}$, which poses an initial challenge to the solution of the problem. However, an additional pressure equation can be derived using the incompressibility constrain (8). For derivation purposes, the momentum equation (12) must be partially discretized. Thus, a semi-discretization of the linearized momentum equation, excluding and preserving the pressure term in its differential form, can be written as:

$$\mathbf{M}\hat{\mathbf{u}} - \mathbf{r} = -\nabla\hat{p}\,, \tag{13}$$

where $\mathbf{M}$ is the momentum matrix and $\mathbf{r}$ is a source term (including terms from the discretization of the time derivative). Defining the following operators

$$\mathbf{D} = \mathrm{diag}\,(\mathbf{M}) \quad \text{and} \quad \mathbf{H}(\hat{\mathbf{u}}) = \mathbf{r} - (\mathbf{M} - \mathbf{D})\hat{\mathbf{u}}\,, \tag{14}$$

where $\mathbf{D}$ is the matrix containing only the diagonal entries of $\mathbf{M}$. By Equations (13) and (14), we have:

$$\hat{\mathbf{u}} + \mathbf{D}^{-1}\nabla\hat{p} = \mathbf{D}^{-1}\mathbf{H}(\hat{\mathbf{u}})\,. \tag{15}$$

Finally, applying the divergence operator in both sides of the Equation (15) and by the constrain given by Equation (8):

Poisson pressure equation

$$\nabla \cdot [\mathbf{D}^{-1}\nabla\hat{p}] = \nabla \cdot [\mathbf{D}^{-1}\mathbf{H}(\hat{\mathbf{u}})]\,. \tag{16}$$

4.3 Transition Layer Model

The previous section detailed the model equations for momentum and mass transport. The set of equations describes the evolution of the powder-snow cloud over time due to gravity. The gravity force causes the acceleration of heavy snow particles in regions where the density difference is high. Besides the gravitational acceleration, mass transport manifests through diffusive convective flow between both phases. However, none of the terms in the equations consider the mass exchange between the DSL and PSL. Moreover, the volume of the powder-snow cloud also varies due to air drag and air intake, which are also not accounted for by the equations.

The TL manages mass addition in PSL from the snow entrainment processes in the interface between the DSL and the PSL. This section presents a novel procedural TL model that considers the snow entrainment as the PSL's primary source of mass and momentum injections (see Figure 7). Since the PSL equations describe the powder-snow mass in terms of its concentration $\alpha \in [0, 1]$, the TL model must define an injected mass α_{inj} driven by an vertical injection velocity $\mathbf{u}_{\text{inj}}$.

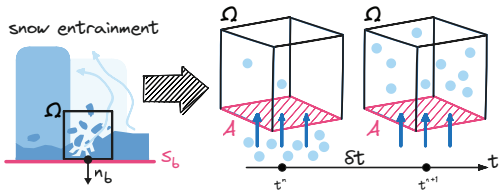


Fig. 7. The vertical injection of snow happens mainly through the entrainment processes. For a volume Ω in the PSL (black square/cube) which shares an area element A with the surface $\mathcal{S}_b$, the snow mass inflow through A increases the powder-snow (light blue dots) concentration inside Ω.

Snow mass injection. The formation of the powder snow cloud can be split into two processes: *air entrainment* and *ice-dust blow-out* [Bartelt et al. 2016]. Both processes occur simultaneously but at different locations of the avalanche. As heavy snow particles in the DSL collide with the ground, the dispersive pressure expands the dense avalanche core, which causes air entrainment (intake) and, consequently, the dilution of the air-snow mixture. Then, the downward motion of heavy snow particles displaces the enclosed air, causing the compression of the avalanche in DSL. The *vertical velocity* of expansion/compression is given by the material derivative

of the DSL height, i.e., $w_h = Dh/Dt$. From Equation (1), we can rewrite the vertical velocity in terms of the entrainment rate as $w_h = q_{er}/\bar{\rho}_s$. Let Ω be a volume control in the PSL boundary domain with $V = \text{volume}(\Omega)$, which shares an area element A with the basal surface $\mathcal{S}_b$. The entrained snow mass flux through an area element over the time-step δt is given by

$$m_s = \max\{0, w_h \delta_t \bar{\rho}_s A\} = \max\{0, q_{\text{er}} \delta_t A\} .$$

Finally, the injected snow mass $\alpha_{\text{inj}} \in [0, 1]$ is represented by the following mass fraction:

Injected snow mass

$$\alpha_{\text{inj}} = \omega_\alpha \frac{m_s}{m_V} \quad \text{with} \quad m_V = \hat{\rho}_s V . \tag{17}$$

The value m_V is the maximum snow mass capacity supported by the PSL volume control Ω. The scalar field $\omega_\alpha \in [0, 1]$ is a random density distribution representing snow cover heterogeneity.

Injection velocity. The injection velocity is parallel to $\mathbf{n}_b$ and is responsible for an increase in the momentum of the PSL by injecting kinetic energy into the system. The flow acceleration at the avalanche front has numerous sources, such as the displacement of air caused by the injection of snow mass from the ground (snow blow-out) and the air entrainment. These factors produce a violent particle suspension process that forms the snow cloud. The vertical injection of snow mass upwards is twice the vertical velocity w_h [Bartelt et al. 2016], while the flow acceleration caused by the air entrainment is proportional to the avalanche front velocity [Carroll et al. 2013]. Some models consider the front velocity to be the center of mass velocity of the avalanche [Turnbull et al. 2007], so a straightforward approximation of the front velocity is the DSL velocity $\bar{\mathbf{u}}$. In order to take into account both characteristics, we model the injection velocity at the basal surface $\mathcal{S}_b$ as

Injection velocity

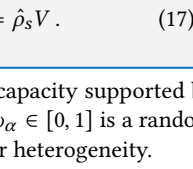
$$\mathbf{u}_{\text{inj}} = -\left(2\frac{q_{\text{er}}}{\bar{\rho}_s} + \omega_{\mathbf{u}} \|\bar{\mathbf{u}}\|_2\right) \mathbf{n}_b . \tag{18}$$

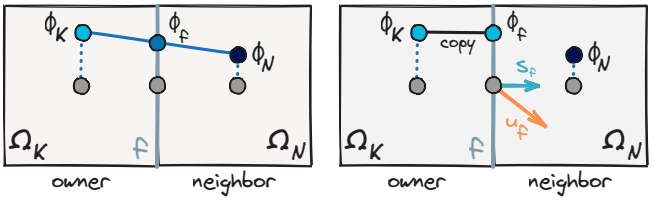
The scalar field $\omega_{\mathbf{u}} \in [0, 1]$ represents a random perturbation that mimics the air entrainment process and turbulent motion in the intermittency region [Ivanova et al. 2022].

In the next section, we will present the numerical tool for discretizing the momentum equation and the other governing equations used in our framework.

5 FVM APPROXIMATION

FVM transforms the PDEs representing the governing equations into discrete algebraic equations over a finite number of cells (i.e., control volumes). The first step in the approximation process is discretizing the spatial domain into a grid or an unstructured mesh. The mesh $\mathcal{M}$ is represented by a set of non-overlapping convex polyhedral cells, where the physical quantities are sampled at





(a) Linear scheme. (b) Upwind scheme.

Fig. 8. Approximation of ϕ_f at a face f using interpolation schemes.

the cell centers (centroids). Then, the PDEs are discretized into algebraic equations by integrating them over each cell $\Omega_K \in \mathcal{M}$. For an arbitrary scalar/vector field ϕ, applying the second-order accurate midpoint rule to approximate the integral of ϕ on Ω_K, we have

$$\int_{\Omega_K} \phi(\mathbf{x})\, d\Omega \approx \phi_K V_K \quad \text{with} \quad \phi_K = \phi(\mathbf{c}_K) ,$$

where $V_K = \text{volume}(\Omega_K)$ and $\mathbf{c}_K$ is the centroid of Ω_K.

Let $\mathcal{F}_K$ be the set of faces of a cell Ω_K, where each face $f \in \mathcal{F}_K$ has a *face area vector* $\mathbf{s}_f = S_f \mathbf{n}_f$, where S_f is the area of the face f and $\mathbf{n}_f$ is the unit outward pointing normal located at the face center $\mathbf{c}_f$. The differential operators are derived as boundary integrals over $\partial\Omega_K$ using the Gauss Theorem, and approximated as follows:

$$\nabla\phi_K \approx \frac{1}{V_K}\int_{\Omega_K} \nabla\phi\, d\Omega = \frac{1}{V_K}\oint_{\partial\Omega_K} \phi\, \mathbf{n}\, dS \approx \frac{1}{V_K}\sum_{f\in\mathcal{F}_K} \phi_f\, \mathbf{s}_f \tag{19}$$

$$\nabla\cdot\boldsymbol{\phi}_K \approx \frac{1}{V_K}\int_{\Omega_K} \nabla\cdot\boldsymbol{\phi}\, d\Omega = \frac{1}{V_K}\oint_{\partial\Omega_K} \boldsymbol{\phi}\cdot\mathbf{n}\, dS \approx \frac{1}{V_K}\sum_{f\in\mathcal{F}_K} \boldsymbol{\phi}_f\cdot\mathbf{s}_f \tag{20}$$

$$\Delta\phi_K \approx \frac{1}{V_K}\int_{\Omega_K} \Delta\phi\, d\Omega = \frac{1}{V_K}\oint_{\partial\Omega_K} \nabla\phi\cdot\mathbf{n}\, dS \approx \frac{1}{V_K}\sum_{f\in\mathcal{F}_K} \nabla\phi_f\cdot\mathbf{s}_f \tag{21}$$

where $\mathbf{n}$ is the outward normal and ϕ_f is the value of ϕ at face (center) f. The meaning of the Gauss Theorem is that the surface integral of an vector field through a closed surface, i.e., the *flux* through the surface, is equal to the volume integral of the divergence over the region inside the surface. This theorem is essential in the FVM since it allows to convert the volume integrals appearing in the governing equations into surface integrals.

The face values ϕ_f with $f \in \mathcal{F}_K$ can be achieved by using the *linear (central) differencing* scheme, which is based on a linear interpolation (Fig. 8a) between a reference cell, often called as *owner* cell, Ω_K and its face-adjacent *neighbor* cell Ω_N that share the face f:

$$\langle\phi_f\rangle_{\text{L}} = (1-\sigma_f)\phi_K + \sigma_f\phi_N \quad \text{with} \quad \sigma_f = \frac{|(\mathbf{c}_f - \mathbf{c}_K)\cdot\mathbf{n}_f|}{|\mathbf{d}_{NK}\cdot\mathbf{n}_f|} ,$$

where $\mathbf{d}_{NK} = \mathbf{c}_N - \mathbf{c}_K$. Despite the approximation $\langle\phi_f\rangle_{\text{L}}$ being second-order accurate, this linear scheme tend to generate unstable and oscillatory (*unbounded*) solutions for the convective term $\nabla\cdot\phi\mathbf{u}$. We can avoid this drawback using a bounded and less accurate first-order *upwind differencing* scheme (Figure 8b), which depends on the flow direction:

$$\langle\phi_f\rangle_{\text{U}} = \begin{cases} \phi_K , & \Psi_f \geq 0 \\ \phi_N , & \Psi_f < 0 \end{cases} ,$$

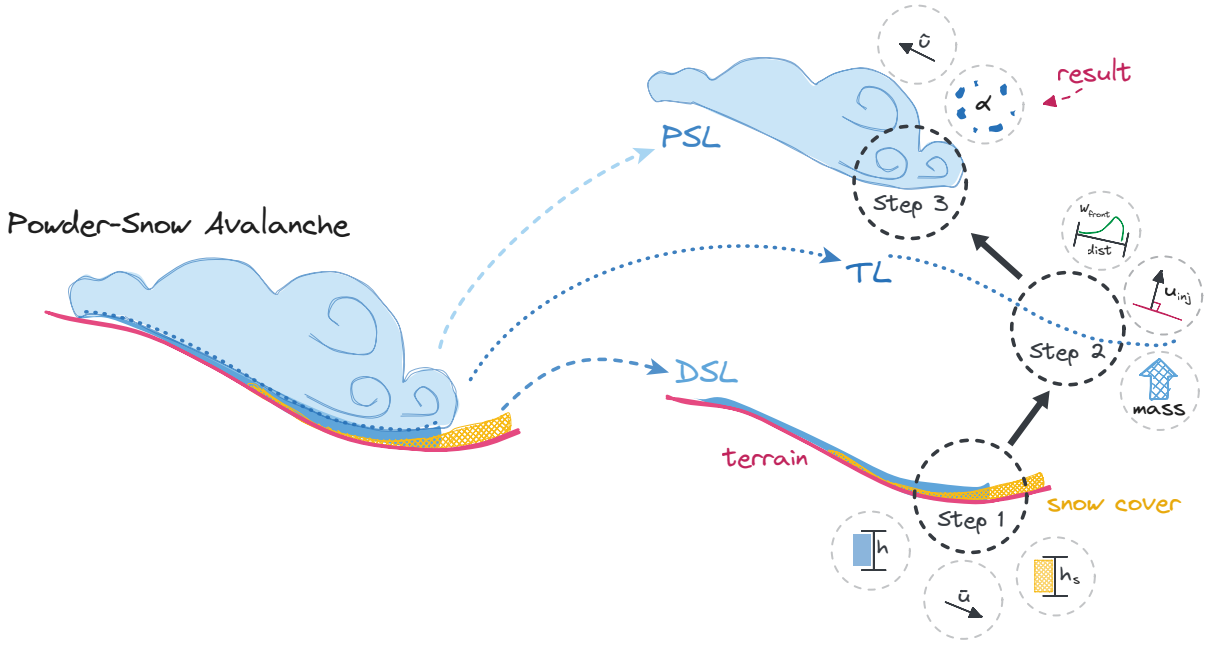


Fig. 9. Overview of the pipeline of our approach.

where $\Psi_f = \mathbf{u}_f \cdot \mathbf{s}_f$ is the face flux. A solution to overcome problems with stability and accuracy of the linear and upwind schemes, respectively, consists in combining both schemes as follows:

$$\langle \phi_f \rangle_\beta = [1 - \beta(r)]\langle \phi_f \rangle_\mathrm{U} + \beta(r)\langle \phi_f \rangle_\mathrm{L} \,.$$

The weight function $\beta(r)$ is the *flux limiter* defined by the ratio between cell and face gradients:

$$r = \max\left\{0, 2\frac{\mathbf{d}_{NK} \cdot \nabla\phi_K}{\|\mathbf{d}_{NK}\|_2 \nabla_\mathbf{n}\phi_f} - 1\right\} . \tag{22}$$

When the limiter detects high gradients or changes in slope, it switches the higher order approximation $\langle \phi_f \rangle_\mathrm{L}$ to low order $\langle \phi_f \rangle_\mathrm{U}$. Many practical approaches design the limiter β using a family of schemes known as *Total Variation Diminishing* (TVD) [Harten 1997].

