Geosci. Model Dev., 19, 3953–3987, 2026
https://doi.org/10.5194/gmd-19-3953-2026

Geoscientific Model Development
Open Access
EGU

SWEpy: an open-source GPU-accelerated solver for near-field inundation and far-field tsunami modeling

Juan Fuenzalida[1], Danilo Kusanovic[2], Joaquín Meza[1], Rodrigo Meneses[3], and Patricio A. Catalán[1]

[1]Departamento de Obras Civiles, Universidad Técnica Federico Santa María, Valparaiso, Chile
[2]Department of Civil and Environmental Engineering, University of California Davis, Davis, CA, 95616, USA
[3]Escuela de Ingenieria Civil, Universidad de Valparaiso, Valparaiso, Chile

Correspondence: Joaquín Meza (joaquin.meza@usm.cl)

Received: 12 August 2025 – Discussion started: 4 October 2025
Revised: 3 February 2026 – Accepted: 25 March 2026 – Published: 13 May 2026

Abstract. We present SWEpy, an open-source Python finite volume (FV) software for solving the shallow water equations (SWEs) on unstructured triangular meshes. The framework combines flexibility and high performance through GPU acceleration and a well-balanced, positivity-preserving, higher-order central-upwind (CU) scheme. These features are required for simulation of hydrodynamic phenomena such as tsunami propagation, flooding, and dam-break flows in complex and large geometries.

To reduce numerical diffusion, a phenomenon commonly encountered in FV methods, SWEpy incorporates a second-order WENO reconstruction together with a third-order strong stability-preserving Runge–Kutta time integration scheme. These numerical components are particularly well-suited for far-field tsunami modeling, where minimizing artificial diffusion is essential to accurately preserve wave amplitude, phase, and dispersion over long propagation distances.

The performance, stability, and accuracy of SWEpy are validated using canonical benchmarks, including Synolakis' conical island and Bryson's flow over a Gaussian bump. Its capabilities are further demonstrated through large-scale simulations of the 1959 Malpasset Dam failure and the 2010 Maule tsunami, highlighting its effectiveness in realistic scenarios. Overall, these results show that SWEpy framework delivers high-resolution solutions on consumer-grade hardware, providing a user-friendly and computationally efficient platform for both research applications and operational forecasting.

1 Introduction

Accurate simulation of hazardous hydrological events – such as dam failures, tsunamis, and urban flooding – is essential for risk assessment, emergency planning, and the operation of early warning systems (Catalan et al., 2020; Fernández-Nóvoa et al., 2024; Lin et al., 2015; Behrens et al., 2010; Harig et al., 2019). Accordingly, over the past three decades, a wide range of numerical tools has been developed for these applications, including both open-source and commercial software (ANSYS Inc., 2013; Jodhani et al., 2023). These software enable stakeholders to generate reliable data for evaluating community vulnerability.

However, emerging global challenges – including rapid urbanization, climate change, and increasing socio-economic uncertainty, particularly in developing regions (United Nations, 2019) – are placing greater demands on risk management frameworks. Meeting these demands requires advanced modeling tools that are not only accurate and computationally efficient, but also flexible, scalable, and accessible to a diverse range of users across research, planning, and operational settings. In this context, open-source software plays an important role by enabling collaborative development, encouraging users to extend, adapt, and improve models while remaining accessible without commercial barriers.

To better understand the current landscape, we compile a representative (though not exhaustive) set of freely available shallow water equation (SWE) solvers in Table 1, highlighting key features such as numerical schemes, grid types, and parallelization capabilities. As shown, while many tools excel in specific applications – such as rainfall–runoff modeling

or flood simulation – there remains a gap in flexible solvers that combine unstructured triangular meshes with high-order reconstruction methods for improved accuracy in complex geometries.

These programs solve nonlinear SWEs, which have become the cornerstone of two-dimensional free-surface flow modeling across a wide range of applications (Delis and Nikolos, 2021). However, the growing demand for higher spatial resolution and real-time performance has motivated alternative approaches. Some efforts have focused on simplified formulations (Courty et al., 2017) or machine-learning surrogates (Kabir et al., 2020; Zhou et al., 2022; Shaeri Karimi et al., 2019) to reduce computational cost, often at the expense of physical fidelity (Fernández-Pato et al., 2018). In contrast, a more robust strategy relies on parallelization, leveraging high-performance computing on CPUs and GPUs (Caviedes-Voullième et al., 2023; Morales-Hernández et al., 2021; Reinarz et al., 2020) to achieve significant speedups without compromising accuracy.

Indeed, several mature SWE solvers have demonstrated strong performance and robustness in large-scale, real-world applications. Examples include Tsunami-HySEA (Macías et al., 2017), GeoClaw (Berger et al., 2011), MOST (Titov et al., 2016), ADCIRC (Tanaka et al., 2010), exaHYPE (Reinarz et al., 2020), and TsunAWI (Behrens, 2008). These software exploit parallelization through OpenMP, MPI, CUDA, or hybrid approaches, and employ either fixed meshes or adaptive mesh refinement (AMR) to efficiently capture localized dynamics. Their successful deployment in tsunami early-warning systems, coastal hazard analysis, and continental-scale flooding underscores the critical role of scalable high-performance computing in SWE modeling.

Despite their advanced capabilities, many existing solvers are implemented in low-level languages such as FORTRAN, C++, or C, and depend on specialized APIs (e.g., CUDA, MPI, or OpenMP). While these choices enable high performance, they can also limit accessibility for users without advanced programming expertise. In contrast, high-level languages like Python provide a more accessible alternative. However, efficient parallelization in Python is not straightforward due to limitations such as the Global Interpreter Lock (Turner and Wouters, 2024). To overcome these challenges, libraries such as Numba (Lam et al., 2015), PyCUDA (Kloeckner et al., 2025), TensorFlow (Abadi et al., 2015), PyTorch (Ansel et al., 2024), and Dask (Rocklin, 2015) enable efficient parallel execution – particularly on GPUs – through compiled kernels and SIMD-based approaches.

In this context, CuPy (Okuta et al., 2017) provides a particularly attractive solution. As a drop-in replacement for NumPy (Harris et al., 2020), CuPy executes array operations using NVIDIA CUDA kernels, enabling seamless GPU acceleration with minimal code modification. By building on the mature CUDA ecosystem, it supports custom kernels, reproducible high-performance execution, and scalable multi-GPU workflows – all while maintaining a user-friendly interface that avoids low-level GPU programming. While today experimental support exists for alternative backends in CuPy, the focus on a well-established CUDA platform provides a robust and reliable foundation for advanced numerical modeling without sacrificing efficiency or accessibility.

Although most SWE solvers incorporate some form of parallelization, they vary significantly in scope, numerical methods, grid geometries, and implementation/-parallelization strategies (e.g., CUDA, Kokkos, MPI, or OpenMP). Some are tailored to specific applications, such as rainfall–runoff modeling (e.g., SERGHEI-SWE, HiPIMS) or tsunami generation-propagation-inundation (e.g., COMCOT, Tsunami-HySEA, TsunAWI), while others are designed for more general-purpose simulations. This diversity reflects necessary design trade-offs, but also highlights the lack of a unified framework that combines flexibility across these dimensions.

A key factor influencing both solver performance and applicability is grid discretization. Structured Cartesian grids are widely used due to their simplicity and computational efficiency, particularly when combined with adaptive mesh refinement (AMR). In contrast, unstructured triangular meshes provide greater geometric flexibility, enabling localized refinement in regions with complex bathymetry, irregular coastlines, or urban features without requiring hierarchical grid structures (Schubert et al., 2008). This flexibility makes them especially well-suited for multi-scale problems – such as tsunami propagation coupled with near-shore inundation – where localized resolution is critical, and can eliminate the need for nested grids in complex simulations (Harig et al., 2008; Bomers et al., 2019).

A large class of SWE solvers is based on finite-volume (FV) Godunov-type schemes, which employ approximate Riemann solvers – such as Roe, HLL, or HLLC – to compute numerical fluxes. These methods are well established, robust, and highly optimized for large-scale applications, providing accurate resolution of wave propagation and discontinuities. An alternative class of methods are offered by central-upwind (CU) schemes, which avoid the explicit solution of Riemann problems by estimating local propagation speeds to construct numerical fluxes. Although not intended to replace Riemann-solver-based methods, CU schemes provide a conceptually simpler and more flexible framework, particularly attractive for implementations on unstructured grids and for coupling with high-order reconstruction techniques. Originally introduced for Cartesian grids by Kurganov and Tadmor (2000) and later extended to triangular meshes by Kurganov and Petrova (2005), CU schemes approximate fluxes by integrating over local Riemann fans defined by these estimated speeds.

Despite their advantages, CU formulations for the SWE are often affected by excessive numerical diffusion. Recent efforts to mitigate this issue have largely focused on Cartesian-grid finite-difference formulations (Kurganov and Xin, 2023; Chu et al., 2025; Cui et al., 2025), leaving tri-

Table 1. Overview of some of the openly available SWE solvers, highlighting application scope, numerical formulation, grid type, and parallel execution strategies. The *Parallelization framework* column identifies the primary model-level parallel approach (e.g., MPI, OpenMP, CUDA, Kokkos, CuPy), while the *GPU* column indicates the availability of native GPU acceleration. The table is not intended to be exhaustive, but rather to contextualize SWEpy relative to commonly used flooding and tsunami models.

Model	Reference	GPU	Parallelization framework	License/ Availability	Scope	Scheme type	Grid
SERGHEI-SWE	Caviedes-Voullième et al. (2023)	Yes	MPI + Kokkos	Open-source (BSD)	Rainfall runoff	FV Roe	Cartesian
TRITON	Morales-Hernández et al. (2021)	Yes	MPI + CUDA	Open-source (BSD)	Flooding	FV Roe	Cartesian
PARFLOOD	Vacondio et al. (2014)	Yes	MPI + CUDA	Upon request	Flooding	FV HLLC	Cartesian
HiPIMS	Xia et al. (2019)	Yes	CUDA	Open-source (GPLv3)	Rainfall runoff	FV HLLC	Cartesian
DRR/FI	Kobayashi et al. (2015)	No	MPI	–	Rainfall runoff	FD Leapfrog	Cartesian
SW2D-GPU	Carlotto et al. (2021)	Yes	CUDA	Open-source	Flooding, lake water level	FD Leapfrog	Cartesian
LisFlood-FP 8.0	Shaw et al. (2021)	Yes	CUDA	Open-source (GPLv3)	Flooding	FE/FV DG	Cartesian
IBER	García-Feal et al. (2018)	Yes	CUDA	Freeware	Flooding, rivers, estuaries	FV Roe	Unstr. tri. & quad.
SW2D-Lemon	Steinstraesser et al. (2022)	No	–	Freeware	Flooding (upscaled model)	FV HLL	Unstr. poly.
B-flood	Kirstetter et al. (2021)	No	–	Open-source (GPL)	Flash flooding	FV HLLC	Adaptive quad.
FullSWOF	Delestre et al. (2017)	No	MPI	Open-source (CeCILL)	Rainfall runoff	FV HLLC	Cartesian
TELEMAC	Moulinec et al. (2011)	No	MPI	Open-source (GPLv3)	General purpose	Various FE/FV Godunov	Unstr. tri.
GeoClaw	Berger et al. (2011)	Partial[c]	OpenMP	Open-source (BSD)	General purpose	FV wave-propagation (Godunov-type)	Adaptive Cartesian
HEC-RAS 2D	Brunner (2021)	Partial[a]	–	Freeware	Channel, floodplain	Implicit FV	Unstr. poly.
HMS	Simons et al. (2013)	No	MPI	Open-source (GPL)	Overland flow with transport-reaction	FV HLLC	Unstr. tri. & quad.
COMCOT	Wang and Power (2011)	No	OpenMP	Open-source (GPL)	Tsunami GPI[b]	FD Leapfrog	Cartesian
Tsunami-HySEA	Macías et al. (2017)	Yes	CUDA	Open-source (GPL)	Tsunami GPI	FV WAF	Cartesian
TsunAWI	Behrens (2008)	No	MPI	Open-source (GPL)	Tsunami GPI	FE LC–LNC	Unstr. tri.
MOST	Titov et al. (2016)	No	OpenMP/OpenACC/OpenCL	Upon request	Tsunami GPI	FD	Cartesian
ADCIRC	Tanaka et al. (2010)	No	MPI	Upon request	Coastal, storm surge	FE Galerkin	Unstr. tri.
exaHyPE	Reinarz et al. (2020)	Yes	Multiple	Open-source (BSD)	General hyperbolic PDEs	ADER-DG/FV	Adaptive Cartesian
`SWEpy`	This article	Yes	CuPy (CUDA)	Open-source (GPL)	General purpose	FV Central-Upwind	Unstr. tri.

[a] GPU support in development (HEC-RAS 2025 Alpha). [b] Generation, Propagation, Inundation. [c] GPU support under active development (not yet merged into the main release).
MPI is available in the dispersive (SGN) version through PETSc.

angular FV implementations comparatively underexplored. One promising strategy to reduce diffusion while preserving the CU framework is the incorporation of high-order polynomial reconstructions. However, advanced schemes such as ENO/WENO (Zhang and Shu, 2016) introduce significant computational overhead due to large stencils, smoothness indicator evaluations, and potentially complex neighbor-search procedures. In this context, high-performance computing becomes essential. Although high-order WENO schemes can be computationally prohibitive in serial CPU implementations, their high arithmetic intensity makes them particularly well suited for GPU acceleration, where their computational cost can be effectively amortized.