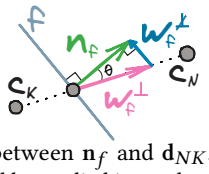
The surface normal gradient $\nabla_\mathbf{n}\phi = \nabla\phi \cdot \mathbf{n}$ evaluated at a face f that appears in Equation (22) can be discretized using centered finite difference scheme:

$$\langle \nabla_\mathbf{n}\phi_f \rangle_\mathrm{orth} = \frac{\phi_N - \phi_K}{\|\mathbf{d}_{NK}\|_2} \,.$$

Thus, the flux that appears in Laplacian approximation (21) can be approximated as

$$\nabla\phi_f \cdot \mathbf{s}_f = S_f \, \langle \nabla_\mathbf{n}\phi_f \rangle_\mathrm{orth} \,.$$

This discretization is second-order accurate when the vector $\mathbf{d}_{NK}$ is orthogonal to the face f, i.e., when $\mathbf{n}_f$ and $\mathbf{d}_{NK}$ are parallel. For non-orthogonal meshes, the error associated with the approximation $\langle \nabla_\mathbf{n}\phi_f \rangle_\mathrm{orth}$ increases proportionally with the angle θ between $\mathbf{n}_f$ and $\mathbf{d}_{NK}$. Therefore, a non-orthogonal correction should be applied in meshes



with significant non-orthogonality and skewness to improve the gradient accuracy [Greenshields and Weller 2022]:

$$\langle \nabla_\mathbf{n}\phi_f \rangle_\mathrm{corr} = \overbrace{\|\mathbf{w}_f^\perp\|_2 \langle \nabla_\mathbf{n}\phi_f \rangle_\mathrm{orth}}^{\text{orthogonal part}} + \overbrace{\langle \nabla\phi_f \rangle_\mathrm{L} \cdot \mathbf{w}_f^\mathcal{L}}^{\text{non-orthogonal part}} ,$$

where $\mathbf{w}_f^\perp = \mathbf{d}_{NK}/|\mathbf{d}_{NK} \cdot \mathbf{n}_f|$ and $\mathbf{w}_f^\mathcal{L} = \mathbf{n}_f - \mathbf{w}_f^\perp$. The correction is stable for angles $\theta \leq 75°$. Otherwise, in bad cells, the non-orthogonal part is *limited* by some fraction of the orthogonal part.

In our physically-based framework, we choose FVM due to its ability to discretize arbitrary geometries with guarantees of local conservation of quantities. For more details about the FVM, a comprehensive introduction can be found on [Moukalled et al. 2016].

6 METHODOLOGY

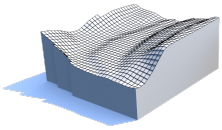
Our approach consists of three main steps that perform the numerical simulation of each avalanche layer based on FVM. The input values of thickness h_s, provided by the user, give the initial snow cover distribution in the ground layer. The simulations run over discrete meshes representing the terrain surface and the entire computational domain. Following, we will present the FVM discretization of each pipeline step (see Figure 9):

Step 1. Since it is a one-way coupling system, the simulation of the DSL flow can be executed without any dependency. The two essential resulting quantities are the height h and the DSL velocity $\bar{\mathbf{u}}$;

Step 2. The TL converts the resulting data from the DSL into boundary conditions for the PSL flow. The output is the amount of injected snow mass into the powder cloud and its injection velo-city $\mathbf{u}_\mathrm{inj}$;

Step 3. The final step simulates the snow cloud of the PSL. The resulting data is a volume fraction field α representing the powder-snow concentration in the computational domain.

6.1 DSL discretization



The terrain surface $\mathcal{S}_b$ is discretized by a polygonal mesh $\overline{\mathcal{M}}$, where each face $\Omega_K \in \overline{\mathcal{M}}$ has area V_K, surface normal $\mathbf{n}_K$, and is bounded by a set of straight edges $\mathcal{E}_K$. The endpoints of an edge $e \in \mathcal{E}_K$ determine an edge vector $\mathbf{e}$ with length $S_e = \|\mathbf{e}\|$, and an outward pointing normal of Ω_K given by the binormal $\mathbf{m}_e = \mathbf{n}_e \times \mathbf{e}/\|\mathbf{n}_e \times \mathbf{e}\|_2$, where $\mathbf{n}_e$ is the surface normal at edge e located at the edge midpoint $\mathbf{c}_e$. In our method, a quadrangular mesh can represent $\overline{\mathcal{M}}$ (see inset figure).

In order to solve the DSL equations (1)-(3), we use the specialization of the FVM for curved surfaces called *Finite Area Method* (FAM) [Rauter et al. 2018] (see Figure 10). The equations are solved sequentially in each time step by an iterative algorithm that repeats itself until its convergence is reached regarding a certain threshold. Firstly, the algorithm solves the Equation (3) to find a new value for the pressure $\bar{p}$. Then, it uses the Equation (2) to compute a new value for the velocity $\bar{\mathbf{u}}$ with updated values of $\bar{p}$ and old values of h. Finally, the algorithm uses new values of $\bar{\mathbf{u}}$ and $\bar{p}$ to update h with Equation (1). The temporal derivatives at time level $t^n = n\delta t$ are discretized using the (first-order) backward Euler scheme. We adopt the superscript $(*)^n$ for the unknown values at current time level, while the superscripts $(*)^{n-1}$ denote known values from previous time levels. The nonlinear terms are linearly approximated using an intermediary solution highlighted with $(*)^\star$, i.e., the intermediary solution is a prediction of the unknown value in the current time level. Therefore, for each face cell $\Omega_K \in \overline{\mathcal{M}}$, we have

FAM discretization of Eq. (3)

$$\begin{aligned}\bar{p}_K^n = {} & \bar{\rho}_s h_K^\star \mathbf{n}_K \cdot \mathbf{P}^\perp_{\mathbf{n}_K} \mathbf{g} \\ & - \frac{\bar{\rho}_s}{V_K} \mathbf{n}_K \cdot \mathbf{P}^\perp_{\mathbf{n}_K} \sum_{e \in \mathcal{E}_K} \langle h_e^\star \bar{\mathbf{u}}_e^\star \otimes \bar{\mathbf{u}}_e^\star \rangle_{\mathrm{U}} \, \mathbf{s}_e \\ & - \frac{1}{2V_K} \mathbf{n}_K \cdot \mathbf{P}^\perp_{\mathbf{n}_K} \sum_{e \in \mathcal{E}_K} \langle h_e^\star \bar{p}_e^\star \rangle_{\mathrm{L}} \, \mathbf{s}_e\end{aligned} \tag{23}$$

FAM discretization of Eq. (2)

$$\begin{aligned}& \frac{h_K^\star \bar{\mathbf{u}}_K^n - h_K^{n-1} \bar{\mathbf{u}}_K^{n-1}}{\delta t} V_K + \mathbf{P}^\parallel_{\mathbf{n}_K} \sum_{e \in \mathcal{E}_K} \langle h_e^\star \bar{\mathbf{u}}_e^\star \otimes \bar{\mathbf{u}}_e^n \rangle_{\mathrm{U}} \, \mathbf{s}_e \\ & = - \frac{\bar{\mathbf{u}}_K^n}{\|\bar{\mathbf{u}}_K^\star\|_2 + \varepsilon} \left[\frac{\upsilon \bar{p}_K^n}{\bar{\rho}_s} + \frac{g \|\bar{\mathbf{u}}_K^\star\|_2^2}{\xi} \right] V_K \\ & + h_K^\star \mathbf{P}^\parallel_{\mathbf{n}_K} \mathbf{g} V_K - \frac{1}{2\bar{\rho}_s} \mathbf{P}^\parallel_{\mathbf{n}_K} \sum_{e \in \mathcal{E}_K} \langle h_e^\star \bar{p}_e^n \rangle_{\mathrm{L}} \, \mathbf{s}_e\end{aligned} \tag{24}$$

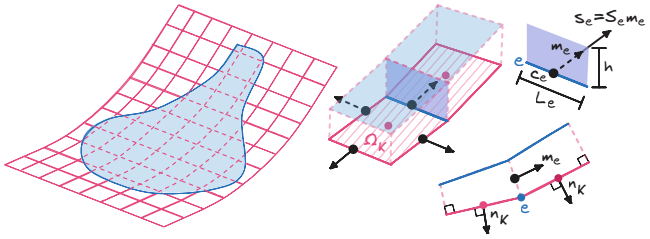


Fig. 10. FAM solves PDEs on discrete surface domains represented by a polygonal mesh $\overline{\mathcal{M}}$ (pink). The discretization process uses the fluxes passing through the edges of each face cell $\Omega_K \in \overline{\mathcal{M}}$. The convective flux in an edge e is defined by $\Phi_e = (\phi_e \bar{\mathbf{u}}_e) \cdot \mathbf{s}_e$, where $\mathbf{s}_e$ is the edge length vector pointing outward the cell Ω_K, and ϕ is an scalar field transported by the velocity $\bar{\mathbf{u}}$. Note that the binormal vector $\mathbf{m}_e$ is not necessarily orthogonal to the surface normal $\mathbf{n}_K$.

FAM discretization of Eq. (1)

$$\begin{aligned}& \frac{h_K^n - h_K^{n-1}}{\delta t} V_K + \sum_{e \in \mathcal{E}_K} \langle h_e^n \bar{\mathbf{u}}_e^n \rangle_{\mathrm{U}} \cdot \mathbf{s}_e \\ & = \frac{\|\bar{\mathbf{u}}_K^n\|_2}{E_b} \left[\frac{\upsilon \bar{p}_K^n}{\bar{\rho}_s} + \frac{g \|\bar{\mathbf{u}}_K^n\|_2^2}{\xi} \right] V_K\end{aligned} \tag{25}$$

We denote as $\mathbf{s}_e = S_e \mathbf{m}_e$ the *edge length vector* and ε is a small number to avoid zero division. To ensure that the velocities $\bar{\mathbf{u}}_e$ at the edges are surface-tangential, after the edge interpolation, we project $\bar{\mathbf{u}}_e$ back to the surface using the projection operator $\mathbf{P}^\parallel_{\mathbf{n}_e}$.

Since the edge interpolations are performed with cells that share the same edge, the implicit scheme used in Equations (24)-(25) results in the following sparse linear systems $\mathbf{A}\boldsymbol{\phi}^n = \mathbf{b}^\phi$ with $\phi = \bar{u}, \bar{v}, \bar{w}, h$, where the K-th row is given by:

$$\mathbf{A}^\phi_{KK} \phi_K^n + \sum_{N \in \mathcal{N}_K} \mathbf{A}^\phi_{KN} \phi_N^n = b_K^\phi \, .$$

$\mathcal{N}_K$ stands for the index list of the neighbor cells of Ω_K. Moreover, all terms not in time level t^n are grouped in b_K^ϕ. Implicit schemes are numerically more stable than explicit schemes for solving SWE (24)-(25), especially when dealing with large time steps, avoiding the stiffness issue raised from complex topographies and nonlinear interactions between convective and frictional terms.

Linear solver. The matrices $\mathbf{A}^\phi$ are non-symmetric due to the convective terms. For this reason, the systems are solved iteratively using the bi-conjugate gradient stabilized (BiCGStab) method preconditioned with the diagonal-based incomplete LU (DILU), and their convergence is checked by the residual $r^\phi = \|\mathbf{b}^\phi - \mathbf{A}^\phi \boldsymbol{\phi}^n\|_2$.

Boundary conditions. Since the basal pressure $\bar{p}$ is explicitly solved in Equation (23), the DSL model requires boundary conditions only for the velocity and height fields. For sake of simplicity, we impose homogeneous Neumann boundary conditions along the direction of the binormal $\mathbf{m}$ at the boundary surface $\partial \mathcal{S}_b$:

$$\nabla_{\mathbf{m}} \phi = 0 \quad \text{with} \quad \phi = \bar{u}, \bar{v}, \bar{w}, h \, .$$

ALGORITHM 1: DSL simulation

Input: time t^{end}, initial height h^0, threshold ϵ.

```
t = 0 ; h = h^0 ; p̄ = 0 ; ū = 0 ;
while t ≤ t^end do
    while residuals > ϵ do
        p̄ = update_DSL_pressure (ū , h , p̄) ; // Equation (23)
        ū= update_DSL_velocity (ū , h , p̄) ; // Equation (24)
        h= update_DSL_height (ū, h, p̄) ; // Equation (25)
    end
    t = t + δt ;
end
```

In FVM, homogeneous Neumann boundary condition specifies a *zero-gradient condition* for the flux at the boundary edges $e \in \partial\overline{\mathcal{M}}$. For the discretization of the convective flux, the edge values ϕ_e can be determined as follows:

$$\frac{\phi_e - \phi_K}{\|\mathbf{c}_e - \mathbf{c}_K\|_2} \approx \nabla_{\mathbf{m}}\phi = 0 \implies \phi_e = \phi_K\,,$$

where ϕ_K is the value in the cell Ω_K adjacent to the boundary edge e.

Algorithm 1 summarizes each step of the DSL simulation. In our experiments, we consider the body of snow cover at rest, i.e., the initial values for pressure and velocity are settled as zero. Moreover, an implementation of the DSL can be found on [Rauter et al. 2018].

6.2 TL discretization

The TL is implemented as inlet boundary conditions in the PSL. We impose Dirichlet boundary conditions for velocity and snow mass at the terrain mesh. The dynamical boundary values for the powder-snow concentration α_b and the PSL velocity field $\hat{\mathbf{u}}_b$ are defined for each terrain patch face f_b and updated on each time step. These values are mainly based on the DSL velocity field $\bar{\mathbf{u}}$ and the snow entrainment rate q_{er}. Therefore, the terrain patch geometry represents the interface between both layers since each face f_b will carry the state of the DSL for a particular time level t^n.

Avalanche trigger. In an avalanche, the suspension of snow particles occurs after the DSL achieves enough velocity and intensifies as the velocity increases [Bartelt et al. 2016]. From this observation, we model the intensification of the suspended particles induced by the entrainment processes by a sigmoid-like function of the velocity:

$$W_{\text{trigger}} = \frac{1}{1 + \exp\left(-\|\bar{\mathbf{u}}\|_2 + U_{\text{min}}\right)}\,,$$

where the value U_{min} controls the minimum velocity required for the powder cloud formation. In our experiments, we use $U_{\text{min}} = 10m/s$.

Avalanche front. The frontal region of the snow avalanche comprises the eruption entrainment process and is where the snow mass and momentum exchanges predominate in the TL. Consequently, these exchanges intensify in regions closer to the avalanche front [Sovilla et al. 2006]. Therefore, the injected mass α_{inj} and the injection velocity $\mathbf{u}_{\text{inj}}$ estimated at terrain position $\mathbf{x}_b$ depend on its distance to the avalanche front $\mathcal{A}_{\text{front}}$. We emulate such dependence

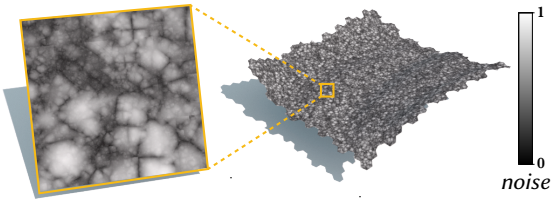


Fig. 11. Procedural noise functions can represent a complex snow cover distribution on a terrain surface. Such random variations in the injection velocity mimic the turbulent behavior of the PSA.

by a Gaussian weight function:

$$W_{\text{front}} = \exp\left(-\left[\frac{\text{dist}(\mathbf{x}_b, \mathcal{A}_{\text{front}})}{L_{\text{front}}}\right]^2\right). \tag{26}$$

The value L_{front} represents the *avalanche front size*, i.e., the first L_{front} meters from the leading edge of the avalanche. As the front advances, the distance field $\text{dist}(\mathbf{x}_b, \mathcal{A}_{\text{front}})$ is computed in every time-step by a *front-propagation algorithm* (see Section 7.1).

Dynamic boundary conditions. The TL boundary conditions are defined from Equations (17) and (18) and should include the trigger and frontal dynamics of the avalanche. Moreover, the inlet velocity must take into account the DSL motion. From these considerations, the TL is represented by enforcing the following constraints updated in each time-step for each face f_b:

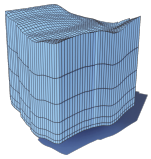
TL boundary conditions

$$\alpha_b = \gamma_\alpha W \alpha_{\text{inj}} \quad \text{and} \quad \hat{\mathbf{u}}_b = \gamma_{\mathbf{u}} W \mathbf{u}_{\text{inj}} + \bar{\mathbf{u}}\,, \tag{27}$$

where $W = W_{\text{trigger}} W_{\text{front}}$, the parameters $\gamma_\alpha \in [0, 1]$ and $\gamma_{\mathbf{u}} \in \mathbb{R}^+$ are scaling factors for mass and velocity injection, respectively.

Noise function. A practical solution to model the fields ω_α and $\omega_{\mathbf{u}}$ in Equations (17) and (18), respectively, is to use a procedural noise function that causes random perturbations in the injected snow mass and its injection velocity. In particular, we adopt the family of cellular noise functions, such as the *Worley noise* [Worley 1996], which produces distinct regions resembling natural-like patterns. Figure 11 shows the noise computed over the terrain.

6.3 PSL discretization



In this stage, the volumetric domain is discretized by a hexahedral mesh $\widehat{\mathcal{M}}$ constrained by the terrain mesh $\overline{\mathcal{M}}$. The 3D cells of $\widehat{\mathcal{M}}$ consist of axis-aligned boxes extruded from the cells of the terrain mesh $\overline{\mathcal{M}}$. In order to accurately capture the dynamics near the TL, we increase the mesh resolution near the terrain using a mesh grading in height-direction (see inset figure).

Before solving the Navier-Stokes Equations (8)-(9) to determine the PSL velocity and pressure fields sampled at the cell centers, we need to compute new values of the mixture density ρ and viscosity μ, given by Equation (7), in terms of the powder-snow concentration α obtained by solving the Equation (10).

The convective-diffusive equation (10) becomes increasingly complex in mixture flows due to phase interactions and quantity variations such as density and viscosity. The diffusive term tends to be stiff, requiring explicit solvers to take prohibitively small time steps to maintain stability. By treating the diffusive term implicitly, we enable significantly larger time steps without sacrificing stability. Meanwhile, handling the convective term explicitly prevents excessive computational costs while maintaining numerical robustness. This balance between efficiency and stability is the primary motivation for adopting an IMEX (implicit/explicit) solver. The IMEX Euler scheme splits the Equation (10) into two subproblems. The idea is to solve the first subproblem (convection term) using the forward Euler method and the second subproblem (diffusion term) using the backward Euler method. Thus, for each cell $\Omega_K \in \widehat{\mathcal{M}}$, we compute the values α_K^n as follows:

FVM discretization of Eq. (10)

$$\alpha_K^\star = \alpha_K^{n-1} - \frac{\delta t}{V_K} \sum_{f \in \mathcal{F}_K} \langle \alpha_f^{n-1} \hat{\mathbf{u}}_f^{n-1} \rangle_\beta \cdot \mathbf{s}_f \tag{28}$$

$$\begin{aligned} \frac{\alpha_K^n - \alpha_K^\star}{\delta t} V_K &= \Gamma \sum_{f \in \mathcal{F}_K} S_f \, \|\mathbf{w}_f^\perp\|_2 \langle \nabla_\mathbf{n} \alpha_f^n \rangle_{\text{orth}} \\ &+ \Gamma \sum_{f \in \mathcal{F}_K} S_f \, \langle \nabla \alpha_f^\star \rangle_\text{L} \cdot \mathbf{w}_f^\perp \end{aligned} \tag{29}$$

Equation (29) results in the linear system $\mathbf{A}^\alpha \boldsymbol{\alpha}^n = \mathbf{b}^\alpha$, where terms with predicted values $\alpha_K^\star$, including the non-orthogonal correction in surface normal gradient, are source terms and are appended in the vector $\mathbf{b}^\alpha$. In addition, we perform Equation (28) with TVD van Leer scheme $\beta(r) = r+|r|/1+|r|$ [van Leer 1974] in convective term to provide a second-order accurate and bounded solution $\alpha_K^\star \in [0, 1]$.

The next stage is computing the PSL velocity field from the momentum equation (12). We use the well-known predictor-corrector algorithm known as *Pressure Implicit with Splitting of Operators* (PISO) [Issa 1986], which comprises four main steps (see Figure 12):

Momentum prediction. In this step, an intermediary velocity $\hat{\mathbf{u}}^\star$ is computed from the new values of ρ^n and μ^n, and previous values of velocity $\hat{\mathbf{u}}^{n-1}$ and pressure $\hat{p}^{n-1}$. The intermediary solution is built from the FVM discretization of the momentum equation (12) using backward Euler scheme. Thus, for each cell $\Omega_K \in \widehat{\mathcal{M}}$, we have:

Momentum prediciton – FVM discretization of Eq. (12)

$$\begin{aligned} &\frac{\rho_K^n \hat{\mathbf{u}}_K^\star - \rho_K^{n-1} \hat{\mathbf{u}}_K^{n-1}}{\delta t} V_k + \sum_{f \in \mathcal{F}_K} \langle \rho_f^n \hat{\mathbf{u}}_f^\star \otimes \hat{\mathbf{u}}_f^{n-1} \rangle_\text{L} \, \mathbf{s}_f \\ &= - \sum_{f \in \mathcal{F}_K} \langle \hat{p}_f^{n-1} \rangle_\text{L} \, \mathbf{s}_f - (\mathbf{g} \cdot \mathbf{c}_K) \sum_{f \in \mathcal{F}_K} \langle \rho_f^n \rangle_\text{L} \, \mathbf{s}_f \\ &+ \mu_K \sum_{f \in \mathcal{F}_K} S_f \, \|\mathbf{w}_f^\perp\|_2 \langle \nabla_\mathbf{n} \hat{\mathbf{u}}_f^\star \rangle_{\text{orth}} \\ &+ \mu_K \sum_{f \in \mathcal{F}_K} S_f \, \langle \nabla \hat{\mathbf{u}}_f^{n-1} \rangle_\text{L} \cdot \mathbf{w}_f^\perp \end{aligned} \tag{30}$$

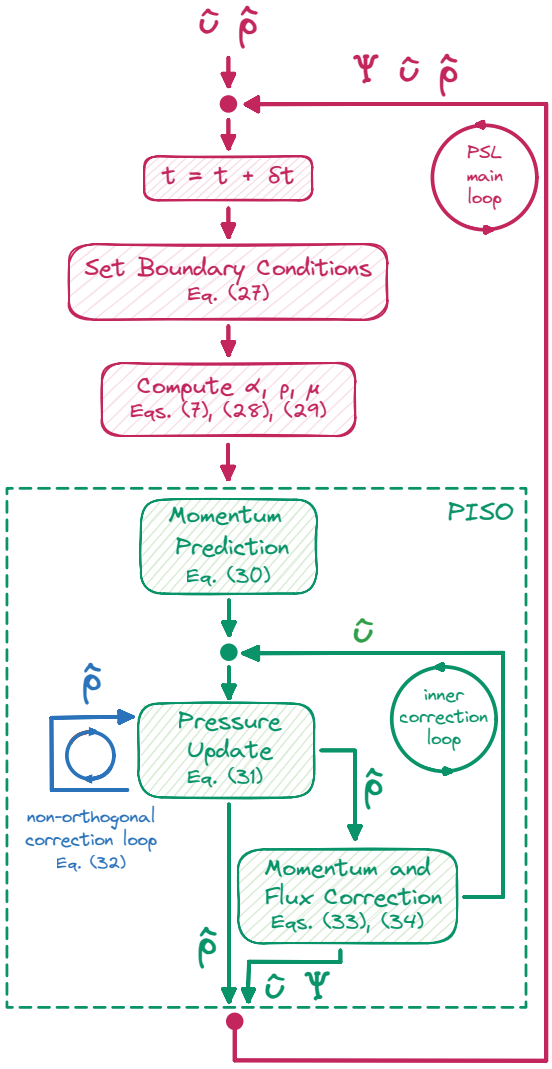


Fig. 12. PSL algorithm loop.

The intermediary solution $\hat{\mathbf{u}}^\star$ is obtained by solving the following linear system resulting from Equation (30):

$$\overbrace{\mathbf{M}_{KK} \hat{\mathbf{u}}_K^\star + \sum_{N \in \mathcal{N}_K} \mathbf{M}_{KN} \hat{\mathbf{u}}_N^\star}^{\mathbf{M}\hat{\mathbf{u}}^\star} = \overbrace{\mathbf{r}_K^{n-1} - \sum_{f \in \mathcal{F}_K} \langle \hat{p}_f^{n-1} \rangle_\text{L} \, \mathbf{s}_f}^{\mathbf{b}^{n-1}}$$

The linear system $\mathbf{M}\hat{\mathbf{u}}^\star = \mathbf{b}^{n-1}$ is solved once before the PISO loop.