Furthermore, the behavior of the SWE is governed not only by flux terms but also by source terms in the balance equations, as well as domain characteristics such as bathymetry and boundary conditions. This diversity of physical settings requires numerical formulations that can accommodate a broad range of scenarios while maintaining both accuracy and stability. In addition to the choice of numerical flux, the design of spatial discretization plays a critical role and must satisfy key physical and numerical properties, including well-balancing and positivity preservation, as extensively discussed in the literature (e.g., Kurganov and Tadmor, 2000; Kurganov and Petrova, 2005; Toro et al., 1994). Well-balancing ensures the exact preservation of steady states – such as a lake at rest over variable topography or geostrophic equilibrium – thereby preventing spurious oscillations that can degrade accuracy in long-term or large-scale simulations. Positivity preservation, on the other hand, guarantees non-negative water depths, which is essential for stable and physically meaningful solutions near wet-dry interfaces. Achieving these properties requires carefully designed reconstruction operators that are consistently integrated with the underlying numerical scheme.

Motivated by these considerations, we introduce `SWEpy`, an open-source, Python-based FV solver for the SWE on unstructured triangular grids with GPU acceleration. The proposed framework is designed to overcome limitations of existing approaches by enabling efficient and flexible modeling of both near-field and far-field phenomena within a unified formulation. The main contributions of this work are threefold: (1) the development of a CU scheme extended to higher-order accuracy through quadratic WENO reconstruction and third-order strong-stability-preserving (SSP) Runge–Kutta time integration, reducing numerical diffusion while accurately capturing both shock-dominated and wave-propagation regimes; (2) an open-source release under the GNU General Public License (GPL), promoting transparency, reproducibility, and community-driven development; and (3) a Python/CuPy implementation that enables efficient GPU acceleration on CUDA-compatible hardware, lowering the barrier to high-performance computing while maintaining computational efficiency. `SWEpy` is designed to be well-balanced and positivity-preserving, and its performance is validated against both canonical benchmarks and real-world applications, including the Malpasset dam-break (near-field inundation) and the 2010 Maule tsunami (basin-scale propagation). Building on these results, the remainder of this paper is organized as follows: Sect. 2 presents the governing equations and FV formulation; Sect. 3 describes the numerical scheme and reconstruction procedures; Sect. 4 details the Python/CuPy implementation; Sect. 5 presents validation and performance results; and Sect. 6 concludes with a discussion of implications and future directions.

2 Problem Setting

This section defines the physical problem, presents the governing equations, and derives the semi-discrete form of the SWEs, which provides the foundation for the CU scheme with WENO reconstruction on unstructured triangular grids implemented in `SWEpy`.

2.1 Governing Equations

The SWEs, originally introduced in one dimension by Saint-Venant (1871), are obtained in two dimensions through a depth-averaging of the Navier–Stokes equations under the hydrostatic pressure assumption. Assuming incompressibility and an appropriate scaling in which vertical accelerations are negligible, the vertical momentum equation reduces to a hydrostatic pressure distribution. This simplification allows the continuity and horizontal momentum equations to be integrated from the bed to the free surface, yielding a system expressed in terms of depth-averaged variables (Castro-Orgaz and Hager, 2019; Chow et al., 1988; Hervouet, 2007; Vreugdenhil, 1994). The resulting equations constitute the foundation of computational models for free-surface flows in rivers, coastal regions, and urban floodplains, and are capable of describing a wide range of phenomena, including flood waves, tsunamis, and storm surges (Delis and Nikolos, 2021).

To establish a consistent notation, we define the state vector $\boldsymbol{q} = (h, hu, hv)^\top$, where $h(x, y, t)$ denotes the water depth measured from the bathymetry $B(x, y)$ to the free surface $w(x, y, t)$ relative to the $z = 0$ reference plane. The variables $u(x, y, t)$ and $v(x, y, t)$ represent the depth-averaged velocity components in the x and y directions, while $hu(x, y, t)$ and $hv(x, y, t)$ correspond to the associated discharge fluxes. Scalar quantities (e.g., h) are denoted by regular symbols, whereas vector quantities (e.g., $\boldsymbol{q}$) are indicated in bold. A schematic representation of these variables is provided in Fig. 1.

This definition of variables allows to express the SWEs in conserved vector form as:

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x + \boldsymbol{g}(\boldsymbol{q})_y = \boldsymbol{S}_{\mathrm{B}}(\boldsymbol{q}) + \boldsymbol{S}(\boldsymbol{q}), \tag{1}$$

where $(\cdot)_t$, $(\cdot)_x$, and $(\cdot)_y$ denote partial derivatives with respect to t, x, and y, respectively. In addition, the associated

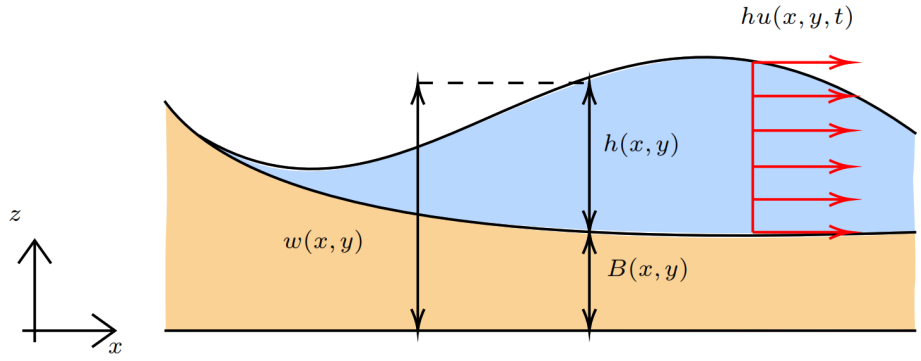

Figure 1. Model setting and physical variables used in the SWE.

vector quantities are

$$\boldsymbol{q} = (h, hu, hv)^\top, \tag{2}$$

$$\boldsymbol{f}(\boldsymbol{q}) = \left(hu, hu^2 + \frac{gh^2}{2}, huv\right)^\top, \tag{3}$$

$$\boldsymbol{g}(\boldsymbol{q}) = \left(hv, huv, hv^2 + \frac{gh^2}{2}\right)^\top, \quad \text{and} \tag{4}$$

$$\boldsymbol{S}_\mathrm{B}(\boldsymbol{q}) = \left(0, -ghB_x, -ghB_y\right)^\top, \tag{5}$$

where $\boldsymbol{S}(\boldsymbol{q})$ denotes additional source terms, including Coriolis forcing, bottom friction, rheological effects, and turbulence.

In this study, we consider bottom friction $\boldsymbol{S}_\mathrm{F}(\boldsymbol{q})$ and Coriolis forcing $\boldsymbol{S}_\mathrm{C}(\boldsymbol{q})$, which are particularly relevant for modeling dam-break flows and large-scale tsunami propagation, respectively. Bottom friction is modeled using the semi-empirical Manning–Strickler formulation (Chow et al., 1988), while the Coriolis term accounts for rotational effects. These source terms are given by

$$\boldsymbol{S}_\mathrm{C}(\boldsymbol{q}) = (0, fhv, -fhu)^\top, \text{and} \tag{6}$$

$$\boldsymbol{S}_\mathrm{F}(\boldsymbol{q}) = \left(0, -g\frac{n^2}{h^{7/3}}hu\sqrt{(hu)^2+(hv)^2}, -g\frac{n^2}{h^{7/3}}hv\sqrt{(hu)^2+(hv)^2}\right)^\top, \tag{7}$$

where n denotes the Manning friction coefficient and f is the Coriolis parameter, typically approximated as $10^{-4}, \mathrm{s}^{-1}$ (Kundu et al., 2012). For long-range tsunami propagation, both bathymetric and Coriolis source terms ($\boldsymbol{S}_\mathrm{B}$ and $\boldsymbol{S}_\mathrm{C}$) are considered, whereas flooding applications incorporate bathymetry and bottom friction ($\boldsymbol{S}_\mathrm{B}$ and $\boldsymbol{S}_\mathrm{F}$).

The resulting system of nonlinear hyperbolic equations requires robust numerical methods to accurately capture shocks and maintain stability, particularly in the presence of complex geometries and wet-dry interfaces. These challenges are addressed in SWEpy through a CU scheme on unstructured triangular grids, as detailed in Sect. 3.

2.2 Semi-discrete formulation

FVMs provide a robust framework for solving the SWEs, ensuring conservation of mass and momentum while accurately handling discontinuities such as shocks and wet-dry fronts (LeVeque, 2002; Moukalled et al., 2015; Toro, 2001). In this approach, the governing equations are integrated over discrete control volumes, yielding cell-averaged quantities that are evolved in time through numerical fluxes evaluated at cell interfaces. This formulation naturally captures sharp features, such as hydraulic jumps and bore propagation, without the need for additional artificial viscosity (Stiernström et al., 2021), making it particularly well suited for shallow water flows characterized by abrupt regime transitions.

To improve well-balancing[1], we reformulate the system by expressing the conserved variables in terms of the free-surface elevation $w = h + B$ rather than the water depth h. Specifically, the state vector becomes $\boldsymbol{q}(x, y, t) = (w, hu, hv)^\top$. This substitution mitigates numerical imbalances associated with topographic gradients and leads to the following system:

$$\boldsymbol{q}_t + \boldsymbol{F}(\boldsymbol{q}, B)_x + \boldsymbol{G}(\boldsymbol{q}, B)_y = \boldsymbol{S}_\mathrm{B}(\boldsymbol{q}, B) + \boldsymbol{S}(\boldsymbol{q}, B), \tag{8}$$

where

$$\boldsymbol{F}(\boldsymbol{q}, B) = \left(hu, \frac{(hu)^2}{w-B} + \frac{1}{2}g(w-B)^2, \frac{(hu)(hv)}{w-B}\right)^\top, \tag{9}$$

$$\boldsymbol{G}(\boldsymbol{q}, B) = \left(hv, \frac{(hu)(hv)}{w-B}, \frac{(hv)^2}{w-B} + \frac{1}{2}g(w-B)^2\right)^\top. \tag{10}$$

We then consider a triangular discretization of the polygonal spatial domain $\Omega = \bigcup_j \Omega_j$, including additional "ghost" cells for boundary conditions, as displayed in Fig. 2 (bottom border). Integrating Eq. (8) over each cell Ω_j and applying the Gauss divergence theorem yields:

$$\frac{d}{dt}\int_{\Omega_j} \boldsymbol{q}\,\mathrm{d}\Omega + \oint_{\partial\Omega_j} (\boldsymbol{F}, \boldsymbol{G})\hat{\boldsymbol{n}}_j\,\mathrm{d}l_j = \int_{\Omega} (\boldsymbol{S}_B + \boldsymbol{S})\mathrm{d}\Omega. \tag{11}$$

[1]This means the exact preservation of steady states, particularly in the presence of variable bathymetry.

Let Ω_{jk} $(k = 1, 2, 3)$ denote the cells neighboring Ω_j. Now, let (x_j, y_j) be the barycenter of Ω_j, and Γ_{jk} the edge shared with Ω_{jk}, with length l_{jk}. The outward unit normal vector to Γ_{jk} is given by $\hat{\boldsymbol{n}}_{jk} = \big(\cos(\theta_{jk}), \sin(\theta_{jk})\big)$, where θ_{jk} denotes its orientation. Then, the semi-discrete formulation is written as

$$\frac{d}{dt}\overline{\boldsymbol{q}}_j(t) + \frac{1}{|\Omega_j|}\sum_{k\in\mathcal{N}_j}\mathcal{F}_{jk}\, l_{jk} = \overline{\boldsymbol{S}}_{\mathrm{B}_j} + \overline{\boldsymbol{S}}_j, \tag{12}$$

where $\overline{\boldsymbol{q}}_j = \frac{1}{|\Omega j|}\int_{\Omega_j}\boldsymbol{q}(x, y, t)\,\mathrm{d}\Omega$ denotes the cell-averaged state, and $\mathcal{F}_{jk}$ is the numerical flux across the interface Γ_{jk}, accounting for interactions between neighboring cells. This flux is constructed from the physical flux functions $\boldsymbol{F}$ and $\boldsymbol{G}$ in Eq. (9) and depends on reconstructed states at the interface, denoted $\boldsymbol{q}^{\mathrm{in}}_{jk}$ and $\boldsymbol{q}^{\mathrm{out}}_{jk}$, obtained via suitable reconstruction operators.

The boundary integrals are evaluated using Gaussian quadrature, with the number of quadrature points determined by the order of the reconstruction. Since the reconstructed states are time-dependent, Eq. (12) defines a system of ordinary differential equations that needs to be integrated numerically. In this equation, the terms $\overline{\boldsymbol{S}}_{\mathrm{B}_j}$ and $\overline{\boldsymbol{S}}_j$ represent consistent discretizations of the bathymetric and additional source terms, respectively, and will be detailed in the following sections. This formulation provides the basis for the CU discretization described in Sect. 3.

3 SWEpy Numerical Model

Building on the semi-discrete formulation, SWEpy employs a CU FVM for solving hyperbolic conservation laws on unstructured triangular grids. This approach was originally introduced by Kurganov and Petrova (2005), with parallel developments addressing well-balancing and related formulations by Bryson and Levy (2005) and Xie et al. (2005), and further refined in subsequent studies. This section presents a concise description of the numerical formulation adopted in this work.

3.1 Central-Upwind Numerical Fluxes on Triangular Grids

The CU scheme avoids the explicit solution of Riemann problems by estimating local wave propagation speeds to construct numerical fluxes. This results in a method that balances computational simplicity with robustness, making it particularly well suited for shallow water flows over complex geometries, including variable bathymetry and wet-dry interfaces.