Pressure update. The system (30) uses the previous pressure $\hat{p}^{n-1}$ instead of the coupled pressure field. A new pressure field $\hat{p}^\star$ is achieved from $\hat{\mathbf{u}}^\star$ using a FVM discretization of the pressure equation (16). Before to solve the PPE, we update the vector $\mathbf{H}(\hat{\mathbf{u}}^\star)$, as follows:

$$\mathbf{H}_K(\hat{\mathbf{u}}^\star) = \mathbf{r}_K^{n-1} - \sum_{N \in \mathcal{N}_K} \mathbf{M}_{KN} \hat{\mathbf{u}}_N^\star \, .$$

Considering the non-zero entries of the matrix diag(**M**) as a discrete scalar field $D_K = \mathbf{M}_{KK}$ defined at the cell centers. FVM approximation of the pressure equation for each cell Ω_K is given by:

FVM discretization of the PPE (16)

$$\sum_{f\in\mathcal{F}_K} S_f \langle D_f^{-1}\rangle_{\mathrm{L}} \langle \nabla_{\mathbf{n}} \hat{p}_f^{\star}\rangle_{\mathrm{ortho}} = \sum_{f\in\mathcal{F}_K} \langle D_f^{-1}\mathbf{H}_f(\hat{\mathbf{u}}^{\star})\rangle_{\mathrm{L}} \cdot \mathbf{s}_f \,. \quad (31)$$

The updated pressure field is obtained by solving the linear system $\mathbf{L}\hat{p}^{\star} = \mathbf{B}(\hat{\mathbf{u}}^{\star})$ provided by Equation (31), where the entries of the Laplacian matrix **L** are:

$$\mathbf{L}_{KN} = \begin{cases} \langle D_f^{-1}\rangle_{\mathrm{L}} \frac{S_f}{\|\mathbf{d}_{NK}\|_2}, & K \neq N \\ -\sum_N \mathbf{L}_{KN}, & K = N \end{cases},$$

with $\langle D_f^{-1}\rangle_{\mathrm{L}} = [(1-\lambda_f)\mathbf{M}_{KK} + \lambda_f \mathbf{M}_{NN}]^{-1}$.

Non-orthogonal correction. Considering the non-orthogonal contribution in surface normal gradients, we improve $\hat{p}^{\star}$ by running an additional correction loop. Essentially, we change the approximation $\langle \nabla_{\mathbf{n}} \hat{p}_f\rangle_{\mathrm{ortho}}$ by $\langle \nabla_{\mathbf{n}} \hat{p}_f\rangle_{\mathrm{corr}}$ in PPE (31). Besides, we assign the values $\hat{p}_f^{\star}$ in the non-orthogonal part as an explicit term, resulting in a modified PPE system, where each row is given by:

Non-orthogonal correction for PPE (31)

$$\mathbf{L}_K^{\perp}\hat{p}^n = \mathbf{B}_K(\hat{\mathbf{u}}^{\star}) - \sum_{f\in\mathcal{F}_K} S_f \langle D_f^{-1}\rangle_{\mathrm{L}} \langle \nabla \hat{p}_f^{\star}\rangle_{\mathrm{L}} \cdot \mathbf{w}_f^{\perp}, \quad (32)$$

$$\text{with} \quad \mathbf{L}_{KN}^{\perp} = \begin{cases} \langle D_f^{-1}\rangle_{\mathrm{L}} \frac{S_f}{|\mathbf{d}_{NK}\cdot\mathbf{n}_f|}, & K \neq N \\ -\sum_N \mathbf{L}_{KN}^{\perp}, & K = N \end{cases}.$$

Momentum and flux correction. The velocity field $\hat{\mathbf{u}}^{\star}$ is not solenoidal. We fix the field $\hat{\mathbf{u}}^{\star}$ using the updated pressure field $\hat{p}^n$. The corrected velocity field $\hat{\mathbf{u}}^n$ at the cell centers is obtained from the Equation (15) as follows:

Momentum correction – FVM discretization of Eq. (15)

$$\hat{\mathbf{u}}_K^n = \frac{1}{\mathbf{M}_{KK}} \left[\mathbf{H}_K(\hat{\mathbf{u}}^{\star}) - \sum_{f\in\mathcal{F}_K} \langle \hat{p}_f^n\rangle_{\mathrm{L}} \mathbf{s}_f \right]. \quad (33)$$

Furthermore, since $\hat{\mathbf{u}}$ is divergence-free, the fluxes $\Psi_f^n = \hat{\mathbf{u}}_f^n \cdot \mathbf{s}_f$ should satisfies the condition $\sum_f \Psi_f^n = 0$. Thus, the corrected fluxes are also obtained from the Equation (15):

$$\Psi_f^n = \langle D_f^{-1}\rangle_{\mathrm{L}} \left[\langle \mathbf{H}_f(\hat{\mathbf{u}}^{\star})\rangle_{\mathrm{L}} \cdot \mathbf{s}_f - S_f \langle \nabla_{\mathbf{n}} \hat{p}_f^n\rangle_{\mathrm{orth}} \right]. \quad (34)$$

Linear solver. The solutions of the sparse linear systems from the PSL discretization are obtained using iterative solvers. The symmetric positive definite linear systems (29), (31), and (32) are solved with conjugate gradient (CG) method preconditioned with incomplete Cholesky. In contrast, the non-symmetric system (30) is solved with BiCGStab preconditioned with DILU.

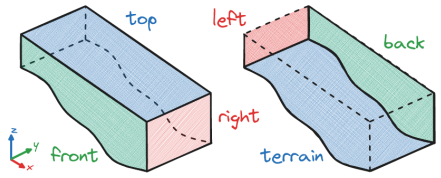


Fig. 13. Regardless the terrain topography, six patches comprise the boundary $\partial\widehat{\mathcal{M}}$. Considering the slope alignment with the axis x, each axis associates a pair of patches respectively: *right* and *left* in $\pm x$-directions, *back* and *front* in $\pm y$-directions, and *top* and *terrain* in $\pm z$-directions.

Boundary conditions. The boundary $\partial\widehat{\mathcal{M}}$ of the computational domain is decomposed into six patches for each direction of the Cartesian coordinates axis, as illustrated in Figure 13. Over the faces of each patch, we impose boundary conditions for the velocity field $\hat{\mathbf{u}}$, the pressure field $\hat{p}$, and the powder-snow concentration field α. We add some constraints, such as wall boundaries, to enforce the powder-snow cloud confinement. Thus, *front* and *back* patches are modeled as wall boundaries by imposing Dirichlet boundary condition (fixed value) for the concentration α and no-slip condition for the velocity $\hat{\mathbf{u}}$. Also, we allow inlet and outlet boundary conditions at *right* and *left* patches, respectively. In this case, we specify a fixed small velocity $\mathbf{u}_{\epsilon}$ for the velocity inlet/outlet to mimic the wind interaction with the avalanche. We model the *top* patch as an open (free) boundary condition that switches between Neumann and Dirichlet boundary conditions according to the inflow ($\hat{\mathbf{u}}\cdot\mathbf{n} < 0$) and outflow ($\hat{\mathbf{u}}\cdot\mathbf{n} \geq 0$):

$$\alpha_{\mathrm{io}} : \begin{cases} \alpha = 0, & \text{inflow} \\ \nabla_{\mathbf{n}}\alpha = 0, & \text{outflow} \end{cases} \text{ and } \hat{\mathbf{u}}_{\mathrm{io}} : \begin{cases} \hat{\mathbf{u}}^{\parallel} = \mathbf{0}, & \text{inflow} \\ \nabla_{\mathbf{n}}\hat{\mathbf{u}} = \mathbf{0}, & \text{outflow} \end{cases}, \quad (35)$$

where $\hat{\mathbf{u}}^{\parallel}$ is the tangential velocity. For the pressure field, we impose a zero-gradient condition for the PPE in all patches, except the *top* and *terrain* patches. To avoid spurious fluctuations in the pressure field at open boundaries, we use the stable *total pressure* boundary condition [Greenshields and Weller 2022] at the *top* patch; it is a Dirichlet boundary condition given by:

$$\hat{p}_{\mathrm{total}} = \begin{cases} -\frac{\|\hat{\mathbf{u}}\|_2^2}{2}, & \text{inflow} \\ 0, & \text{outflow} \end{cases} \quad (36)$$

Moreover, to ensure zero flux condition, we need to correct the flux at the *terrain*. It is achieved by compensating the gravitational force with the hydrostatic pressure gradient. Finally, the *terrain* patch receives the dynamic boundary conditions from Equation (27) representing the TL. Table 3 provides an overview of the set of boundary conditions for the PSL.

Table 3. PSL boundary conditions at the patches of $\partial\widehat{\mathcal{M}}$.

patch	concentration α	velocity $\hat{\mathbf{u}}$	pressure $\hat{p}$
left	$\nabla_{\mathbf{n}}\alpha = 0$	$\hat{\mathbf{u}} = \mathbf{u}_{\epsilon}$	$\nabla_{\mathbf{n}}\hat{p} = 0$
right	$\alpha = 0$	$\hat{\mathbf{u}} = \mathbf{u}_{\epsilon}$	$\nabla_{\mathbf{n}}\hat{p} = 0$
front	$\alpha = 0$	$\hat{\mathbf{u}} = \mathbf{0}$	$\nabla_{\mathbf{n}}\hat{p} = 0$
back	$\alpha = 0$	$\hat{\mathbf{u}} = \mathbf{0}$	$\nabla_{\mathbf{n}}\hat{p} = 0$
top	α_{io}	$\hat{\mathbf{u}}_{\mathrm{io}}$	$\hat{p} = \hat{p}_{\mathrm{total}}$
terrain	$\alpha = \alpha_b$	$\hat{\mathbf{u}} = \hat{\mathbf{u}}_b$	$\nabla_{\mathbf{n}}\hat{p} = -\nabla_{\mathbf{n}}(\rho\mathbf{g}\cdot\mathbf{x})$

ALGORITHM 2: PSL simulation

Input: time t^{end}, number of iterations n_{piso} and n_{corr}.

$t = 0$; $\hat{\mathbf{u}} = \mathbf{0}$; $\alpha = 0$; $\hat{p} = 0$;
while $t \leq t^{\text{end}}$ **do**
  set_boundary_conditions; // Table 3
  α = predict_volume_fraction ($\hat{\mathbf{u}}$, α) ; // Equation (28)
  α = correct_volume_fraction (α) ; // Equation (29)
  [ρ, μ] = compute_density_viscosity (α) ; // Equation (7)
  $\hat{\mathbf{u}}$ = predict_momentum (ρ, μ, $\hat{\mathbf{u}}$, $\hat{p}$) ; // Equation (30)
  for $j = 1$ **to** n_{piso} **do**
    $\hat{p}$ = solve_PPE ($\hat{\mathbf{u}}$) ; // Equation (31)
    for $k = 1$ **to** n_{corr} **do**
      $\hat{p}$ = correct_PPE ($\hat{\mathbf{u}}$, $\hat{p}$) ; // Equation (32)
    end
    $\hat{\mathbf{u}}$ = correct_velocity ($\hat{\mathbf{u}}$, $\hat{p}$) ; // Equation (33)
    Ψ = correct_flux ($\hat{\mathbf{u}}$, $\hat{p}$) ; // Equation (34)
  end
  $t = t + \delta t$;
end

The PISO and non-orthogonal correction loops (Figure 12) are controlled by the maximum number of iterations n_{piso} and n_{corr}, respectively. Algorithm 2 summarizes the PSL simulation.

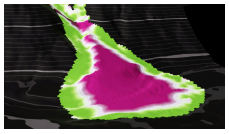
7 GEOMETRIC ALGORITHMS

In the following, we provide details for the geometric algorithms used in our framework, including the avalanche front distance and the mesh data conversion for rendering purposes.

7.1 Avalanche Front Distance

The TL depends on the distance from the avalanche front. More specifically, the surface distance dist($\mathbf{x}_b, \mathcal{A}_{\text{front}}$) used in Equation (26) approximates the shortest geodesic distance between a point $\mathbf{x}_b$ on the terrain and the front $\mathcal{A}_{\text{front}}$. In fact, the discrete distance field is computed in each cell Ω_K of the DSL mesh $\overline{\mathcal{M}}$, i.e., $dist_K$ = dist($\mathbf{c}_K, \mathcal{A}_{\text{front}}$). First, our algorithm detects the cells of $\overline{\mathcal{M}}$ that contain the avalanche front. These *front cells* provide an approximation of $\mathcal{A}_{\text{front}}$. Then, we propagate the discrete distance field *dist* from the avalanche front to the DSL cells, as illustrated by Figure 14.

Front cell detection. The DSL's height h and velocity $\bar{\mathbf{u}}$ specify the front cells. Nevertheless, we need to give some definitions before. A cell Ω_K is *empty* if its DSL height holds $h_K = 0$. A neighbor cell Ω_N of Ω_K is a *directional neighbor* with direction $\mathbf{d}$ if the ray cast from the centroid $\mathbf{c}_K$ with direction $\mathbf{d}$ intersects the shared edge. Therefore, a cell Ω_K is a front cell if its directional neighbor with direction $\bar{\mathbf{u}}_K$ is empty.



Distance propagation. The distance field is propagated from the front cells, setting all front cell distances to zero and putting them into a queue. While the queue is not empty, the next cell in the queue is taken, and any *affected* neighbor is pushed into the queue. A neighbor is affected if its distance value must be updated.

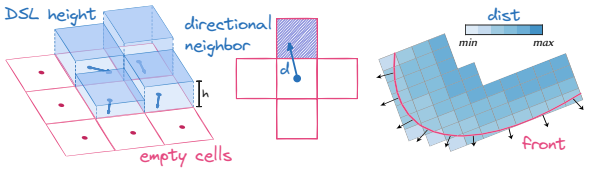


Fig. 14. The injection of powder-snow concentration and velocity depends on the distance from the avalanche front. The cells where the DSL height $h = 0$ are labeled as empty cells (*left*). A directional neighbor is a neighbor cell whose shared edge intersects the ray with direction $\mathbf{d}$ originating from the cell's centroid (*middle*). The distance field propagated from the front cells follows the opposite direction of the DSL velocity field (*right*).