The numerical flux $\mathcal{F}_{jk}$ in Eq. (12) is formulated as the projection onto the edge-normal direction Γ_{jk}:

$$\mathcal{F}_{jk} = \Big(F_{jk}\cos(\theta_{jk}) + G_{jk}\sin(\theta_{jk})\Big) - \frac{a^{\mathrm{in}}_{jk}a^{\mathrm{out}}_{jk}}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}} \sum_{s=1}^{N_s} c_s\big[\boldsymbol{q}^{\mathrm{in}}_{jk}(M^s_{jk}) - \boldsymbol{q}^{\mathrm{out}}_{jk}(M^s_{jk})\big], \tag{13}$$

where M^s_{jk} are the Gaussian quadrature points along the edge, and c_s are the associated weights. The number of integration points N_s depends on the reconstruction order to ensure accurate integration. The terms a^{in}_{jk} and a^{out}_{jk} represent the inward and outward local propagation speeds, given in Eq. (19). In addition, the flux projection components in Eq. (13) are:

$$F_{jk} = \frac{1}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}}\sum_{s=1}^{N_s} c_s\Big[a^{\mathrm{in}}_{jk}\boldsymbol{F}\Big(\boldsymbol{q}^{\mathrm{in}}_{jk}(M^s_{jk}), B(M^s_{jk})\Big) + a^{\mathrm{out}}_{jk}\boldsymbol{F}\Big(\boldsymbol{q}^{\mathrm{out}}_{jk}(M^s_{jk}), B(M^s_{jk})\Big)\Big], \quad \text{and} \tag{14}$$

$$G_{jk} = \frac{1}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}}\sum_{s=1}^{N_s} c_s\Big[a^{\mathrm{in}}_{jk}\boldsymbol{G}\Big(\boldsymbol{q}^{\mathrm{in}}_{jk}(M^s_{jk}), B(M^s_{jk})\Big) + a^{\mathrm{out}}_{jk}\boldsymbol{G}\Big(\boldsymbol{q}^{\mathrm{out}}_{jk}(M^s_{jk}), B(M^s_{jk})\Big)\Big]. \tag{15}$$

In Eqs. (14) and (15), the vectors $\boldsymbol{q}^{\mathrm{out}}_{jk}$ and $\boldsymbol{q}^{\mathrm{in}}_{jk}$ denote the limiting values of $\boldsymbol{q}(x, y)$ at the interface point M^s_{jk} approached from within Ω_j and Ω_{jk}, respectively. Moreover, due to the form of the fluxes F_{jk} and G_{jk}, divisions by zero may arise near wet–dry interfaces. To prevent such singularities, a consistent flux treatment is required, as addressed through the positivity-preserving reconstruction described in Sect. 3.4.

Substituting Eqs. (13), (14) and (15) into Eq. (12) yields

$$\begin{aligned}\frac{\mathrm{d}\overline{\boldsymbol{q}}_j}{\mathrm{d}t} = &-\frac{1}{|\Omega_j|}\sum_{k=1}^{3}\frac{l_{jk}\cos\theta_{jk}}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}}\sum_{s=1}^{N_s}c_s \\ &\Big[a^{\mathrm{in}}_{jk}\boldsymbol{F}\Big(\boldsymbol{q}_{jk}(M^s_{jk}), B(M^s_{jk})\Big) + a^{\mathrm{out}}_{jk}\boldsymbol{F}\Big(\boldsymbol{q}_j(M^s_{jk}), B(M^s_{jk})\Big)\Big] \\ &-\frac{1}{|\Omega_j|}\sum_{k=1}^{3}\frac{l_{jk}\sin\theta_{jk}}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}}\sum_{s=1}^{N_s}c_s\Big[a^{\mathrm{in}}_{jk}\boldsymbol{G}\Big(\boldsymbol{q}_{jk}(M^s_{jk}), B(M^s_{jk})\Big) \\ &+a^{\mathrm{out}}_{jk}\boldsymbol{G}\Big(\boldsymbol{q}_j(M^s_{jk}), B(M^s_{jk})\Big)\Big] \\ &+\frac{1}{|\Omega_j|}\sum_{k=1}^{3}l_{jk}\frac{a^{\mathrm{in}}_{jk}a^{\mathrm{out}}_{jk}}{a^{\mathrm{in}}_{jk} + a^{\mathrm{out}}_{jk}}\sum_{s=1}^{N_s}c_s\Big[\boldsymbol{q}_{jk}(M^s_{jk}) - \boldsymbol{q}_j(M^s_{jk})\Big] \\ &+\overline{\boldsymbol{S}}_{\mathrm{B}_j} + \overline{\boldsymbol{S}}_j,\end{aligned} \tag{16}$$

where $\overline{\boldsymbol{q}}_j$ is the cell-averaged state, the bathymetry $B(M^s_{jk})$ follows the same reconstruction criterion as the flux variables, and $\overline{\boldsymbol{S}}_j$ is the discretized source term, discussed in the following subsection.

The one-sided local speeds[2] a^{in}_{jk} and a^{out}_{jk} are evaluated at Gaussian quadrature points using desingularized velocities to

[2]Maximum wave speeds governing information propagation across the jk interface.

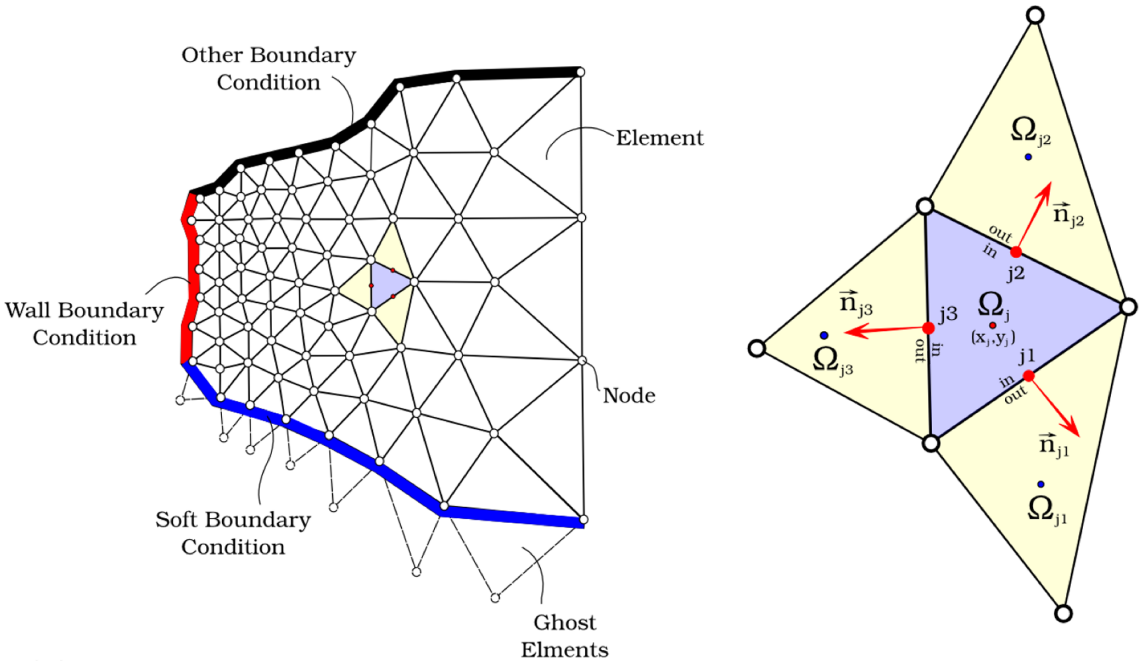

Figure 2. Illustration of a triangular unstructured grid. The figure shows on the left an example of a finite volume grid, while on the right a typical triangular cell with some attributes used in the semi-discrete formulation.

ensure stable behavior near dry states. The velocity components are regularized as

$$u = \frac{\sqrt{2}, h(hu)}{\sqrt{h^4 + \max(h^4, \varepsilon)}}, \text{ and } v = \frac{\sqrt{2}, h(hv)}{\sqrt{h^4 + \max(h^4, \varepsilon)}}, \quad (17)$$

where $h = w - B$ denotes the water depth and ε is a small tolerance parameter that prevents singularities as $h \to 0$. These velocities are then projected onto the outward normal direction of the interface Γ_{jk}, yielding

$$u_j^\theta(M_{jk}^s) = \cos(\theta_{jk}), u_j(M_{jk}^s) + \sin(\theta_{jk}), v_j(M_{jk}^s)$$
$$\text{and } u_{jk}^\theta(M_{jk}^s) = \cos(\theta_{jk}), u_{jk}(M_{jk}^s)$$
$$+ \sin(\theta_{jk}), v_{jk}(M_{jk}^s) \quad (18)$$

where the subscripts j and jk denote reconstructed values from cell Ω_j and its neighbor Ω_{jk}, respectively.

The one-sided local wave speeds are then obtained by taking extrema over all Gaussian points:

$$a_{jk}^{\text{out}} = \max_s \left\{ \max \left\{ u_j^\theta(M_{jk}^s) + \sqrt{g h_j(M_{jk}^s)},\ u_{jk}^\theta(M_{jk}^s) \right.\right.$$
$$\left. + \sqrt{g h_{jk}(M_{jk}^s)},\ 0 \right\} \Big\},$$
$$a_{jk}^{\text{in}} = -\min_s \left\{ \min \left\{ u_j^\theta(M_{jk}^s) - \sqrt{g h_j(M_{jk}^s)},\ u_{jk}^\theta(M_{jk}^s) \right.\right.$$
$$\left. - \sqrt{g h_{jk}(M_{jk}^s)},\ 0 \right\} \Big\}. \quad (19)$$

3.2 Well-balancing of the source terms

A fundamental requirement for robust SWE solvers – particularly in applications involving complex topography such as dam-break flows and tsunamis – is well-balancing. This property is defined as the exact preservation of steady-state solutions without introducing spurious oscillations. Well-balance is essential for maintaining physical accuracy in scenarios such as the lake-at-rest condition or geostrophic equilibrium, where source terms must precisely counterbalance flux gradients (Kurganov and Petrova, 2007; Bryson et al., 2011; Liu et al., 2018; Chertock et al., 2015, 2018; Desveaux and Masset, 2022; Greenberg and Leroux, 1996; Liu, 2021b; Cao et al., 2024). Therefore, in SWEpy, well-balancing is achieved through a consistent discretization of the source terms, ensuring that numerical fluxes remain in equilibrium with the underlying physical forces across both fully wet domains and regions with variable bathymetry.

3.2.1 Bathymetry gradient contribution

For the bathymetry source term, a well-balanced discretization is constructed by enforcing the lake-at-rest condition, ensuring consistency with the corresponding momentum flux contributions (Bryson et al., 2011). Thus, variables are first reconstructed using polynomial approximations, after which the resulting expressions are integrated over both the cell and its interfaces using Gaussian quadrature. This procedure yields a general formulation that is applicable to arbitrary reconstruction orders:

$$\overline{\boldsymbol{S}}_{\mathrm{B}_j}^{(l)} = -g \sum_{s=1}^{N_s^{\mathrm{int}}} c_s \frac{\partial w(M_j^s)}{\partial x} \left(w_j(M_j^s) - B(M_j^s) \right)$$
$$+ \frac{g}{2|\Omega_j|} \sum_{k=1}^{3} \sum_{s=1}^{N_s} c_s l_{jk} (w_j(M_{jk}^s) - B(M_{jk}^s))^2 \cos\theta_{jk},$$
$$\overline{\boldsymbol{S}}_{\mathrm{B}_j}^{(3)} = -g \sum_{s=1}^{N_s^{\mathrm{int}}} c_s \frac{\partial w(M_j^s)}{\partial y} \left(w_j(M_j^s) - B(M_j^s) \right)$$
$$+ \frac{g}{2|\Omega_j|} \sum_{k=1}^{3} \sum_{s=1}^{N_s} c_s l_{jk} (w_j(M_{jk}^s) - B(M_{jk}^s))^2 \sin\theta_{jk}, \quad (20)$$

In Eq. (20), N_s^{int} denotes the number of Gaussian quadrature points M_j^s used over the cell interior Ω_j, and c_s are the corresponding weights. The bathymetry B is reconstructed using the same-order operator employed for the flow variables, ensuring consistency in the discretization.

This formulation enforces a consistent balance between source terms and flux contributions, eliminating spurious flows over irregular topography and ensuring the preservation of equilibrium states, as verified in the steady-state benchmarks in Sect. 5.

3.2.2 Manning friction

The structure of the Manning friction source term allows for a straightforward well-balanced discretization, as it is directly proportional to the discharge components $\overline{hu}$ and $\overline{hv}$. Consequently, the term vanishes at equilibrium without requiring additional modifications. However, near wet–dry interfaces or in regions where $h \to 0$, desingularization is necessary to avoid division by zero and ensure numerical stability.

Following Chertock et al. (2015), we define the discrete friction coefficient as

$$\mathcal{G}(\overline{\boldsymbol{q}}_j) := -gn^2 \left(\frac{2\overline{h}_j}{\overline{h}_j^2 + \max\left(\overline{h}_j^2, \varepsilon^2\right)} \right)^{7/3} \sqrt{(\overline{hu})_j^2 + (\overline{hv})_j^2}, \quad (21)$$

where $\overline{h}_j = \overline{w}_j - \overline{B}_j$ denotes the cell-averaged water depth and ε is a small regularization parameter introduced to avoid singularities.

The corresponding discretized friction source term is then given by

$$\overline{\boldsymbol{S}}_{\mathrm{F}_j} = \mathcal{G}(\overline{\boldsymbol{q}}_j) \begin{bmatrix} 0 & (\overline{hu})_j & (\overline{hv})_j \end{bmatrix}^T. \quad (22)$$

This formulation is treated semi-implicitly during time integration, as described in Sect. 3.5, to account for the stiffness of the friction source term (Chertock et al., 2015). In SWEpy, the coefficient $\mathcal{G}$ is evaluated in a vectorized manner across all cells on the GPU, enabling efficient parallel computation even for large-scale grids. Although the Manning coefficient n may vary spatially, it is taken as constant throughout the domain in the presented numerical experiments.

3.2.3 Coriolis

The Coriolis source term vanishes under zero-discharge equilibria (i.e., $hu = hv = 0$), ensuring inherent well-balancing in such states without requiring specialized discretization beyond direct averaging. Accordingly, the cell-averaged Coriolis source term is:

$$(\overline{\boldsymbol{S}}_{\mathrm{C}})_j = f \begin{bmatrix} 0 & (\overline{hv})_j & -(\overline{hu})_j \end{bmatrix}^\top, \quad (23)$$

where f is the Coriolis parameter, typically approximated as $10^{-4}\,\mathrm{s}^{-1}$ in mid-latitude regions (Kundu et al., 2012), as adopted in the Maule 2010 tsunami validation in Sect. 5.2.2.

For large-scale geophysical flows, however, a more rigurous condition – geostrophic balance – may be required. In this regime, horizontal pressure gradients are balanced by Coriolis forces, leading to nontrivial steady states. Several numerical approaches have been proposed to preserve this balance in rotating SWEs (Desveaux and Masset, 2022; Chertock et al., 2018). In the present work, SWEpy employs standard well-balancing, with extensions toward geostrophic preservation left for future development.

3.3 Spatial Reconstruction Operators and Scheme Formulation

The design of these reconstruction operators is guided by both physical and numerical requirements inherent to shallow water flows, including variable bathymetry, surface roughness, and domain scale. These features demand accurate representation of gradients and discontinuities, particularly in near-field regimes (e.g., shocks and wet–dry fronts) and far-field wave propagation, as demonstrated in Sect. 5.

To ensure physically meaningful and numerically stable solutions, the reconstruction must satisfy key properties. These include well-balancing, which guarantees the exact preservation of steady states – such as lake-at-rest or geostrophic equilibrium – thereby preventing spurious oscillations (Bryson et al., 2011), and positivity preservation, which enforces non-negative water depths and is essential for realistic inundation modeling.

Numerical experiments involving long-range tsunami propagation (in Sect. 5) indicate that low-order reconstructions (constant or linear) introduce excessive numerical diffusion, leading to significant attenuation of wave amplitudes. This motivates the use of higher-order reconstruction operators. Accordingly, the reconstructed variables are represented as piecewise polynomials over each cell Ω_j, given by:

$$\tilde{q}_j(x, y) = \overline{q}_j + p_j^q(x, y), \quad (24)$$

where $\overline{q}_j$ is the cell-averaged variable to be reconstructed, p_j^q the interpolating polynomial with coefficients derived from local geometry and neighboring variable cell-averaged values. This cell-wise approach allows tailored approximations, with stencil selection critical for accuracy and stability.

Figure 3 illustrates the stencil structure, where Ω_{j0} (red) is the reference cell, surrounded by first-order neighbors (blue) and second-order neighbors (green). For each Ω_{j0}, the stencils are defined as

$$\{\Omega_{j_0}, \Omega_{j_1}, \Omega_{j_2}, \Omega_{j_3}\}\ ;\ \{\Omega_{j_0}, \Omega_{j_1}, \Omega_{j_{11}}, \Omega_{j_{12}}\}\ ;$$
$$\{\Omega_{j_0}, \Omega_{j_2}, \Omega_{j_{21}}, \Omega_{j_{22}}\}\ ;\ \{\Omega_{j_0}, \Omega_{j_3}, \Omega_{j_{31}}, \Omega_{j_{32}}\}.$$

Linear reconstructions use only the first stencil, Ω_{ji}, whereas quadratic reconstructions – requiring two Gaussian points per edge – utilize the full stencil, Ω_{jkl}. The right panel of Fig. 3 illustrates the selection of these sub-stencils, where the barycenters of the selected cells are constrained to lie within cones defined by lines connecting the reference cell's barycenter to its vertices.

3.3.1 Linear Piecewise Reconstruction with Minmod Gradient Limiter

The linear reconstruction in Eq. (24) is expressed as

$$\tilde{q}_j(x, y) = \overline{q}_j + (q_x)_j (x - x_j) + (q_y)_j (y - y_j), \tag{25}$$

where $Dq_j = ((q_x)_j, (q_y)_j)$ denotes the numerical gradient.

The selection of this gradient defines the linear reconstruction. A variety of gradient estimation techniques exists in the literature, including classical limiter-based methods (Nessyahu and Tadmor, 1990; Sweby, 1984; Van Leer, 1997), FV formulations (Arminjon and St-Cyr, 2003; Christov and Popov, 2008; Jawahar and Kamath, 2000; LeVeque, 2002), and CU schemes for the Saint-Venant system (Bryson et al., 2011; Kurganov and Petrova, 2005).

In SWEpy, we follow Bryson et al. (2011) and construct three conservative linear polynomials $L^j_{k,l}(x, y)$ over the cell Ω_j and its neighboring pairs $\Omega_{j,k}$ and $\Omega_{j,l}$ (see Fig. 3). The candidate gradient is defined as

$$q'_j = \vartheta \,\mathrm{minmod}\left\{\nabla L^j_{k,l}\right\}, \quad \text{with} \quad \theta \in [1, 2]. \tag{26}$$

If substituting q'_j into Eq. (25) yields values at edge midpoints that exceed local extrema, the reconstruction is reduced to a constant state equal to the cell average; otherwise, the gradient is accepted, i.e., $Dq_j = q'_j$.

The minmod operator presented by Bryson et al. (2011), selects the admissible gradient with the smallest magnitude, thereby limiting spurious oscillations while preserving monotonicity. This ensures consistency with the source term discretization and supports well-balancing. However, it may introduce numerical diffusion in smooth regions, as observed in Sect. 5. A summary of the procedure is provided in Algorithm A1 in the Appendix.

3.3.2 Quadratic (WENO)

To reduce the numerical diffusion observed in low-order reconstructions – particularly in long-range wave propagation (e.g., tsunami simulations in Sect. 5) – we implement a quadratic weighted essentially non-oscillatory (WENO) reconstruction. The approach follows Zhu and Qiu (2018) and is adapted to unstructured triangular grids as in Sunder et al. (2021), while preserving the spatial constraints required for stability.

The reconstruction combines a quadratic polynomial p_0, obtained over the full stencil, with four linear polynomials $p_{k,j}$ $(k = 1, \ldots, 4)$ defined on sub-stencils:

$$\tilde{q}_j(x, y) = \frac{w_0}{\gamma_0}\left(p_0(x, y) - \sum_{k=1}^{4} \gamma_k p_{k,j}(x, y)\right) + \sum_{k=1}^{4} w_k p_{k,j}(x, y), \tag{27}$$

where the nonlinear weights w_0, w_k are computed from smoothness indicators β_0 (quadratic) and β_k (linear) according to $w_l = \overline{w}_l / (w_0 + \sum_{k=1}^{4} \overline{w}_k)$, with $\overline{w}_l = \gamma_l (1 + \tau/(\epsilon + \beta_l))$ and τ a correction term derived from the β_l values (Zhu and Qiu, 2018). The quadratic interpolant $p_{q,j}$ is constructed via a least-squares fit to the cell-averaged states over Ω_j and its first- and second-order neighbors, while preserving the mean value in Ω_j. Its implementation relies solely on geometric quantities (e.g., barycenters and area moments), avoiding numerical quadrature and improving efficiency, as detailed in (Fuenzalida Alarcón et al., 2025).

The stencil selection is performed efficiently using index-based searches on structured grids; more general geometric search strategies can be incorporated to relax grid constraints. The overall procedure is summarized in Algorithm A2, and is designed for efficient GPU parallelization within SWEpy.

This WENO reconstruction achieves high-order accuracy in smooth regions while limiting oscillations near sharp gradients, making it particularly suitable for far-field simulations where controlling numerical diffusion is critical without relying on Riemann solvers (Kurganov and Petrova, 2005).

3.4 Wet/dry fronts reconstruction

High-order reconstructions, while effective in reducing numerical diffusion in smooth regions, may produce nonphysical negative water depths near wet–dry interfaces, where the free surface intersects the bathymetry. To enforce positivity – i.e., $h \geq 0$ – we adopt the conservative correction of Bryson et al. (2011). When negative depths are detected, the reconstruction is replaced by a linear polynomial that preserves the cell average, thereby maintaining mass conservation.

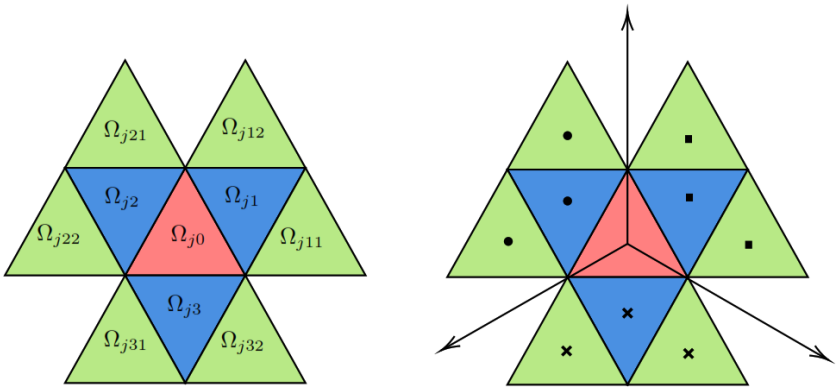

Figure 3. Stencil illustration for the jth cell (left) and its sectorial division (right). The blue triangles represent first-order neighbors Ω_{jk}, while the green triangles denote second-order neighbors Ω_{jkl}.

The correction depends on the number of dry vertices. If two vertices are dry, the reconstructed surface is defined by a plane passing through those vertices at bathymetric elevation and the cell barycenter at the mean surface level. If only one vertex is dry, the plane passes through the dry vertex at bathymetry, a neighboring wet vertex at an adjusted elevation, and the barycenter. In both cases, the plane is constructed to ensure $h = w - B \geq 0$ throughout the cell while preserving the mean value.

In practice, reverting locally to a first-order, positivity-preserving reconstruction at wet–dry fronts provides improved robustness with minimal impact on overall accuracy, since high-order reconstruction remains active away from these regions and governs wave propagation. A schematic illustration of this procedure is shown in Fig. 4.

This approach guarantees non-negative water depths across the domain, which is essential for stability in inundation problems such as the Conical Island test and the Malpasset dam-break case (Sect. 5). The correction procedure is summarized in Algorithm A3 and implemented efficiently using GPU-based parallelization.

While this method ensures positivity, it does not strictly preserve well-balancing near wet–dry fronts, where small imbalances may arise (Liu et al., 2018). This behavior was examined using the analytical solution of Synolakis for wave run-up[3], where the method demonstrates good accuracy in the wet–dry region. The results also reveal sensitivity to temporal discretization and the Courant number. A detailed analysis of these aspects is the subject of ongoing work, focusing on the numerical and theoretical properties of wet–dry reconstruction techniques and their suitability for different flow regimes.

[3]Refer to the Analytical Benchmarks section of the User Manual & Technical Reference (Meza et al., 2026).

3.5 Temporal discretization

Following the spatial discretization, the resulting semi-discrete system of Eq. (12) is integrated in time to ensure stability and accuracy across different flow regimes. In SWEpy, we employ both the explicit Euler (EE) method and a four-stage, third-order strong stability-preserving Runge–Kutta scheme (SSP RK4,3, hereafter RK4,3) described by Gottlieb et al. (2001).

For flows involving Manning friction, a semi-implicit treatment is adopted to improve stability while maintaining computational efficiency (Chertock et al., 2015). We define the discrete flux operator as

$$\mathcal{H}_j(\overline{\boldsymbol{q}}_j, \tilde{\boldsymbol{q}}_j) = -\frac{1}{|\Omega_j|}\sum_{k\in\mathcal{N}_j}(\mathcal{F}_{jk})l_{jk} + \overline{S}_{\mathrm{B}_j}, \tag{28}$$

where $\overline{\boldsymbol{q}}_j$ denotes the cell-averaged conserved variables and $\tilde{\boldsymbol{q}}_j$ their reconstructions.

The semi-implicit Runge–Kutta update, written in Shu–Osher form, reads

$$\begin{aligned}(\overline{\boldsymbol{q}}_j)^i = &\sum_{l=0}^{i-1}\alpha_{i,l}(\overline{\boldsymbol{q}}_j)^l + \frac{\Delta t}{2}\sum_{l=0}^{i-1}\beta_{i,l}\mathcal{H}_j^l \\ &+ \Delta t\,(\mathcal{G}(\overline{\boldsymbol{q}}_j))^{i-1}\begin{bmatrix}0 & (\overline{hu}_j^i) & (\overline{hv}_j^i)\end{bmatrix}^T, \\ &i = 1,2,3,4\end{aligned} \tag{29}$$

where $\overline{\boldsymbol{q}}_j^l$ are intermediate stage values, with $\overline{\boldsymbol{q}}_j^0 = \overline{\boldsymbol{q}}_j^{(n)}$ and $\overline{\boldsymbol{q}}_j^{(n+1)} = \overline{\boldsymbol{q}}_j^4$. The nonzero RK4,3 coefficients are

$$\begin{aligned}(\alpha_{1,0} \quad \alpha_{2,1} \quad \alpha_{3,0} \quad \alpha_{3,2} \quad \alpha_{4,3}) &= (1 \quad 1 \quad 2/3 \quad 1/3 \quad 1), \\ (\beta_{1,0} \quad \beta_{2,1} \quad \beta_{3,2} \quad \beta_{4,3}) &= (1 \quad 1 \quad 1/3 \quad 1).\end{aligned} \tag{30}$$

The EE scheme follows directly with $\alpha_{1,0} = 1$ and $\beta_{1,0} = 1$.