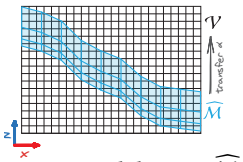
Each cell will hold the smallest distance propagated into it. The propagation follows the direction against the DSL flow. Given a cell Ω_K, only its non-empty directional neighbors Ω_N in the direction $-\bar{\mathbf{u}}_K$ are considered. The inset figure shows a narrow-band (green colors) in a hexagonal DSL mesh, with size L_{front}, around $\mathcal{A}_{\text{front}}$ defined by our front distance. Algorithm 3 provides a detailed pseudocode for computing the front distance in a time level t^n.

ALGORITHM 3: Avalanche front distance in the DSL

Input: mesh $\overline{\mathcal{M}}$, velocity $\bar{\mathbf{u}}^n$, height h^n.
Output: discrete distance field $dist^n$.

for *each cell* $\Omega_K \in \overline{\mathcal{M}}$ **do**
  $dist_K^n = \inf$;
end
$\mathcal{A}_{\text{front}}$ = detect_front_cells ($\overline{\mathcal{M}}, \bar{\mathbf{u}}^n, h^n$) ;
create a queue $\mathcal{Q}$;
for *each cell* $\Omega_K \in \mathcal{A}_{\text{front}}$ **do**
  $dist_K^n = 0$
  $\mathcal{Q}$.push (Ω_K) ;
end
while $\mathcal{Q} \neq \emptyset$ **do**
  $\Omega_K = \mathcal{Q}$.pop_front ( ) ;
  Ω_N = neighbor of Ω_K with direction $-\bar{\mathbf{u}}_K^n$;
  $\mathcal{D} = dist_K^n + \|\mathbf{c}_N - \mathbf{c}_K\|_2$;
  if $h_N^n \neq 0$ *and* $dist_N^n > \mathcal{D}$ **then**
    $dist_N^n = \mathcal{D}$;
    $\mathcal{Q}$.push (Ω_N) ;
  end
end
return $dist^n$;

7.2 Mesh Data Conversion



The PSL cloud can be represented as a volumetric density field, meaning the final result will be rendered as a volume of a scalar field $F : \mathbb{R}^3 \rightarrow \mathbb{R}$. Usually, the discrete field representation considers a regular fine grid $\mathcal{V}$ that encloses the computational domain $\widehat{\mathcal{M}}$, where a value of F_i is defined for each voxel $C_i \in \mathcal{V}$. The mesh data conversion from $\widehat{\mathcal{M}}$ into $\mathcal{V}$ comes from the approximation of the powder-snow concentration field α into density values for the

voxels. From the FVM, each cell $\Omega_K \in \widehat{\mathcal{M}}$ contains a single value of α_K, localized at the center of Ω_K, representing the mean value of the field over Ω_K. Similarly, each voxel C_i contains a single density value F_i placed in its center. However, the output result depends on the resolution of $\mathcal{V}$ and the accuracy of the approximation of the field α. For this reason, we use a higher-order meshless approximation provided by the *Hermite Radial Basis Function* (HRBF) interpolation [Fasshauer 2007].

Given a set of distinct cell centers $\{\mathbf{c}_I\}_{I=1}^N$, we want to find an interpolant $s^\alpha : \mathbb{R}^3 \to \mathbb{R}$ such that

$$s^\alpha(\mathbf{c}_I) = \alpha_I \quad \text{and} \quad \nabla s^\alpha(\mathbf{c}_I) = \nabla\alpha_I\,, \quad I = 1,\ldots,N\,. \tag{37}$$

The general form of an HRBF interpolant with polynomial augmentation is given by:

$$s^\alpha(\mathbf{x}) = \sum_{K=1}^{N} \boldsymbol{\Lambda}_K \cdot \begin{bmatrix} \varphi(\mathbf{x}-\mathbf{c}_K) \\ \nabla\varphi(\mathbf{x}-\mathbf{c}_K) \end{bmatrix} + \sum_{J=1}^{M} a_J\, P_J(\mathbf{x})\,, \tag{38}$$

where $\varphi : \mathbb{R}^3 \to \mathbb{R}$ is a radial function and $\{P_1(\mathbf{x}),\ldots,P_M(\mathbf{x})\}$ is a basis for the space Π_m^3 of all trivariate polynomials with degree less than or equal to m. Therefore, for determining s^α requires to discover the coefficients $\boldsymbol{\Lambda}_K \in \mathbb{R}^4$ and $a_J \in \mathbb{R}$.

To ensure uniqueness of the unknown coefficients $\boldsymbol{\Lambda}_K$ and a_J, we need to enforce additional constraints:

$$\sum_{K=1}^{N} \boldsymbol{\Lambda}_K \cdot \begin{bmatrix} P_J(\mathbf{c}_K) \\ \nabla P_J(\mathbf{c}_K) \end{bmatrix} = 0\,, \quad J = 1,\ldots,M = \tbinom{m+3}{3}\,. \tag{39}$$

The coefficients are determined by solving the following linear system of order $4N + M$ resulting from the constraints (37) and (39):

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{0}_M \end{bmatrix},$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{4N\times 4N}$, $\mathbf{P} \in \mathbb{R}^{4N\times M}$, $\mathbf{O}_M$ is a square zero matrix of order M, $\boldsymbol{\alpha} \in \mathbb{R}^{4N}$, and $\mathbf{0}_M$ is the null vector of length M. The entries of matrices $\boldsymbol{\Phi}$, $\mathbf{P}$, and the vector $\boldsymbol{\alpha}$ are given by the blocks:

$$\boldsymbol{\Phi}_{IK} = \begin{bmatrix} \varphi(\mathbf{c}_I-\mathbf{c}_K) & -\nabla\varphi(\mathbf{c}_I-\mathbf{c}_K)^\top \\ \nabla\varphi(\mathbf{c}_I-\mathbf{c}_K) & -H\varphi(\mathbf{c}_I-\mathbf{c}_K) \end{bmatrix} \in \mathbb{R}^{4\times 4}\,,$$

$$\mathbf{P}_{IJ} = \begin{bmatrix} P_J(\mathbf{c}_I) \\ \nabla P_J(\mathbf{c}_I) \end{bmatrix} \in \mathbb{R}^4 \quad \text{and} \quad \boldsymbol{\alpha}_I = \begin{bmatrix} \alpha_I \\ \nabla\alpha_I \end{bmatrix} \in \mathbb{R}^4\,.$$

The operator H is the Hessian operator. In our approach, we use polynomials of Π_1^3 and the cubic polyharmonic spline $\varphi(\mathbf{x}) = \|\mathbf{x}\|_2^3$, whose the derivatives are:

$$\nabla\varphi(\mathbf{x}) = 3\mathbf{x}\|\mathbf{x}\|_2 \quad \text{and}$$

$$H\varphi(\mathbf{x}) = \begin{cases} \frac{3}{\|\mathbf{x}\|_2}\left(\|\mathbf{x}\|_2^2\,\mathbf{I}_3 + \mathbf{x}\otimes\mathbf{x}\right), & \|\mathbf{x}\|_2 \neq 0 \\ \mathbf{O}_3\,, & \|\mathbf{x}\|_2 = 0 \end{cases}.$$

For a large number of cell centers, the global HRBF interpolation (38) produces large and ill-conditioned matrices that also become computationally intractable due to the high memory footprint. To avoid these drawbacks, we break the global interpolation into several local interpolations and then glue them using a *partition of unity* (PU) [Fasshauer 2007]. First, we define a coarse regular grid $\mathcal{G}$ from a low-resolution version of $\mathcal{V}$. Then, the coarse voxels are grouped into $2\times 2\times 2$ overlapping regions $\{R_\ell\}_{\ell=1}^L$ such that

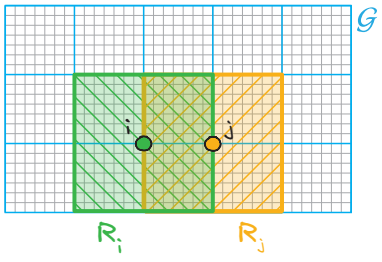


Fig. 15. The configuration of the PU on a coarse grid $\mathcal{G}$ (blue) generated from the rendering grid $\mathcal{V}$ (gray) in 2D. The PU regions R_i are formed by 2^{dim} voxels centered on *internal* grid nodes $\mathbf{i}$ of $\mathcal{G}$. We compute HRBF interpolations s_i^α in each region R_i. Then, the local interpolations s_i^α are smoothly blended in the overlapped regions using PU weights to provide an HRBF-PU approximation F^α. In particular, the regions R_i (green) and R_j (yellow) overlap in two voxels of $\mathcal{G}$.

$\widehat{\mathcal{M}} \subset \bigcup_{\ell=1}^L R_\ell$ (see Figure 15). For each non-empty region R_ℓ that encloses a subset of centers, we compute the local HRBF interpolation s_ℓ^α. The global approximation of scalar field α is composed by joining s_ℓ^α via PU weights functions $\{w_\ell\}_{\ell=1}^n$:

$$F^\alpha(\mathbf{x}) = \sum_{\ell=1}^{n} w_\ell(\mathbf{x})\, s_\ell^\alpha(\mathbf{x})\,. \tag{40}$$

The PU weights w_ℓ are nonnegative functions regarding the covering R_ℓ, which satisfy $\sum_{\ell=1}^n w_\ell(\mathbf{x}) = 1$, $\forall\mathbf{x} \in \widehat{\mathcal{M}}$. We define the PU weights as follows:

$$w_\ell(\mathbf{x}) = \frac{\chi_\ell(\mathbf{x})}{\sum_{j=1}^n \chi_j(\mathbf{x})}\,,$$

where χ_ℓ is the characteristic function:

$$\chi_\ell(\mathbf{x}) = \begin{cases} 1\,, & \mathbf{x} \in R_\ell \\ 0\,, & \mathbf{x} \notin R_\ell \end{cases}.$$

Finally, we evaluate the HRBF-PU approximation (40) in the centroids of the voxels $C_i \in \mathcal{V}$ to achieve the discrete density field for the PSL cloud. In our implementation, the voxels are stored using the sparse grid data structure provided by OpenVDB [Museth 2013]. Figure 16 compares our HRBF-PU against other meshless methods to demonstrate the interpolation accuracy. We approximate a discrete scalar field on a fine grid $\mathcal{V} \in [-5,5]^3$ with resolution 100^3 by splatting a Gaussian function $f(\mathbf{x}) = \exp(-0.5\|\mathbf{x}\|_2^2)$ sampled in a coarse version of $\mathcal{V}$ with resolution 20^3 . We can notice that our HRBF-PU preserves the smoothness of the field and achieves higher accuracy regarding root-mean-square error (*RMSE*).

8 RESULTS

We implemented our framework in C++ using the FVM software package OpenFOAM [Greenshields 2024]. Regarding time integration, we use adaptive time-steps managed by the CFL condition. All experiments were performed on a computer equipped with a processor Intel i9-13900k with 24 cores of 3.0GHz and 64GB RAM. The simulations ran in parallel on CPU (up to 20 threads). Unless otherwise specified, the simulations presented in this section use the set of parameter values listed in Table 4.

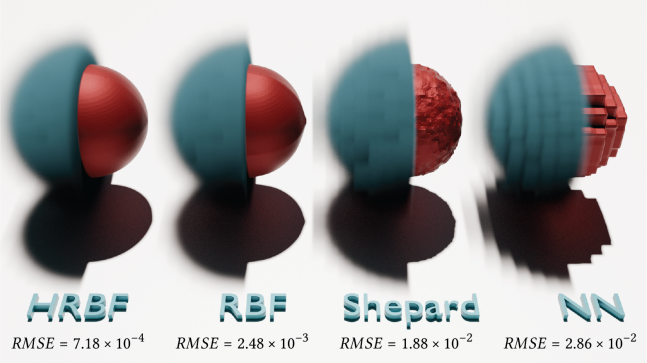


Fig. 16. Interpolating a sampled Gaussian from a coarse to fine grid using our HRBF-PU, RBF-PU, Shepard, and nearest neighbor (NN) interpolation (*from left to right*). As can be seen, the volumetric field (blue) and its isosurface (red) provided by HRBF-PU is more resilient to the aliasing effect.

Table 4. Model parameters values.

layer	parameter	description	value
all	g	standard gravity $[m/s^2]$	9.81
DSL	$\bar{\rho}_s$	snow density $[kg/m^3]$	500
	υ	Voellmy's dry friction	0.155
	ξ	Voellmy's dynamic friction $[m/s^2]$	5000
	E_b	specific erosion energy $[m^2/s^2]$	50
TL	γ_α	mass injection factor	0.1
	$\gamma_\mathbf{u}$	velocity injection factor	1.6
	L_{front}	avalanche front size $[m]$	80
PSL	$\hat{\rho}_s$	powder-snow density $[kg/m^3]$	1.4
	$\hat{\rho}_a$	air density $[kg/m^3]$	1.2
	$\hat{\mu}_s$	powder-snow viscosity $[m^2/s]$	10^{-4}
	$\hat{\mu}_a$	air viscosity $[m^2/s]$	1.4×10^{-5}
	Γ	diffusion coefficient $[m^2/s]$	2.0×10^4

Table 5 shows the computational times and some statistics for a set of experiments presented in the paper. The first column **simulation** contains the performed experiment (*scene*) using our approach, the dimension (*dim*) of the domain, and the duration (*dur.*) of the resulting animation in seconds. The column **#cells** provides the number of cells of the DSL and PSL meshes. The column $\overline{\delta x}$ presents the average cell sizes in meters. The column **parameters** shows the input values used in the Algorithm 2 and also the maximum Courant number $Co_{\max}$ employed in the CFL condition. The last column **time** presents the computational times for DSL and PSL (with TL) times in the entire simulation and the average time dt for a single time-step. As can be seen, in 3D simulations, the DSL computational time is less than 1% of the overall simulation time.