This semi-implicit formulation improves robustness in the presence of stiff source terms, particularly in friction-dominated regimes and near wet–dry interfaces, as demonstrated in the dam-break simulations (Sect. 5).

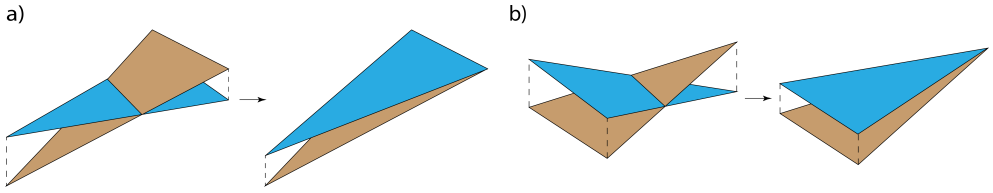

Figure 4. Schematic representation of the wet/dry treatment: **(a)** two dry points and **(b)** one dry point.

Time-step selection is governed by a Courant–Friedrichs–Lewy (CFL) condition,

$$\Delta t = \mathrm{CFL} \cdot \min_{jk} \frac{r_{jk}}{\max\{a_{jk}^{\mathrm{in}}, a_{jk}^{\mathrm{out}}\}}. \tag{31}$$

where r_{jk} is the perpendicular distance from edge jk to the opposite vertex, and CFL is user-defined. Theoretical stability bounds suggest $\mathrm{CFL} \leq 1/3$ for the CU scheme (Kurganov and Petrova, 2005), and $\mathrm{CFL} \leq 1/6$ when accounting for wet/dry reconstruction (Bryson et al., 2011) to guarantee positivity ($h = w - B \geq 0$). In practice, slightly larger values may be admissible. For RK4,3, Δt is computed at the first stage and appropriately scaled in subsequent stages to balance stability and efficiency.

4 Architecture & parallel structure

Having introduced the CU-fluxes, spatial reconstructions, and source-term discretizations in Sect. 3, we now describe their GPU implementation in `SWEpy`. The framework is designed to combine modularity, extensibility, and high-performance parallel execution. This enables efficient large-scale simulations while allowing users to incorporate additional physical processes – such as rheology or infiltration – within a consistent spatial and temporal discretization framework.

`SWEpy` adopts a modular programming paradigm (Parnas, 1972), in which the FV solver is decomposed into independent, reusable components. These modules handle different tasks, including grid loading, analysis configuration, preprocessing, time-step integration, spatial reconstruction, numerical flux, source term computations, and data output. This modular structure enhances user accessibility by isolating functionalities into self-contained units. It also promotes community-driven development through simple modification or extension of modules to accommodate additional source terms, boundary conditions, and other user-specific needs.

Although this design supports flexibility, the inclusion of new physical processes (e.g., infiltration or sediment transport) requires careful consideration to ensure consistency with the underlying numerical schemes and GPU-oriented implementation. An overview of the software architecture and its parallel workflow is presented in Fig. 5.

Developed in `Python`, `SWEpy` utilizes `CuPy`[4]. This approach enables significant speedups of parallel operations without requiring low-level programming in C++ or CUDA. Cell-centered variables (e.g., states and fluxes) are stored as indexed arrays – for example, the mean water level in cell Ω_3 is accessed as `Wj[3]` – allowing the solver to naturally exploit the data parallelism of FVM. Key operations, including reconstruction and flux evaluation, are vectorized over the entire mesh, minimizing CPU–GPU data transfers and leveraging efficient single-instruction, multiple-data (SIMD) execution (Harris et al., 2020). This design accelerates the CU scheme on unstructured triangular grids (Sect. 3) while maintaining scalability for large-scale applications, such as the Maule tsunami simulation (Sect. 5). As a result, `SWEpy` can deliver high-resolution solutions on consumer-grade hardware, in some cases achieving real-time or faster-than-real-time performance.

The execution workflow is organized into three stages – Preprocessing, Run Analysis, and Post-processing – summarized in Fig. 5. In the diagram, green boxes denote GPU-accelerated tasks, segmented boxes represent time-stepping and major phases, and arrows crossing green regions indicate CPU–GPU synchronization points, providing a conceptual map of data flow, parallelism, and modularity. The following sections describe each phase and its associated modules, with the primary functions highlighted in italics.

4.1 Preprocessing

4.1.1 Loading input data *(FileLoader.load_from_files)*

As shown in the upper portion of Fig. 5, the preprocessing phase initializes the simulation by organizing all input data for efficient GPU execution, reducing subsequent CPU–GPU transfers and enabling scalability on large unstructured grids.

`SWEpy` takes as input a user-defined triangular mesh – specified through vertex coordinates, connectivity, and bathymetry values – along with initial conditions for the conserved variables, boundary ghost-cell definitions, and a configuration file containing simulation parameters (e.g., numerical tolerances, final time, gravitational acceleration, CFL number, and optional source-term coefficients such as Manning roughness or Coriolis effects).

[4] A CUDA-enabled analogue of NumPy – to perform array-based computations directly on GPUs (Okuta et al., 2017).

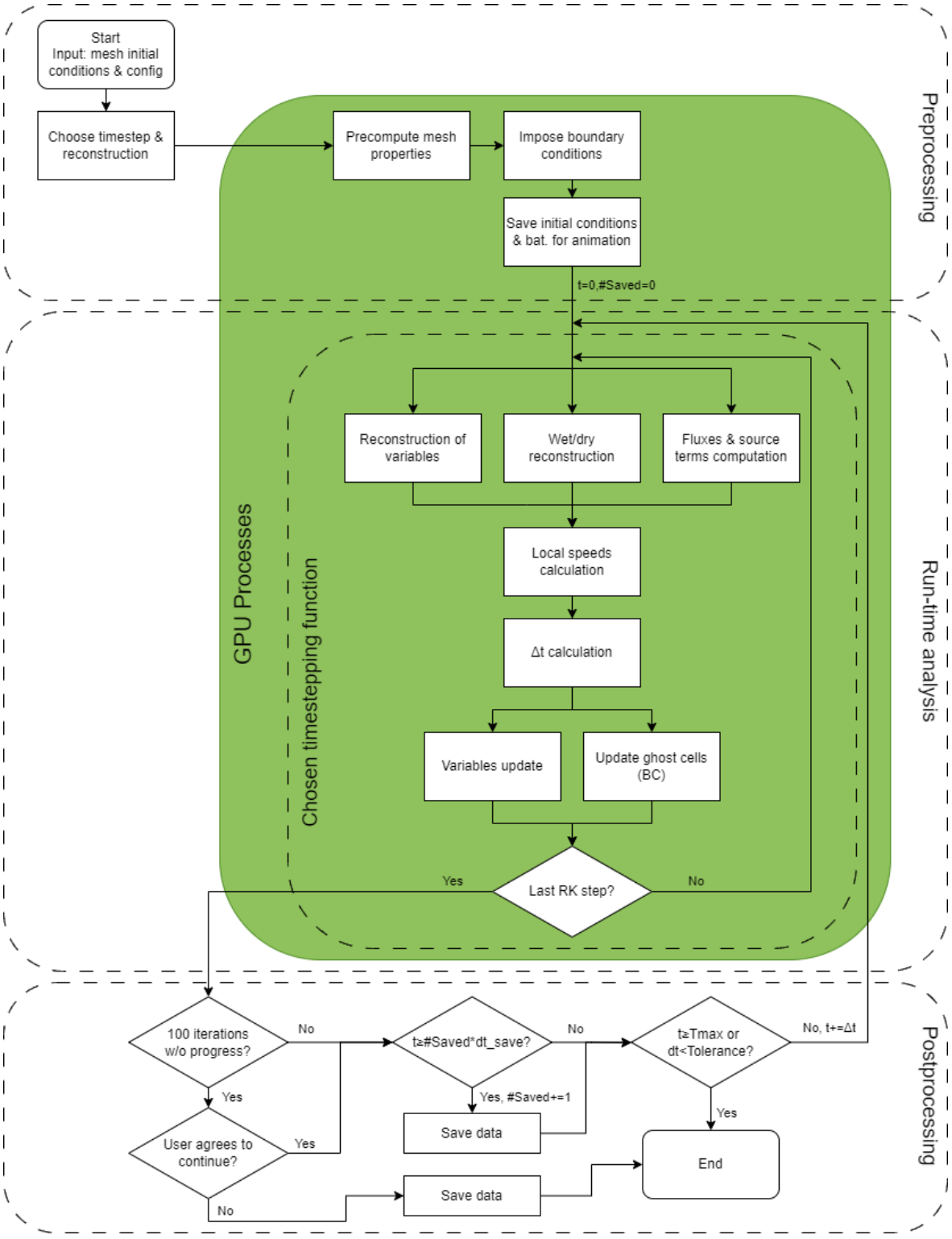

Figure 5. Overview of the SWEpy software parallel structure and architecture. The green box contains tasks performed on the GPU. The innermost segmented box contains tasks done by the chosen timestepping method. Outer segmented boxes indicate phases. Postprocessing phase highlights stagnation and divergence detection, and user controlled data saving. Arrows going into/out of green box indicate CPU-GPU synchronization. We highlight that the figure provides a high-level overview of module organization and data flow, rather than a function-level representation of the code structure.

All inputs are consolidated into the *mesh* dictionary, which serves as the central data structure throughout the simulation.

4.1.2 Bathymetry generation *(Utilities.bathymetry_midpoint/2)*

Bathymetry is reconstructed using an interpolator consistent with the chosen spatial reconstruction. For instance, a linear interpolator is employed with the minmod scheme (Bryson et al., 2011). This process involves interpolating vertex values to both edge Gaussian points and the cell interiors, which are required for source term evaluations in Eq. (20).

The interpolation relies on simple algebraic operations (such as computing the gradient of a plane defined by the three vertices of each triangle) and is vectorized for efficient execution on the GPU. Relevant quantities are precomputed and stored to further enhance performance.

4.1.3 Geometric computations on the mesh *(FileLoader.compute_element_properties)*

Key geometric properties required by the numerical scheme, such as cell areas $|\Omega_j|$, edge lengths l_{jk}, barycenters (x_j, y_j), perpendicular heights r_{jk}, and second area moments I_x, I_y, I_{xy} *(Utilities.second_moments*; see Fuenzalida Alarcón et al., 2025), are computed in parallel for all cells.

Using CuPy's vectorized array operations, these quantities are evaluated directly from vertex coordinates, enabling simultaneous computation across the entire grid on the GPU. This approach significantly accelerates the preprocessing stage, particularly for the high-resolution meshes typical of flood and tsunami simulations.

4.1.4 Stencil generation for high-order reconstruction *(Utilities.precomp_weno2)*

For quadratic WENO reconstruction, a vectorized procedure identifies first- and second-order neighbors in a single time, while flagging boundary cells with incomplete stencils for fallback to the minmod scheme. The least-squares matrices (right-hand side of Eq. A2), which depend only on grid geometry, are precomputed using CuPy's `linalg` routines. Matrix inversions and multiplications are performed in parallel over stacked systems, and the resulting coefficients are stored as arrays for efficient reuse during the run-analysis phase.

By shifting these computations to the GPU and avoiding repeated stencil assembly, the preprocessing establishes a data-parallel foundation, enabling `SWEpy` to handle complex simulations with minimized runtime delays.

4.2 Run-analysis *(ShallowWater.run/run_with_TS)*

The run-analysis phase – depicted by the central segmented box in Fig. 5 – forms the core of `SWEpy`'s time integration. In this stage, the solution is advanced iteratively through spatial reconstruction, source-term evaluation, flux computation, and state updates, all implemented to exploit GPU parallelism. By leveraging CuPy's array-based operations, these calculations are performed concurrently across the entire mesh, minimizing serial bottlenecks and enabling efficient simulation of large-scale flows, such as those presented in Sect. 5.

Prior to entering the main time-stepping loop, the program chooses the "timestep" function to be used according to user selection *(ShallowWater.choose_timestep)*. This timestep function is central as it defines the sequence of operations performed at each iteration – including reconstruction, flux evaluation, and source-term treatment – consistent with the CU formulation and the chosen temporal discretization. In essence, it encapsulates the definition of a single update step.

This design provides flexibility: by modifying or extending the timestep routines using the components described below, users can readily switch between reconstruction strategies, source-term models, time integration schemes, and even flux definitions.

4.2.1 Reconstruction *(PieceWiseReconstruction.minmod/weno2)*

For minmod reconstruction, `SWEpy` evaluates cell-centered gradients using neighboring average states and reconstructs the conserved variables $\boldsymbol{q}_j(M_{jk})$ at edge midpoints for flux computations in Eq. (16). Water surface values are also reconstructed at vertices to support wet/dry treatment. An optional diffusion parameter, $\vartheta \in [1, 2]$ (cf. Eq. 26, can be specified to control numerical dissipation: larger values reduce diffusion but may increase the risk of spurious oscillations.

All gradient evaluations and interpolations are implemented using CuPy's vectorized operations, exploiting SIMD parallelism to process all cells simultaneously on the GPU. As a result, the sequential loops described in Algorithm A1 are effectively replaced by fully parallel array-based computations, significantly improving performance.

For WENO reconstruction, `SWEpy` solves the LSQ linear systems in Eq. (A2) using precomputed matrices to obtain the reconstruction polynomials. These polynomials are then evaluated at the required quadrature points: edge Gaussian points for flux computations in Eq. (16), interior Gaussian points for source-term evaluation in Eq. (20), and cell vertices for wet/dry treatment.