Procedural and natural mountains. The inherent complexity of procedural and natural terrains shapes the path taken by an avalanche. Figure 1 shows the ability of our method to simulate large-scale PSAs in a synthetic mountain from a height map generated with a procedural noise. Besides, we have real-world topographical data, usually represented by a Digital Elevation Model (DEM). Figure 17 shows our PSA in a natural mountain at Tyrolean Alps (St. Anton,

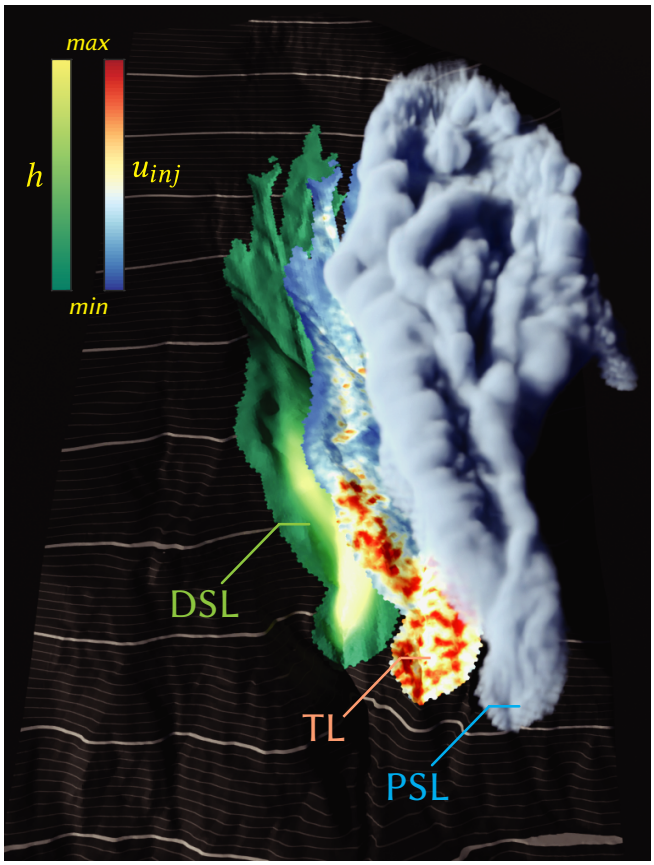


Fig. 17. A layered digital PSA along a natural mountain. The topographic lines represent the vertical height of the terrain with a bold line every 100 m. The colormaps indicate the DSL height field h and the injection velocity magnitude u_{inj} controlled by the TL. Notice that the high mass injection occurs in the avalanche front due to the snow entrainment.

Austria), using a PSL mesh generated from a DEM file with 6.9M cells occupying a total volume of 0.67 km^3. The mountain topography drives the DSL flow through its channel, leaving a clear runout area. Note how the PSL cloud triggered by the DSL follows the terrain features by being channeled in the middle of the track, reaching a runout distance of 2.2 km. In this experiment, we replaced the DSL parameters with $\hat{\rho}_s = 200$, $\upsilon = 0.26$, $\xi = 8650$, and $E_b = 11500$.

Terrain topography. The geometry of the terrain shapes the path taken by an avalanche. The DSL follows the terrain topography, tunneling into fissures, bifurcating over sharp protuberances, and spreading across open areas. The DSL flow's behavior over the terrain's complexities directly impacts the appearance of the PSL cloud. Such an implication happens due to the influence of the velocity and entrainment of snow, which depend on the DSL and the snow cover. Figure 18 shows the effect of the terrain topography through a comparison of a PSA descending a modified ramp mimicking riverbed-like terrain followed by bumps against a flat ramp with the same 25°slope and initial conditions.

Table 5. Timings (in seconds), statistics and parameters of our method.

simulation			#cells		$\overline{\delta x}$		parameters			time			
scene	*dim*	*dur.*	DSL	PSL	DSL	PSL	n_{piso}	n_{corr}	$Co_{\max}$	*dt*	DSL	PSL	*total*
Fig. 1	3	42	31K	17.8M	7.75	2.99	2	2	0.2	237.17	74	588224	588298
Fig. 17	3	78	41K	6.9M	8.65	4.61	2	2	0.2	61.89	317	345454	345771
Fig. 18.(a)	3	35	54K	2.7M	2.77	2.00	2	2	0.2	14.59	62	59175	59237
Fig. 18.(b)	3	35	54K	2.7M	2.10	1.99	2	2	0.2	14.51	63	54367	54430
Fig. 19.(a)	2	30	24K	86K	3.15	1.00	2	2	0.2	0.82	141	3429	3570
Fig. 19.(b)	2	30	23K	82K	3.22	1.03	2	2	0.2	0.83	135	3671	3806
Fig. 19.(c)	2	30	22K	78K	3.32	1.08	2	2	0.2	0.83	133	4086	4219
Fig. 20.(a)	3	90	34K	2.4M	30.56	3.30	2	2	0.2	11.24	102	75549	75651
Fig. 20.(b)	3	90	34K	2.4M	30.56	3.30	1	1	1.0	4.20	102	15800	15902
Fig. 21.(a)	3	100	99K	11.4M	37.97	2.47	1	1	0.7	27.23	535	113568	114103
Fig. 21.(b)	3	100	99K	11.4M	37.97	2.47	1	1	0.7	28.33	535	114588	115123

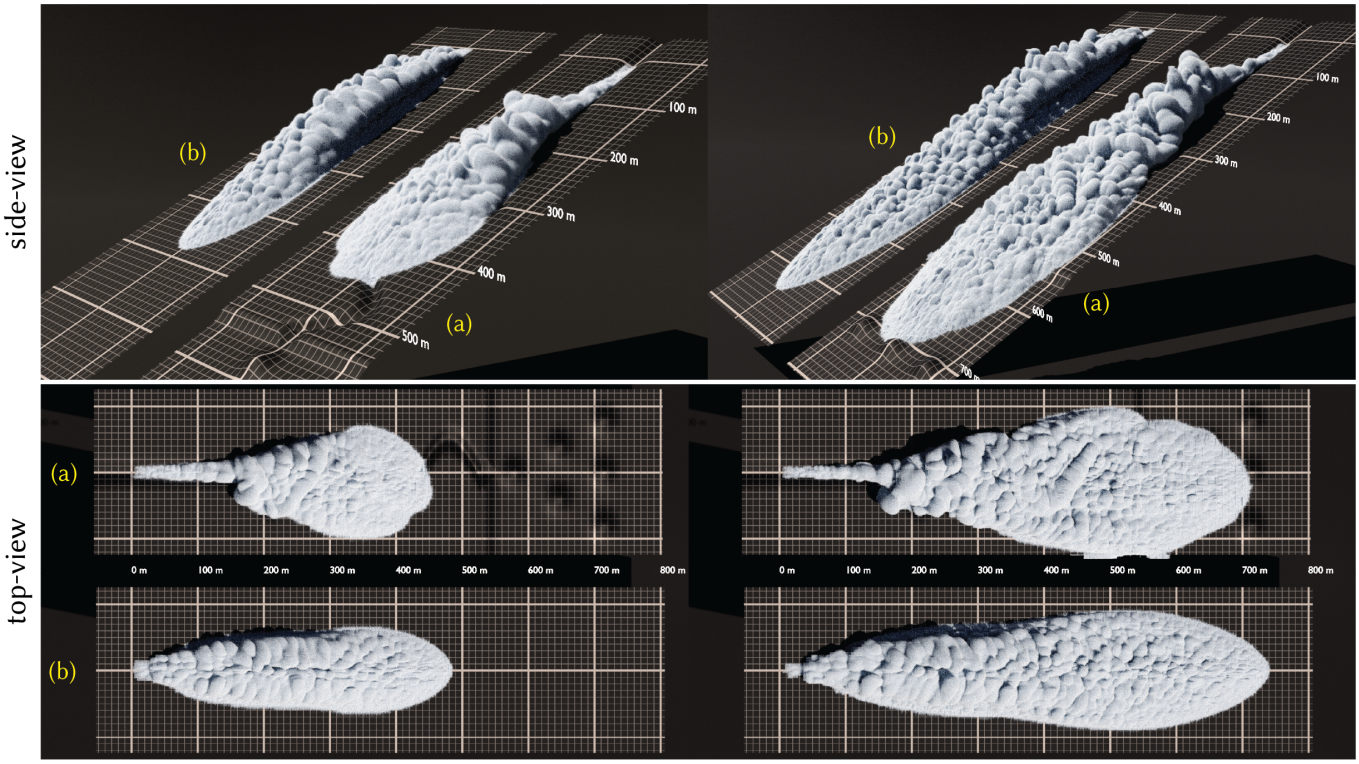


Fig. 18. Influence of the terrain topography in a PSA descending a modified bumped ramp (a) and the equivalent avalanche in a flat ramp (b). Note how the features of the underlying terrain geometry manifest in the PSL. The avalanche in (a) is channeled near the release zone, while the surface bumps break its symmetry.

Slope influence. PSAs may achieve different velocities, heights, and volume profiles of powder clouds due to the influence of the slope. The different velocities result directly from the action of gravity. The greater the angle, the bigger the tangential acceleration. Figure 19 shows the resulting PSA clouds in flat ramps with slopes varying from 25° to 35°. As can be seen, the density difference in the mixture phases gives rise to Kelvin-Helmholtz instabilities and air entrainment in the turbulent interface between ambient air and the suspended powder-snow. The graphs show higher angles produce greater velocities, leading to longer runout distances. Since the velocity directly impacts the PSL's entrainment, the slope affects the height and volume reached by the powder cloud in suspension. In our experiment, we used the following TL parameters $\gamma_\alpha = 2$ and $\gamma_{\mathbf{u}} = 1.4$ in this experiment.

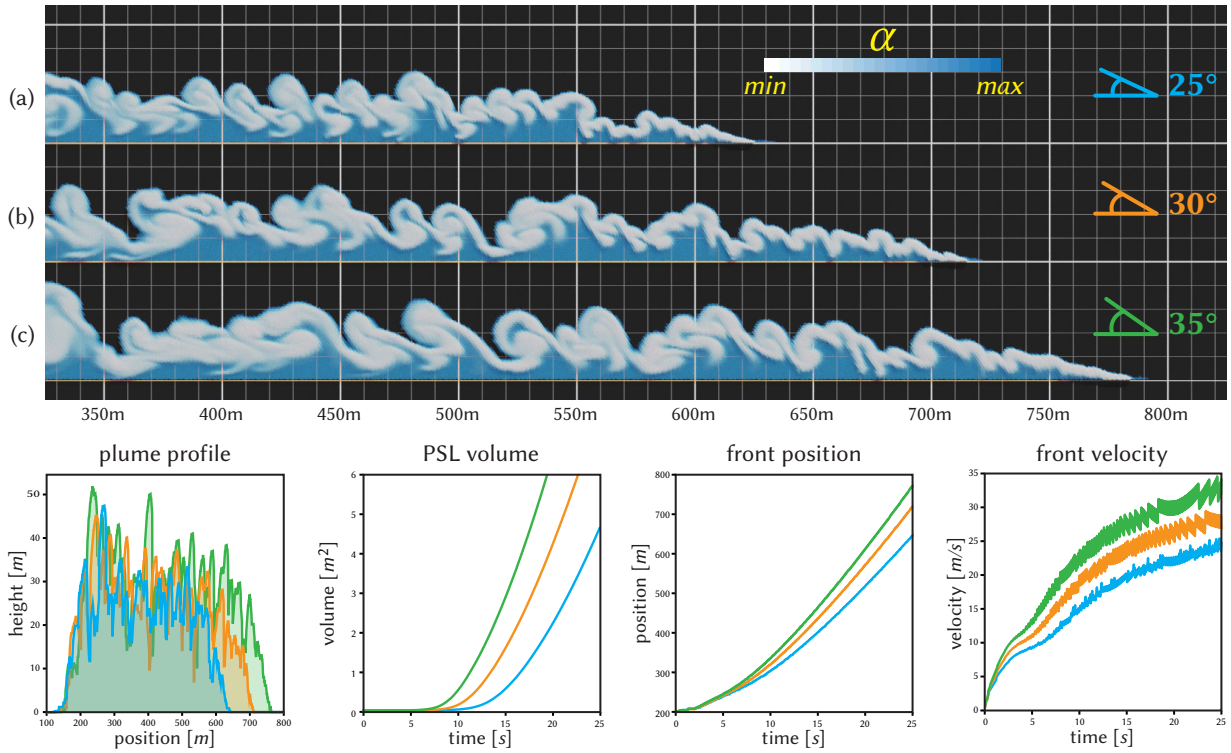


Fig. 19. On the top, resulting powder-snow clouds 25 seconds after release for different slopes. The colors encode the powder-snow concentration field α; the bluish colors represent higher α values. At the bottom, the graphs from left to right depict the profile of the plume heights, the powder-snow cloud volume, the distance covered by the avalanche, and the evolution of its front velocity for 25°(■), 30°(■), and 35°(■) slopes.

Performance profiling and speed-up. Figure 20 presents the performance profiling of our PSL algorithm (as shown in Figure 12) performed in a simulation of an avalanche descending a curved ramp discretized by a PSL mesh with 2.4M cells. This simulation bottleneck is to solve large linear systems resulting from the pressure equations (31)-(32). This computational overwhelming is due to the inner correction loops in the PISO algorithm, where the pressure solver is called $n_{\text{piso}}(n_{\text{corr}} + 1)$ times for each time-step by the Algorithm 2. Considering the number of sub-cycles as $n_{\text{piso}} = n_{\text{corr}} = sc \in \mathbb{N}$, an optimized solution (with $sc = 1$) is 2.7× faster than an accurate solution (with $sc = 2$) for a single time-step. Still, on the issue of the trade-off between accuracy and computational performance, we can also relax the CFL condition allowing large time-steps by increasing the Co_{max}. Thus, increasing Co_{max} from 0.2 to 1.0, we obtain a speed-up of 4.8× in the overall simulation time (see Table 5) without sacrificing the visual quality of the result. The simulation bottleneck is to solve a large linear system resulting from PPEs (31)-(32), taking 65% of the computational time in the optimized solution. On the other hand, the stages before the PISO loop (including the TL computations) spent only 15% of the computational time.