Because the reconstruction stencils are preassembled during preprocessing, all evaluations can be carried out simultaneously across the mesh using GPU-accelerated operations. Consequently, the cell-wise iterations described in Algorithm A2 are fully vectorized and executed in parallel, significantly enhancing computational efficiency.

Boundary cells lacking full second-order neighbors, identified during preprocessing, default to minmod reconstruction. Wet/dry fronts are corrected by replacing invalid re-

constructions (negative depths) using CuPy's fancy indexing to locate and adjust affected cells in parallel, executing Algorithm A3 via SIMD commands on the GPU without explicit loops. This means that "for-loops" in Algorithm A3 are replaced by SIMD commands for all cells, efficiently performed over the GPU.

The modular design of SWEpy further allows users to define custom reconstruction operators, provided they supply interpolated values at the required points. This facilitates extensions – such as hybrid reconstruction strategies – while maintaining GPU efficiency.

4.2.2 Source Terms *(CentralUpwindMethod.source_term/2, coriolis, friction_term)*

Bathymetry source terms are evaluated in all cells using Eq. (20), based on the precomputed bathymetry reconstruction and the selected spatial operator. Additional contributions, such as Coriolis and friction terms, are computed from the cell states and stored for use during the update step.

All source-term evaluations are fully vectorized and executed on the GPU, ensuring efficient parallel computation, including for optional physics. The modular framework also allows users to incorporate additional source-term models, provided they remain consistent with the chosen spatial discretization, reconstruction strategy, and time-integration scheme.

4.2.3 Local speeds and time steps *(CentralUpwindMethod.one_sided_speed/2)*

Using the reconstructed states, velocities are desingularized and projected onto edge normals to evaluate the local propagation speeds in Eq. (19). These computations are performed simultaneously for all edges using GPU-parallel operations. When adaptive time stepping is enabled, the time increment Δt is determined from the CFL stability condition based on the computed wave speeds, maximizing the allowable step size to improve efficiency while limiting numerical diffusion (cf. Eq. 31).

Explicit Euler (EE) integration applies Eq. (16) through fully vectorized GPU operations, with CPU synchronization limited to advancing the timestep (one cycle in Fig. 5). The SSP RK(4,3) scheme (Gottlieb et al., 2010) and the classical four-stage RK4 method are implemented as sequences of scaled explicit Euler updates. These stages are executed sequentially on the GPU, avoiding iterative solvers and minimizing overhead, with optional friction corrections applied at each stage.

The modular design of the *timestep* function allows user-defined integration schemes to be seamlessly incorporated into the GPU workflow. This enables future extensions, such as predictor–corrector methods or modified Newton–Raphson iterations for implicit formulations, although the latter remain less suited to efficient GPU execution.

4.2.4 Variable update and correction (*timestep* function)

Fluxes are assembled from the reconstructed states and local wave speeds, and the cell averages are updated according to the CU scheme in Eq. (16) through a single GPU-parallel operation, followed by the enforcement of ghost-cell boundary conditions. For Manning friction, an intermediate semi-implicit correction is applied to the discharge, performed vector-wise across the grid.

In multi-stage RK schemes, intermediate states and fluxes are computed and stored on the GPU, with optional friction corrections applied at each stage. This design minimizes CPU–GPU synchronization while preserving the intended order of accuracy as represented in Table 2.

4.2.5 Boundary conditions *(BoundaryConditions.impose)*

The program enforces user-defined boundary conditions through the ghost cell states. Because each ghost cell is independent, mixed boundary conditions along domain boundaries can be applied simultaneously. The modular structure of the code further supports the implementation of more advanced boundary treatments, such as transport-based models for coupling with external solvers, commonly referred to as sequential or iterative coupling.

The boundary conditions currently implemented through ghost cells include: (a) a zero-order extrapolation permeable (soft) boundary, in which the border cell state is replicated in the adjacent ghost cell; and (b) an impermeable (wall) boundary, where the water height is replicated while the flow direction is reversed, effectively reflecting the flux and enforcing a zero-normal-flow condition at the interface.

We note that periodic boundary conditions can be incorporated by defining boundary cells as neighbors of other boundary cells. For example, to impose periodicity between the top and bottom boundaries of a channel, the bottom cells are treated as the upward neighbors of the top boundary cells, and vice versa. This approach is used in several of our experiments. Please refer to the User Manual & Technical Reference (Meza et al., 2026) for details.

4.3 Postprocessing *(FileSaver module)*

The post-processing phase, illustrated in the lower portion of Fig. 5, concludes the simulation by handling data output and termination criteria. Leveraging the modular design of SWEpy, this stage supports customizable workflows tailored to both research analysis and operational monitoring. Users can integrate predefined or custom routines to export results at any point during runtime, enabling both real-time inspection and efficient post-simulation analysis.

For data storage, SWEpy provides built-in functionality to save initial conditions and bathymetry *(FileSaver.save_bathymetry)* in .vtk format prior to the simulation, followed by solution snapshots at user-defined intervals Δt_{save} throughout the run *(FileSaver.save_animation)*. This format is optimized for visualization tools such as ParaView (Ayachit, 2015), allowing users to render both static fields and time-resolved animations over unstructured grids. In addition, time series data at selected locations (e.g., virtual gauges for tsunami validation; see Sect. 5) can be recorded at the same Δt_{save} *(FileSaver.save_TS)*. Because these routines are integrated directly into the runtime, they enable continuous monitoring of the evolving solution – an important feature for identifying anomalies in long simulations without interrupting execution. Efficient data handling is ensured through CuPy's GPU-compatible NumPy routines (e.g., save, savez, savetxt), which allow high-frequency output with minimal performance overhead.

By default, the simulation terminates once the prescribed final time is reached. However, the modular framework allows for additional user-defined stopping criteria, such as divergence detection (e.g., when an adaptive Δt falls below a threshold) or stagnation monitoring (e.g., negligible state evolution over a specified number of iterations). These criteria can be implemented by extending or modifying the *run* functions. In our experiments, such controls proved effective in preventing unproductive simulations and complemented the live visualization capabilities for real-time decision-making. More broadly, this flexibility enhances SWEpy's applicability, enabling integration with external tools for automated error handling or real-time coupling in hybrid modeling workflows.

5 Results and Discussion

This section presents a series of numerical experiments to validate SWEpy's implementation, accuracy, and performance against analytical benchmarks and real-world cases, demonstrating the effectiveness of the CU scheme with WENO reconstructions and GPU acceleration as described in Sects. 3 and 4. We begin with canonical tests that assess spatial and temporal accuracy, well-balancing, positivity preservation, and numerical diffusion reduction, followed by synthetic scenarios that highlight the versatility of the reconstruction operators. Finally, large-scale simulations of the 1959 Malpasset dam failure and the 2010 Maule tsunami assess real-world applicability by comparing results with historical data and established solvers such as TELEMAC (Moulinec et al., 2011). These experiments underscore SWEpy's robustness for inundation modeling over complex topography and long-range wave propagation, achieving high-resolution outcomes on consumer hardware with computation times reduced by factors of up to 21× via CuPy parallelism.

5.1 Benchmark tests

5.1.1 Spatial convergence order study

To validate SWEpy's spatial accuracy and verify the correct discretization of the bathymetry source term (Eq. 20), we replicate the convergence test of Bryson et al. (2011), focusing on scenarios where well-balancing is critical to preserve steady states over non-flat topography free of spurious oscillations. The test quantifies the order of convergence for different spatial reconstruction operators (Sect. 3.3) over a smooth Gaussian bump.

The computational domain is a $2 \times 1\,\text{m}$ rectangle discretized into a regular mesh of equilateral triangles, where triangles along the top and bottom boundaries are halved to ensure consistent boundary treatment. The bottom topography is defined as

$$B(x, y) = 0.5 \exp\big(-25(x-1)^2 - 50(y-0.5)^2\big). \tag{32}$$

The initial conditions consist of a uniform free-surface elevation $w(x, y, 0) = 1.0\,\text{m}$ and a velocity field $u(x, y, 0) = 0.3\,\text{m s}^{-1}$, $v(x, y, 0) = 0\,\text{m s}^{-1}$. Fully permeable (zero-gradient) boundary conditions are applied on all sides, with $g = 1\,\text{m s}^{-2}$. The flow evolves to a steady, non-uniform state by $t \approx 0.07\,\text{s}$, at which point temporal errors become negligible and spatial errors dominate.

The reference solution is computed on a fine grid with $n_x = 512$ horizontal divisions (approximately 1.18×10^6 cells) at $t = 0.07$ s. Figure 6 illustrates the Gaussian bump on a coarse grid ($n_x = 32$) and the corresponding reference solution. The L^2 error is defined as

$$\|e\|_2 = \sqrt{\sum_j |\Omega_j| \left(\overline{w}_j - w_{\text{ref},j}\right)^2},$$

Convergence orders are then estimated via successive grid refinements. Table 2 reports errors and orders for grids $p = 0$ to 3 ($n_x = 32 \cdot 2^p$) across combinations of reconstruction operator and time-stepping scheme, while Fig. 7 shows the log–log relationship between error and effective grid spacing Δx, with fitted power laws consistent with the expected convergence rates.

The results confirm the robustness of all configurations. Constant reconstruction yields better-than-first-order accuracy, while linear and quadratic reconstructions approach second-order convergence, in agreement with the scheme's formal order for smooth solutions.

The theoretically attainable third-order accuracy with quadratic reconstruction is not achieved, suggesting that further refinement of either the numerical flux formulation or the bathymetry source term discretization may be needed to fully realize higher-degree polynomial benefits. This order degradation is an expected behavior, also reported by Bryson et al. (2011), and more recently by Nguyen (2023). However, a detailed convergence analysis of the extended scheme is

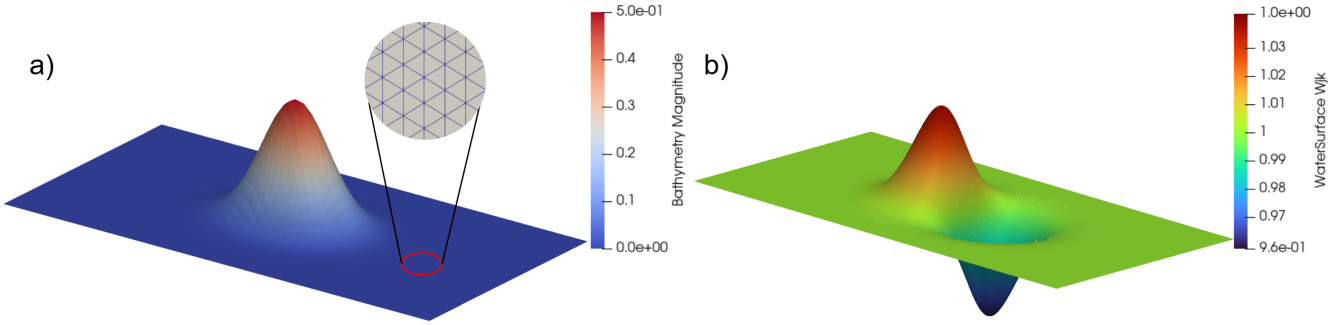

Figure 6. Bottom topography on a $n_x = 32$ point grid (left) and reference solution (right) for the Gauss bump. The zoomed-in circular window highlights the grid structure pattern.

Table 2. L^2 errors and numerical orders of accuracy. Grid number p corresponds to $nx = 32 \cdot 2^p$ horizontal divisions of the domain.

Grid	Explicit Euler – Const.		Explicit Euler – Lin.		RK4,3 – Lin.	
	L^2Err	Ord	L^2 Err	Ord	L^2 Err	Ord
$p = 0$	1.7×10^{-3}	–	6.4×10^{-4}	–	6.4×10^{-4}	–
1	7.6×10^{-4}	1.15	1.6×10^{-4}	1.91	1.6×10^{-4}	1.90
2	3.1×10^{-4}	1.28	3.9×10^{-5}	1.63	3.8×10^{-5}	1.78
3	1.2×10^{-4}	1.43	9.6×10^{-6}	2.01	8.8×10^{-6}	2.12
Grid	Explicit Euler – Quad		RK4,3 – Quad.		RK4 – Quad.	
	L^2 Err	Ord	L^2 Err	Ord	L^2 Err	Ord
$p = 0$	2.9×10^{-4}	–	2.9×10^{-4}	–	2.9×10^{-4}	–
1	6.7×10^{-5}	1.90	6.6×10^{-5}	1.90	6.6×10^{-5}	2.13
2	1.6×10^{-5}	1.78	1.5×10^{-5}	1.78	1.5×10^{-5}	2.11
3	3.9×10^{-6}	2.02	3.6×10^{-6}	2.12	3.5×10^{-6}	2.14

beyond the scope of this work. Nevertheless, WENO-based quadratic reconstruction consistently produces the smallest errors, outperforming lower-order approaches across all resolutions. This improved accuracy is particularly relevant in precision-critical scenarios, where error propagation over long timescales can be significant.

From a practical performance standpoint, generating the fine-grid reference solution with explicit Euler time-stepping and linear reconstruction required approximately 5 min wall-clock time on consumer-grade GPU hardware, including output at a high-frequency interval of 0.01 s for animation purposes. This demonstrates that SWEpy can deliver high-resolution, well-balanced solutions for flows over smooth-bottom at modest computational cost.

5.1.2 Well-balancing test

We simulate a static flat water surface over white-noise-generated bathymetry to verify the well-balanced property for all reconstruction operators. The domain is a $1 \times 1\,\mathrm{m}^2$ domain, with a random bathymetry with mean depth of $d = 2$, uniformly distributed in $[-1.1d; -0.9d]$, discretized using a right-triangles mesh with $\Delta x = \Delta y = 0.01$. The dry tolerance is set to 10^{-17} to assess machine-precision accuracy. Simulations use explicit Euler time-stepping with an adaptive timestep and CFL $= 0.25$.

The initial water surface is set to 0 m and the velocity field is set to zero in both directions. Figure 8 shows the random bathymetry and the maximum water height over time for the three reconstructions, confirming that no spurious oscillations arise. These results, while potentially surprising, are in fact expected since the bathymetry source term discretization is exactly well balanced by construction. This means that even in the presence of boundary conditions, unphysical oscillations or flows will not appear. Notably, this experiment also confirms the well-balancing property with the Runge Kutta scheme, since it is implemented as four successive explicit Euler steps.

5.1.3 Conical island wetting–drying benchmark

To evaluate SWEpy's wet/dry handling and reconstruction performance during wave-obstacle interactions, which are essential for coastal inundation modeling, we simulate the

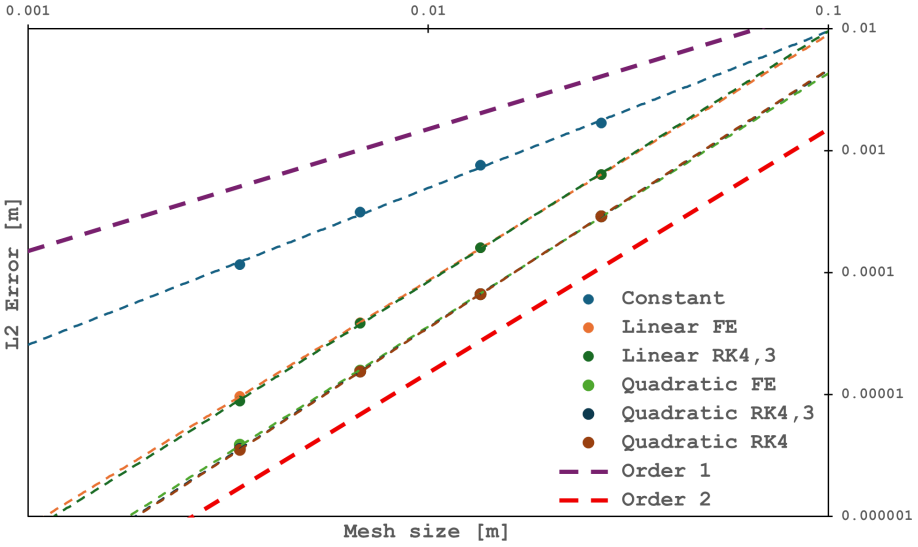

Figure 7. Log-log plot of L^2 error versus grid size Δx, with fitted power-law curves indicating convergence orders.

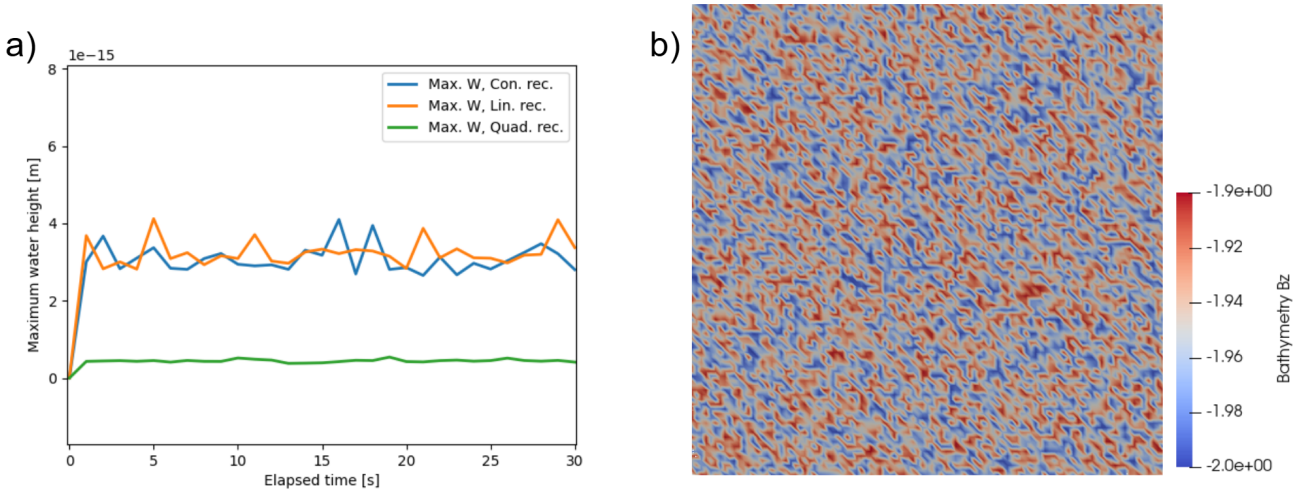

Figure 8. **(a)** Maximum water height over time. **(b)** Randomly generated bathymetry.

conical island benchmark, a laboratory experiment (Briggs et al., 1995) widely used for SWE validation (Synolakis et al., 2008). This test assesses (i) positivity preservation on sloped topography, where runup/rundown induces dynamic wetting without unphysical negative water depths, and (ii) operator fidelity in capturing complex wavefronts. The domain is a 41×30 m tank with permeable (soft) boundaries to absorb reflections, thereby minimizing boundary artifacts. The domain is discretized using an equilateral triangular mesh with edge length 0.5 m (approximately 12 000 cells) , ensuring spatial isotropy. The bathymetry consists of a flat bottom with a truncated cone (base diameter 7.2 m, crest diameter 2.2 m, height 0.625 m) centered at the origin, simulating an island. The initial free-surface elevation is given by the solitary-wave profile:

$$w(x, y, 0) = \frac{H}{d} \operatorname{sech}^2\big(\gamma\,(x - X_1)\big), \tag{33}$$

with $d = 0.320$ m, $H = 0.02976$ m, and $\gamma = \sqrt{3H/4d^3}$, positioned at $X_1 = -13$ m. The initial velocity field is

$$u(x, y, 0) = \frac{g}{d} w(x, y, 0) \left(1 - 0.25 \frac{w(x, y, 0)}{d}\right),$$
$$v(x, y, 0) = 0, \tag{34}$$

The initial condition is derived to ensure consistent propagation and extruded uniformly in the y-direction to simulate a two-dimensional wave front. Simulations are conducted with constant, minmod, and WENO reconstruction operators

for cross-comparison, employing an explicit Euler time integration scheme with a Courant-Friedrichs-Lewy (CFL) number of 0.25. The reason for this temporal integration setup is twofold: (1) it demonstrates the solver's capability to operate at higher effective CFL numbers, and (2) it is sufficiently low to keep oscillations controlled, thereby isolating spatial-reconstruction effects and enabling a focused assessment of how each operator handles wet/dry transitions and wavefront modeling as the soliton interacts with the cone, as shown in the initial setup of Fig. 9.

Figure 10 shows the free-surface elevation at each computational element (W_j) for each reconstruction operator at $t = 7$ s, as the leading wave passes gauge 16. This snapshot provides a spatial context for the subsequent time-series comparison, highlighting differences in wavefront sharpness and height at the gauge location. The WENO reconstruction yields the sharpest and highest wave at gauge 16 ($\eta \approx 0.041$ m), the minmod reconstruction shows moderate attenuation ($\eta \approx 0.03$ m), and the constant reconstruction produces a visibly diffused wavefront ($\eta \approx 0.011$ m). Green markers indicate the positions of gauges 1, 6, 16, and 22 (from left to right), consistent with the original experimental layout for direct validation of runup dynamics.

We note that, due to the nature of the wet/dry reconstruction procedure, minimal numerical artifacts arise near the wet/dry interface in the form of "wet" triangles with very small water column heights (approximately 10^{-5}–10^{-4} m). This effect can be further mitigated by adjusting the dry tolerance parameter in the simulation configuration file or by grid refinement, as shown in Fig. 11: The numerical artefacts are significantly reduced in the finer grid, with water column heights near the interface of order 10^{-10}–10^{-9} m.

Figure 12 presents time-series of water surface elevation for gauges 1, 6, 16, and 22 using the constant, minmod, and WENO reconstruction operators. The laboratory records, originally timed from wavemaker initiation, have been shifted by -25 s to align the incident wave arrival time with simulations. This value corresponds to the estimated paddle deceleration inferred from signal traces in Briggs et al. (1995).

Across gauges, WENO accurately reproduces main crest height and arrival time with minimal phase error, while constant and minmod underestimate secondary oscillations and exhibit greater dissipation during rundown. Quantitatively, the L^2 norm of differences between numerical and experimental series (Table 3) confirms that WENO and minmod exhibit similar performance, with WENO showing slightly higher averaged errors over the four gauges ($\overline{|e|}_2^{\text{WENO}} = 0.0321$ vs. $\overline{|e|}_2^{\text{minmod}} = 0.0320$), while both outperform constant by approximately 40 % ($\overline{|e|}_2^{\text{constant}} = 0.0533$).

Since L^2 is phase-sensitive and the phase shift cannot be determined exactly, we repeated the analysis by applying uniform shifts from -0.10 to $+0.30$ s to the numerical series (Table 4). Across all tested shifts, WENO consistently achieved the lowest error scores, confirming its robustness to phase uncertainty and its suitability for dynamic wetting–drying reconstruction in coastal flows.

Table 3. L^2 error of time series for each reconstructor at the different gauges.

Rec.	Gauge 1	Gauge 6	Gauge 16	Gauge 22
Constant	0.0302	0.0373	0.1014	0.0443
Minmod	0.0254	0.0288	0.0325	0.0412
WENO	0.0262	0.0296	0.0316	0.0409

Figure 13 illustrates spatial diffusion effects by comparing the free-surface elevation at $t = 13$ s across the three reconstruction operators, at a stage where the wave has traversed the island and formed the cardioid-shaped crest pattern observed in laboratory measurements. The WENO reconstruction (panel c) preserves sharp gradients, with minimal attenuation and good retention of intricate structures. By contrast, the minmod reconstruction (panel b) maintains the overall pattern but introduces noticeable smoothing, while the constant reconstruction (panel a) dissipates most fine-scale features, resulting in a blurred profile. To quantify these differences, we computed the depth-integrated potential energy $\mathcal{E} = \rho g \int_{\Omega_{\text{strip}}} \eta^2, \mathrm{d}\Omega$ over a vertical strip from $x_l = 9.5$ m to $x_r = 14.5$ m downstream, where $\eta = w$ since the still-water level is zero in the solitary-wave setup. The energies are $\mathcal{E}_{\text{const}} = 0.015$, $\mathcal{E}_{\text{minmod}} = 0.020$, and $\mathcal{E}_{\text{WENO}} = 0.044$, confirming WENO's superior wave energy retention in the post-interaction field. Combined with the L^2 analysis, these findings underscore WENO's superior balance between low numerical diffusion and faithful waveform reproduction, whereas the constant reconstruction underperforms in both aspects.

In this work, we use the energy-based measure $\mathcal{E}$ to quantify the extent of numerical diffusion. This metric, together with the comparisons at measurement stations, provides sufficient evidence to support the conclusions regarding WENO's superior performance in capturing wave dynamics. As noted above, point measurements capture only one aspect of the numerical result, being useful for specific events such as wavefront arrival time, zones of maximum amplitudes, or inundation extent, among others. In addition, Fig. 13 shows a plan view to emphasize the importance of a global approach to the numerical results and their relation to the physical problem analyzed.

In that direction, and building on the finding that the WENO-based approach captures relevant information for global analysis, Fig. 14 provides a detailed sequence of the wavefront evolution under WENO reconstruction, illustrating how it maintains sharp gradients and structural integrity throughout the interaction at $t = 7$, 8, and 9 s. The leading crest propagates toward the island, splits, and wraps around its flanks, producing a clear diffraction pattern. In the lee,

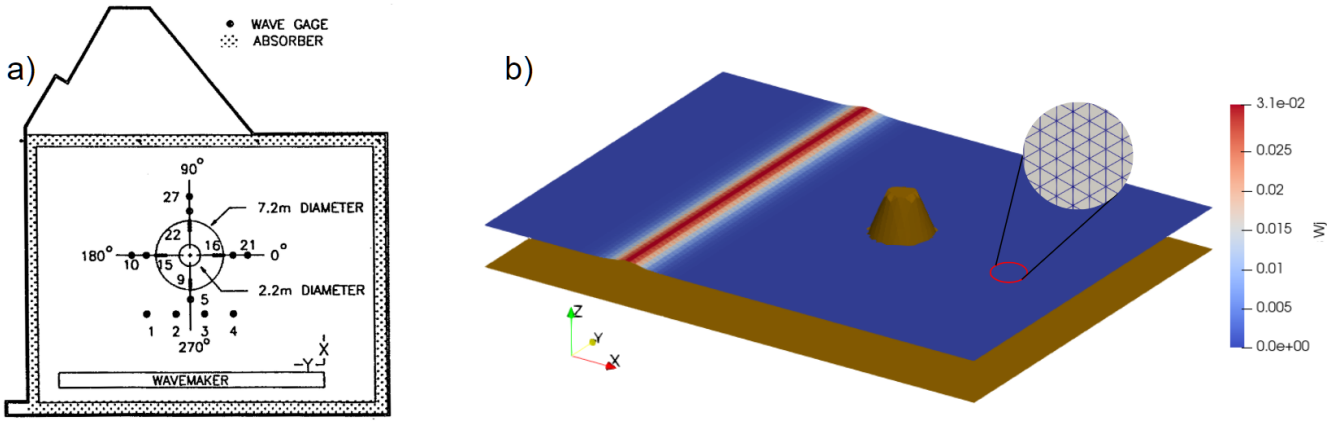

Figure 9. **(a)** Configuration of original experiment (digitized from Briggs et al., 1995) for the conical island benchmark. **(b)** 3D view illustrating the solitary wave profile approaching the truncated cone.