Boundary conditions comparison. Figure 21 shows a PSA simulation at Mount Niobe (Tantalus Range, Canada) from a DEM file provided by [U.S. Geological Survey 2024]. We compare the simulations along a runout distance of 1.5 km using our method with mixed boundary conditions given by Table 3 and open boundaries, where snow mass can freely flow inwards and outwards. We model the open boundary condition by imposing the constraints given by Equations (35)-(36) in the patches surrounding the *terrain* patch. As can be seen, the mixed boundary condition benefits plume formation, while the open condition spreads the powder-snow cloud further along the avalanche track.

Ablation and mesh independence studies. In the Supplemental Material, we provide studies about the influence of the parameters and interpolation schemes on the simulation, and its convergence under mesh refinements.

9 DISCUSSION AND LIMITATIONS

In this section, we discuss some characteristics of our method, highlighting its limitations and potential improvements to the current state of the framework.

Validation. Ground-truth data for PSAs is scarce in Geoscience, as few studies provide direct measurements. Instead, most rely on photogrammetric analysis to derive key avalanche characteristics, such as plume height and powder cloud volume profiles [Bartelt et al. 2013]. Figure 19 shows that our framework successfully reproduces the height profiles (including the blow-out heights) and the parabolic shape of the PSL volume profiles that closely align with those reported in [Bartelt et al. 2013], which validates our method's

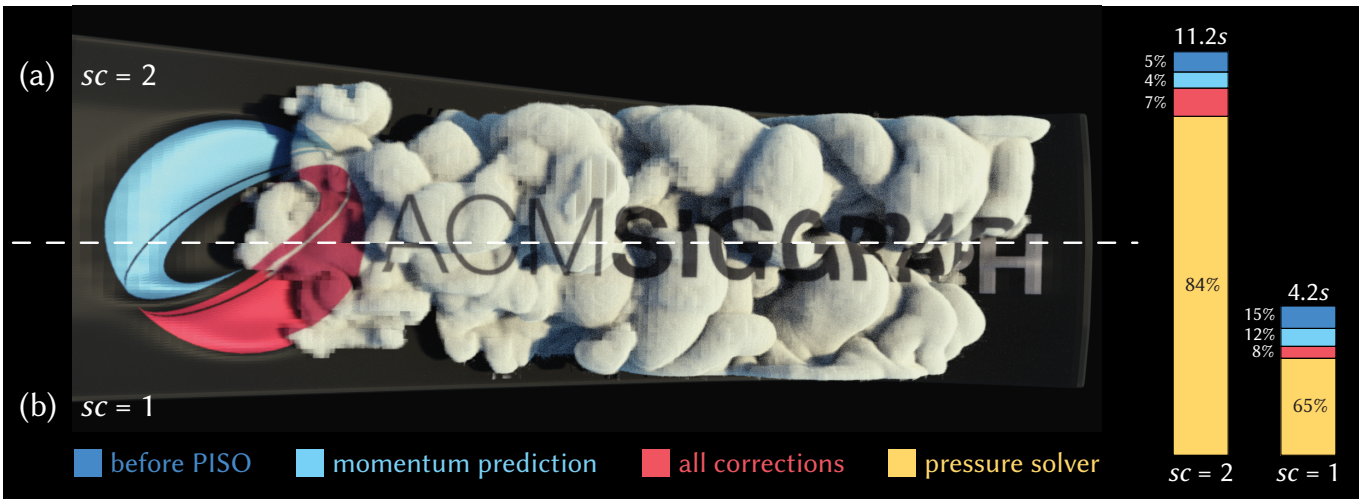
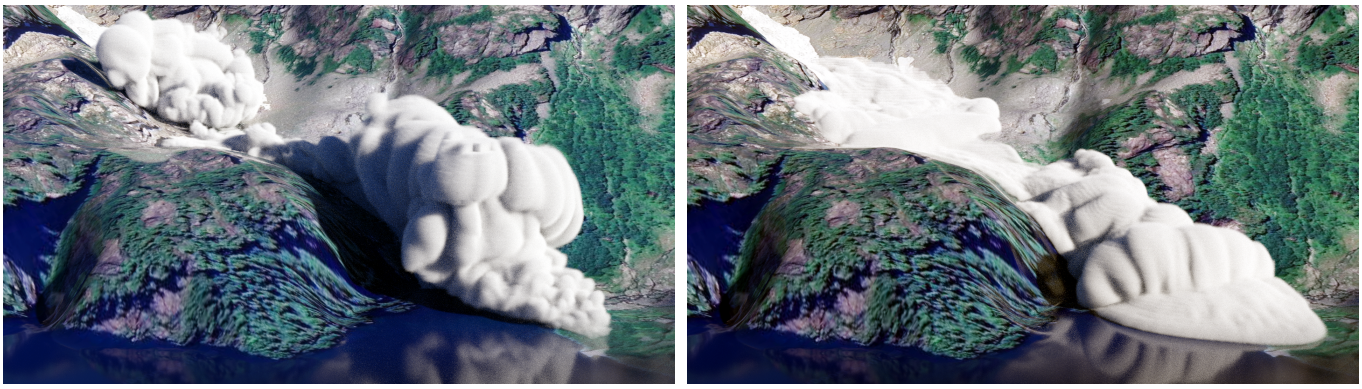


Fig. 20. A half-and-half comparison of a PSA slips down a curved ramp with the starting zone placed at the SIGGRAPH logo and elevated letters in the avalanche track using an accurate (a) and an optimized (b) solution. On the right is a breakdown of the averaged computational time of a single time-step of our PSL solver for each setup.



(a) Mixed boundary condition.

(b) Open boundary condition.

Fig. 21. A powder-snow avalanche simulation at Mount Niobe (Tantalus Range, Canada) using a PSL mesh with 11.4M cells occupying a total volume of $0.17\,km^3$ with different boundary conditions.

ability to capture the PSA dynamics. In addition, regarding a qualitative comparison, our method can reproduce the plume structure formation at the avalanche front as illustrated in the photo of a real avalanche (Figure 2), visually validating our results.

Snow cover initialization. The initial mountain snow cover thickness h_s determines the entrainment rate (5) in the DSL. It is usually estimated through empirical rules built upon direct observations of the snow accumulation measured on the basal surface $\mathcal{S}_b$ at a reference altitude z_0 along a snowfall. We compute the initial snow distribution using the linear model proposed by [Fischer et al. 2015]:

$$h_s(z) = \left[h_s(z_0) + (z - z_0)\frac{\partial h_s}{\partial z} \right] \cos\theta\,,$$

where θ is the angle between the gravitational acceleration and the surface normal (i.e., $\cos\theta = \mathbf{g} \cdot \mathbf{n}_b$), note that the h_s reaches high values on flat slopes and low values on steep slopes. The growth rate $\partial h_s/\partial z$ is an empirical value based on precipitation attributes. In our framework, we use the following parameters $h_s(z_0) = 1.61m$ and $\partial h_s/\partial z = 8 \times 10^{-4}$, as suggested by [Rauter et al. 2018].

Compressibility. The velocity field $\hat{\mathbf{u}}$ of a mixture of incompressible fluids is not divergence-free when the densities of the two fluids are distinct [Joseph 2009]. The diffusion equation (10) is based on Fick's law, which results in the following quasi-compressible constrain [Dutykh et al. 2011; Gurjar 2023]:

$$\nabla \cdot \hat{\mathbf{v}} = 0 \quad \text{with} \quad \hat{\mathbf{v}} = \hat{\mathbf{u}} + \Gamma \nabla \log \rho\,.$$

A solution is rewrite the momentum equation (12) in terms of the *fluid volume velocity* $\hat{\mathbf{v}}$. However, this quasi-compressible formulation increase computational effort due to the appearance of extra terms in the momentum equation due to variable substitution.

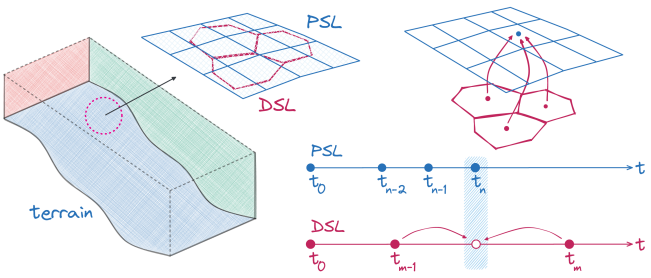


Fig. 22. The DSL and PSL simulations can share different mesh tessellation or time steps. The TL requires the transference of information between both layers. This figure shows a DSL simulated in a hexagonal mesh registered at time steps t_m. The terrain patch of the PSL is a quadrangular mesh and follows different time steps t_n. When simulating the PSL, the DSL state is required at time $t_n \in [t_{m-1}, t_m]$; therefore, DSL values are interpolated temporally and spatially between cell centers.

One-way coupling. Our framework transfers only the snow concentration and momentum from the DSL to PSL, i.e., the PSL simulation does not affect the DSL simulation. Beyond the computational efficiency, the advantage of one-way coupling simulation is that the PSL simulation is agnostic of the method utilized for simulating the DSL, which means that other DSL models serving the same type of information can replace the presented DSL model. In practical terms, the DSL simulation can provide a preview of the avalanche motion for a VFX designer. However, the cells of the DSL mesh and the PSL terrain patch may not share the same geometry or connectivity. Furthermore, since the DSL simulation is independent of the PSL simulation, it may use a different value for the time steps. In order to synchronize the simulations, we interpolate the DSL quantities temporally and spatially between cell centers at the same PSL time level (see Figure 22).

Limitations. The current DSL method is limited to mildly curved terrains due to the nature of SWE. Such a model requires small flow heights compared to the curvature. Besides, our method does not deal with discontinuous terrains (e.g., avalanche over a cliff). However, other hybrid alternatives exist, such as the recent use of the MPM method for dense snow flows [Li et al. 2022]. We believe this class of particle-mesh methods can bring solutions for both the DSL and the PSL simulations.

10 CONCLUSION AND FUTURE WORK

We have introduced a computational framework for simulating powder-snow avalanches under complex terrains. Our physically-based approach relies on a multi-layer model that splits the avalanche into two main layers: dense and powder-snow. The dense-snow layer flow is simulated by solving the Savage-Hutter equations, while the powder-snow layer flow is modeled as a miscible air-snow mixture and simulated using the incompressible Navier-Stokes equations. We also present a novel procedural TL modeled as boundary conditions for the snow mass and momentum exchange between the simulation layers. Moreover, we attest with set experiments that our approach provides visually realistic simulations of the turbulent snow cloud rising from an avalanche sliding down a basal surface.

Future work. The resolution of the PSL simulation is the main factor for the visual quality of the result. However, the simulation domains are inherently too large to utilize desirable cell sizes. Although our current simulation framework can handle parallel runs efficiently on the CPU, we did not profoundly explore performance optimization in our implementation. A natural and appealing path for future developments is using GPUs to solve numerical systems or handle mesh element computations such as interpolation of fields. Indeed, the leading edge of the PSL is one of its mesmerizing and distinct features. However, the front's appearance depends on the resolution of the numerical grid. In this sense, a topic for further investigation is to explore adaptive grids [Nakanishi et al. 2020] to develop solutions that provide outstanding details on the cloud interface, mainly in the front. The adaptivity should provide dynamic refinement close to the evolving interface between air and powder snow. Also, turbulence is the core phenomenon in such avalanches. Although the noise factors in the entrainment model agitate the flow and lead to plume billow shapes, the method does not handle internal flows and fluctuations caused by the intermittent region. The next step would be to resort to numerical methods for turbulent flows such as large-eddy simulation (LES) and direct numerical simulation (DNS) or exploring procedural turbulence models [Kim et al. 2008]. Finally, physical interaction is of utmost importance in animating physical phenomena. Future developments should consider solid-fluid interactions [Ng et al. 2009] between avalanches and solid obstacles, such as trees and buildings.

ACKNOWLEDGMENTS

We want to thank Geovan Tavares for presenting the importance of this subject years ago, and the SIGGRAPH reviewers for their comments. This study was financed in part by National Council for Scientific and Technological Development (CNPq, Brazil) fellowships #310319/2023-4 and #312372/2023, and São Paulo Research Foundation (FAPESP) under grants #2018/06145-4 and #2019/23215-9. The Center for Mathematical Sciences Applied to Industry (CeMEAI) provided the computational resources, also funded by FAPESP (grant #2013/07375).

REFERENCES

C. Ancey. 2001. *Geomorphological Fluid Mechanics.* Springer, Chapter Snow Avalanches, 319–338.

P. Bartelt, Y. Bühler, O. Buser, and C. Ginzler. 2013. Plume Formation in Powder Snow Avalanches. In *International Snow Science Workshop.*

P. Bartelt, O. Buser, C. Vera Valero, and Y. Bühler. 2016. Configurational energy and the formation of mixed flowing/powder snow and ice avalanches. *Ann. Glaciol.* 57, 71 (2016), 179–188. https://doi.org/10.3189/2016aog71a464

C. Calgaro, E. Creusé, and T. Goudon. 2015. Modeling and simulation of mixture flows: Application to powder–snow avalanches. *Comput. Fluids* 107 (Jan. 2015), 100–122. https://doi.org/10.1016/j.compfluid.2014.10.008

C. S. Carroll, M. Y. Louge, and B. Turnbull. 2013. Frontal dynamics of powder snow avalanches. *J. Geophys. Res. Earth Surf.* 118, 2 (2013), 913–924. https://doi.org/10.1002/jgrf.20068

A. Chourasia, S. Cutchin, Y. Cui, R. W. Moore, K. Olsen, S. M. Day, J. B. Minster, P. Maechling, and T. H. Jordan. 2007. Visual Insights into High-Resolution Earthquake Simulations. *IEEE Comput. Graph. Appl.* 27, 5 (2007), 28–34. https://doi.org/10.1109/mcg.2007.138

M. Christen, J. Kowalski, and P. Bartelt. 2010. RAMMS: Numerical simulation of dense snow avalanches in three-dimensional terrain. *Cold Reg. Sci. Technol.* 63, 1-2 (2010), 1–14. https://doi.org/10.1016/j.coldregions.2010.04.005

G. Cordonnier, P. Ecormier, E. Galin, J. Gain, B. Benes, and M.-P. Cani. 2018. Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches. *Comput. Graph. Forum* 37, 2 (2018), 497–509. https://doi.org/10.1111/cgf.13379

D. Dutykh, C. Acary-Robert, and D. Bresch. 2011. Mathematical Modeling of Powder-Snow Avalanche Flows. *Stud. Appl. Math.* 127, 1 (2011), 38–66. https://doi.org/10.1111/j.1467-9590.2010.00511.x

J. Etienne, M. Rastello, and E. J. Hopfinger. 2006. Modelling and simulation of powder-snow avalanches. *Comptes Rendus Mécanique* 334, 8-9 (2006), 545–554. https://doi.org/10.1016/j.crme.2006.07.010

J. Etienne, P. Saramito, and E. J. Hopfinger. 2004. Numerical simulations of dense clouds on steep slopes: application to powder-snow avalanches. *Ann. Glaciol.* 38 (2004), 379–383. https://doi.org/10.3189/172756404781815031

I. Failes. 2021. Behind 'Black Widow': A helicopter, a prison, an avalanche and a digital pig. https://www.wetafx.co.nz/articles/behind-black-widow-a-helicopter-a-prison-an-avalanche-and-a-digital-pig/

G. F. Fasshauer. 2007. *Meshfree Approximation Methods with MATLAB.* World Scientific Publishing.

J.-T. Fischer, A. Kofler, W. Fellin, M. Granig, and K. Kleemayr. 2015. Multivariate parameter optimization for computational snow avalanche simulation. *J. Glaciol.* 61, 229 (2015), 875–888. https://doi.org/10.3189/2015jog14j168

C. Gissler, A. Henne, S. Band, A. Peer, and M. Teschner. 2020. An Implicit Compressible SPH Solver for Snow Simulation. *ACM Trans. Graph.* 39, 4 (2020). https://doi.org/10.1145/3386569.3392431

C. Greenshields. 2024. *OpenFOAM v12 User Guide.* The OpenFOAM Foundation. https://doc.cfd.direct/openfoam/user-guide-v12

C. Greenshields and H. Weller. 2022. *Notes on Computational Fluid Dynamics: General Principles.* CFD Direct Ltd.

L. Guillet, L. Blatny, B. Trottet, D. Steffen, and J. Gaume. 2023. A Depth-Averaged Material Point Method for Shallow Landslides: Applications to Snow Slab Avalanche Release. *J. Geophys. Res. Earth Surf.* 128, 8 (2023). https://doi.org/10.1029/2023jf007092

S. Gurjar. 2023. *Numerical Simulation of a Powder Snow Avalanche.* PhD Thesis. ETH Zurich. https://doi.org/10.3929/ethz-b-000600974

T. Hädrich, M. Makowski, W. Pałubicki, D. T. Banuti, S. Pirk, and D. L. Michels. 2020. Stormscapes: simulating cloud dynamics in the now. *ACM Trans. Graph.* 39, 6 (2020), 1–16. https://doi.org/10.1145/3414685.3417801

A. Harten. 1997. High Resolution Schemes for Hyperbolic Conservation Laws. *J. Comput. Phys.* 135, 2 (1997), 260–278. https://doi.org/10.1006/jcph.1997.5713

J. A. Amador Herrera, J. Klein, D. Liu, W. Pałubicki, S. Pirk, and D. L. Michels. 2024. Cyclogenesis: Simulating Hurricanes and Tornadoes. *ACM Trans. Graph.* 43, 4 (2024). https://doi.org/10.1145/3658149

Imageworks. 2020. Mulan. https://www.imageworks.com/our-craft/vfx/movies/mulan

R.I Issa. 1986. Solution of the implicitly discretised fluid flow equations by operator-splitting. *J. Comput. Phys.* 62, 1 (1986), 40–65. https://doi.org/10.1016/0021-9991(86)90099-9

D. Issler. 1998. Modelling of snow entrainment and deposition in powder-snow avalanches. *Ann. Glaciol.* 26 (1998), 253–258. https://doi.org/10.3189/1998aog26-1-253-258

D. Issler. 2014. Dynamically consistent entrainment laws for depth-averaged avalanche models. *J. Fluid Mech.* 759 (2014), 701–738. https://doi.org/10.1017/jfm.2014.584

K. Ivanova, A. Caviezel, Y. Bühler, and P. Bartelt. 2022. Numerical modelling of turbulent geophysical flows using a hyperbolic shear shallow water model: Application to powder snow avalanches. *Comput. Fluids* 233 (2022), 105211. https://doi.org/10.1016/j.compfluid.2021.105211

D. D. Joseph. 2009. *Fluid Dynamics of Mixtures of Incompressible Miscible Liquids.* Springer, 127–145. https://doi.org/10.1007/978-90-481-3239-3_10

A. Kapler. 2003. Avalanche! Snowy FX for XXX. In *ACM SIGGRAPH 2003 Sketches & Applications (SIGGRAPH '03).* 1. https://doi.org/10.1145/965400.965492

T. Kim, N. Thürey, D. James, and M. Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Trans. Graph.* 27, 3 (2008), 1–6. https://doi.org/10.1145/1360612.1360649

T-Y. Kim and L. Flores. 2008. Snow Avalanche Effects for Mummy 3. In *ACM SIGGRAPH 2008 Talks (SIGGRAPH '08).* Article 7, 1 pages. https://doi.org/10.1145/1401032.1401041

J. Künnap. 1985. *Laviin MAL-Pamir 85 02.* Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Laviin_MAL-Pamir_85_02.jpg (photo licensed under CC BY-SA 4.0).

X. Li, M. Li, X. Han, H. Wang, Y. Yang, and C. Jiang. 2024. A Dynamic Duo of Finite Elements and Material Points. In *SIGGRAPH '24.* https://doi.org/10.1145/3641519.3657449

X. Li, B. Sovilla, C. Jiang, and J. Gaume. 2021. Three-dimensional and real-scale modeling of flow regimes in dense snow avalanches. *Landslides* 18, 10 (2021), 3393–3406. https://doi.org/10.1007/s10346-021-01692-8

X. Li, B. Sovilla, C. Ligneau, C. Jiang, and J. Gaume. 2022. Different erosion and entrainment mechanisms in snow avalanches. *Mech. Res. Commun.* 124 (2022), 103914. https://doi.org/10.1016/j.mechrescom.2022.103914

X. Liu, Y. Chen, H. Zhang, Y. Zou, Z. Wang, and Q. Peng. 2021. Physically based modeling and rendering of avalanches. *Vis. Comput.* 37, 9-11 (2021), 2619–2629. https://doi.org/10.1007/s00371-021-02215-1

M. Mergili, J.-T. Fischer, J. Krenn, and S. P. Pudasaini. 2017. r.avaflow v1, an advanced open-source computational framework for the propagation and interaction of two-phase mass flows. *Geosci. Model Dev.* 10, 2 (2017), 553–569. https://doi.org/10.5194/gmd-10-553-2017

F. Moukalled, L. Mangani, and M. Darwish. 2016. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab.* Springer. https://doi.org/10.1007/978-3-319-16874-6

K. Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3 (2013), 1–22. https://doi.org/10.1145/2487228.2487235

R. Nakanishi, F. Nascimento, R. Campos, P. Pagliosa, and A. Paiva. 2020. RBF liquids: an adaptive PIC solver using RBF-FD. *ACM Trans. Graph.* 39, 6 (2020), 1–13. https://doi.org/10.1145/3414685.3417794

Y. T. Ng, C. Min, and F. Gibou. 2009. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829. https://doi.org/10.1016/j.jcp.2009.08.032

P. C. Pretorius, J. Gain, M. Lastic, G. Cordonnier, J. Chen, D. Rohmer, and M.-P. Cani. 2024. Volcanic Skies: coupling explosive eruptions with atmospheric simulation to create consistent skyscapes. *Comput. Graph. Forum* 43, 2 (2024). https://doi.org/10.1111/cgf.15034

M. Rauter, A. Kofler, A. Huber, and W. Fellin. 2018. faSavageHutterFOAM 1.0: depth-integrated simulation of dense snow avalanches on natural terrain with OpenFOAM. *Geosci. Model Dev.* 11, 7 (2018), 2923–2939. https://doi.org/10.5194/gmd-11-2923-2018

M. Rauter and Ž. Tuković. 2018. A finite area scheme for shallow granular flows on three-dimensional surfaces. *Comput. Fluids* 166 (2018), 184–199. https://doi.org/10.1016/j.compfluid.2018.02.017

H. Rusche. 2002. *Computational fluid dynamics of dispersed two-phase flows at high phase fractions.* PhD Thesis. Imperial College London. http://hdl.handle.net/10044/1/8110

P. Sampl and M. Granig. 2009. Avalanche simulation with SAMOS-AT. In *International Snow Science Workshop.* 519–523.

S. B. Savage and K. Hutter. 1991. The dynamics of avalanches of granular materials from initiation to runout. Part I: Analysis. *Acta Mech.* 86, 1-4 (1991), 201–223. https://doi.org/10.1007/bf01175958

J. E. Simpson. 1999. *Gravity Currents: In the environment and the laboratory.* Cambridge University Press.

B. Sovilla, P. Burlando, and P. Bartelt. 2006. Field experiments and numerical modeling of mass entrainment in snow avalanches. *J. Geophys. Res. Earth Surf.* 111, F3 (2006). https://doi.org/10.1029/2005jf000391

B. Sovilla, J. N. McElwaine, and A. Köhler. 2018. The Intermittency Regions of Powder Snow Avalanches. *J. Geophys. Res. Earth Surf.* 123, 10 (2018), 2525–2545. https://doi.org/10.1029/2018jf004678

B. Sovilla, J. N. McElwaine, and M. Y. Louge. 2015. The structure of powder snow avalanches. *C. R. Phys.* 16, 1 (2015), 97–104. https://doi.org/10.1016/j.crhy.2014.11.005

J. Stam. 1999. Stable fluids. In *SIGGRAPH '99.* https://doi.org/10.1145/311535.311548

A. Stomakhin, . Schroeder, L. Chai, J. Teran, and . Selle. 2013. A material point method for snow simulation. *ACM Trans. Graph.* 32, 4 (2013), 1–10. https://doi.org/10.1145/2461912.2461948

Y. Tsuda, Y. Yue, Y. Dobashi, and T. Nishita. 2010. Visual simulation of mixed-motion avalanches with interactions between snow layers. *Vis. Comput.* 26, 6-8 (2010), 883–891. https://doi.org/10.1007/s00371-010-0491-5

B. Turnbull and P. Bartelt. 2003. Mass and Momentum Balance Model of a Mixed Flowing/Powder Snow Avalanche. *Surv. Geophys.* 24, 5/6 (2003), 465–477. https://doi.org/10.1023/b:geop.0000006077.82404.84

B. Turnbull, J. N. McElwaine, and C. Ancey. 2007. Kulikovskiy–Sveshnikova–Beghin model of powder snow avalanches: Development and application. *J. Geophys. Res. Earth Surf.* 112, F1 (2007). https://doi.org/10.1029/2006jf000489

U.S. Geological Survey. 2024. 1/3rd arc-second Digital Elevation Models (DEMs) - USGS National Map 3DEP Downloadable Data Collection. https://www.usgs.gov

B. van Leer. 1974. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *J. Comput. Phys.* 14, 4 (1974), 361–370. https://doi.org/10.1016/0021-9991(74)90019-9

A. Voellmy. 1955. Über die Zerstörungskraft von Lawinen. 73 (1955), 212–217. https://doi.org/10.5169/SEALS-61891

Y. Wang, K. Hutter, and S.P. Pudasaini. 2004. The Savage-Hutter theory: A system of partial differential equations for avalanche flows of snow, debris, and mud. *Z. Angew. Math. Mech.* 84, 8 (2004), 507–527. https://doi.org/10.1002/zamm.200310123

S. Worley. 1996. A cellular texture basis function. In *SIGGRAPH '96.* https://doi.org/10.1145/237170.237267

K. Wu, N. Truong, C. Yuksel, and R. Hoetzlein. 2018. Fast Fluid Simulations with Sparse Volumes on the GPU. *Comput. Graph. Forum* 37, 2 (2018), 157–167. https://doi.org/10.1111/cgf.13350

Y. Xu, X. Wang, J. Wang, C. Song, T. Wang, Y. Zhang, J. Chang, J. J. Zhang, J. Kosinka, A. Telea, and X. Ban. 2023. An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations. In *SIGGRAPH Asia 2023.* https://doi.org/10.1145/3610548.3618215

Received 23 January 2025