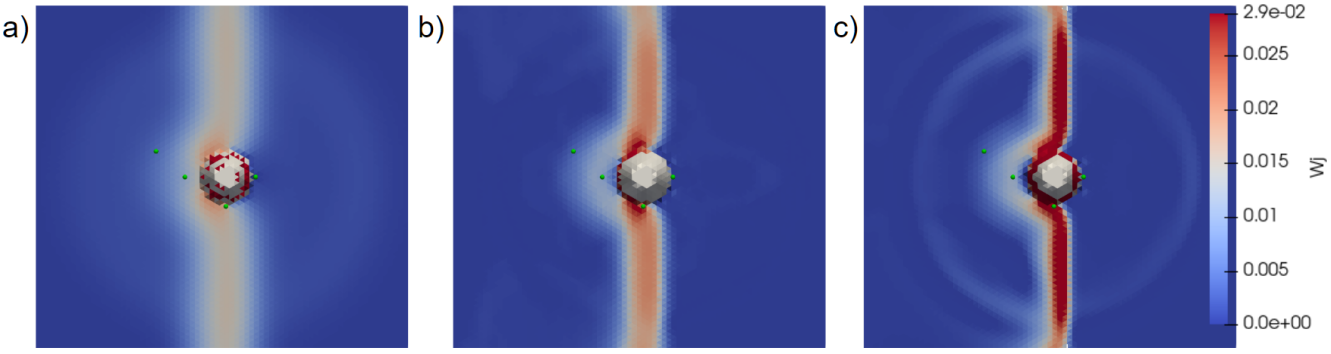

Figure 10. Comparison of wave structure while passing through gauge 16. Free-surface elevation at $t = 7$ s for **(a)** constant, **(b)** minmod, and **(c)** WENO reconstruction operators. Green markers indicate the positions of gauges 1, 6, 16, and 22 (from left to right).

the opposing wavefronts converge and form a coherent cusp that advances shoreward. A small, nearly circular secondary wave is visible behind the main crest; this is a residual artifact of the wetting–drying correction process, but it decays rapidly and does not trigger further spurious oscillations. For the whole simulation, the shoreline evolves smoothly, and the wavefront retains sharp gradients throughout the interaction, even in the presence of zeroth-order boundary conditions. Thus, the wavefront sequence in Fig. 14 illustrates that SWEpy's implementation incorporates the appropriate numerical foundations, enabling detailed descriptions of more complex real-world phenomena.

Overall, the conical-island benchmark confirms that SWEpy reproduces the key hydrodynamic processes involved in wave–obstacle interaction, including runup and rundown on sloping topography, diffraction around an emergent feature, and convergence in the lee. The comparison with laboratory measurements demonstrates that the WENO reconstruction consistently achieves the most faithful representation of surface elevation amplitude, phase, and post-interaction structure, while maintaining stability at wetting–drying fronts, and ensuring positivity preservation. Although small discrepancies remain–particularly in negative water levels following the first rundown–these can be attributed to physical processes not represented in the depth-averaged SWE framework (e.g., vertical accelerations) and to uncertainties in the temporal alignment between the laboratory and numerical time series. Taken together with the other validation cases, these results highlight the model's capability to resolve complex nearshore hydrodynamics with high numerical fidelity, especially when paired with high-order reconstruction.

5.1.4 Computational Performance and Scalability

To assess SWEpy's computational efficiency, we performed scaling benchmarks from coarse to fine resolutions using the Conical Island and Gaussian Bump test cases introduced previously. Table 5 summarizes execution times on an NVIDIA GeForce GTX 1650 GPU versus a serial single-core CPU implementation on an Intel Core i5-10300H.

The results illustrate a characteristic performance transition between latency-bound and throughput-bound regimes typical of GPU-accelerated frameworks. For small meshes

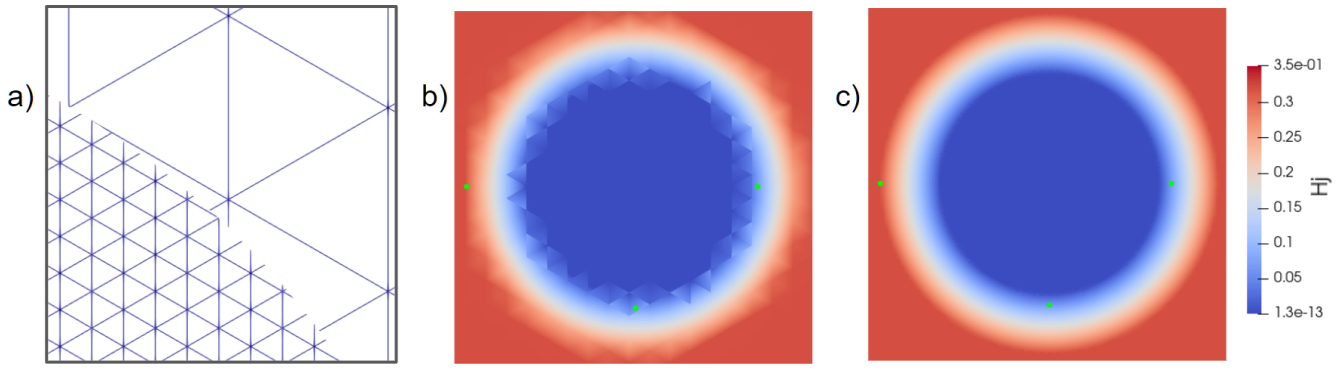

Figure 11. **(a)** Grid size comparison closeup. Water column heights (Hj) around protrusion of conical island diminish from the **(b)** coarse grid to the **(c)** fine grid.

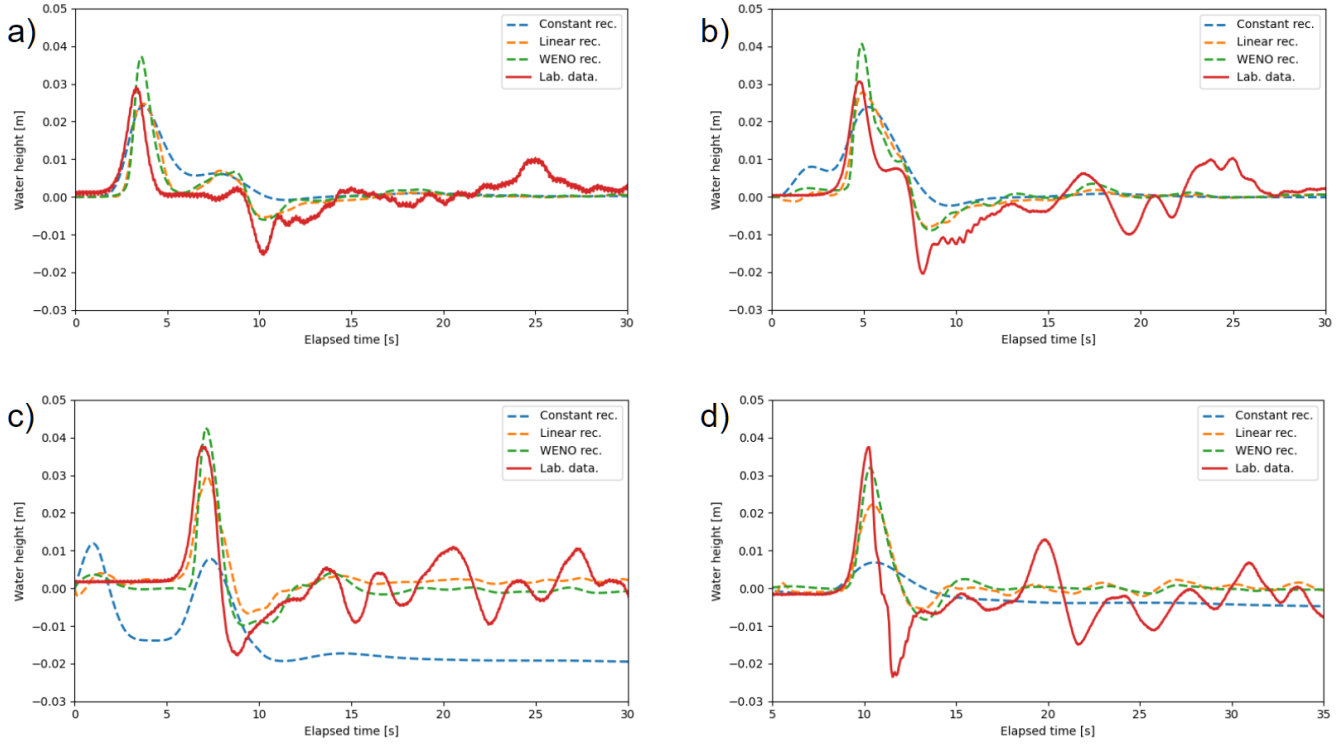

Figure 12. Time series of water heights at gauges 1 **(a)**, 6 **(b)**, 16 **(c)**, and 22 **(d)**. Solid line is laboratory data, and dashed lines are SWEpy's solutions.

(e.g., Conical Island coarse, ∼3000 elements), the GPU execution time is dominated by fixed overheads–kernel launch latencies and host-device synchronization–resulting in negligible or even fractional speedups relative to the CPU. In this regime, the serial CPU implementation is competitive due to its lack of launch overhead.

However, scaling behavior improves dramatically with problem size. As the element count increases, the massive parallelism of the GPU is more effectively utilized. In the Gaussian Bump case with ∼1.2 million elements, the solver transitions into a throughput-bound regime, achieving a speedup of 26.2×. This confirms that SWEpy's architecture is well-suited for high-resolution scenarios where the arithmetic intensity of WENO reconstruction saturates the GPU's CUDA cores.

It is worth noting that, to maintain a fair comparison with respect to code structure and accessibility, the CPU implementation was developed as a direct port of the GPU version. This design choice results in worse-than-linear time scaling with problem size, as seen in Table 5, and has two primary causes. First, both implementations follow a deliberately straightforward computational structure to preserve user accessibility. Second, the wet/dry algorithm relies on comparison and indexing operations (`np.where`,

Table 4. L^2 average error for each reconstructor with different time shifts of laboratory data to account for phase error.

Rec.	+0.30	+0.25	+0.20	+0.15	+0.10	+0.05	−0.05	−0.10
Constant	0.0516	0.0518	0.0520	0.0523	0.0526	0.0529	0.0537	0.0540
Minmod	0.0285	0.0288	0.0292	0.0298	0.0304	0.0312	0.0329	0.0338
WENO	0.0274	0.0277	0.0282	0.0289	0.0298	0.0309	0.0334	0.0348

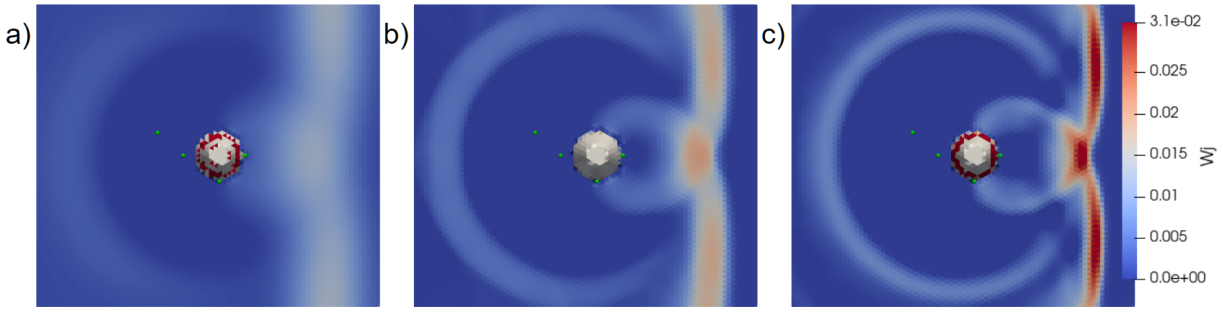

Figure 13. Diffusion study. Solution at time $t = 13$ s for constant **(a)**, linear **(b)**, and quadratic **(c)** reconstructors. Green marks are the locations of gauges 1, 6, 16, and 22 (from left to right).

`np.count_nonzero`, and fancy indexing) over arrays that may not be in contiguous memory layout due to transposition and stacking; resolving this would require restructuring the code in ways that would break the structural similarity between the CPU and GPU versions, making the comparison less fair. As a consequence, the reported speedup values should be interpreted with care: they reflect the advantage of the GPU implementation over this specific unoptimized serial baseline.

Furthermore, system traces reveal the hardware utilization patterns across different scales. As shown in Table 6, GPU throughput – measured as SM occupancy, i.e., the average percentage of warps active in streaming multiprocessors during runtime – rises from 22.9 % at coarse resolution to 91.1 % at fine resolution in the Gaussian Bump benchmark, confirming effective saturation of the computing resources. In the Conical Island case, however, the wetting–drying correction procedure combined with the extended simulation duration significantly increases the number of GPU kernel launches without a proportional gain in throughput. This indicates that the high density of kernel launches throttles the efficient use of computational resources, suggesting that CuPy optimizations such as kernel fusion or element-wise kernels for arithmetic-intensive operations could substantially improve performance.

In terms of memory usage, the software exhibits a predictable VRAM utilization behavior, measured as the ratio of allocated to total available memory, that increases with mesh size. However, memory usage does not depend exclusively on mesh size. As shown in Table 6, the fine-resolution Conical Island grid has fewer elements than the coarse Gaussian Bump grid, yet exhibits higher memory utilization, owing to the additional memory demands of the active wetting-drying handling algorithm.

In addition, we quantified the cost of adaptive time-stepping. Profiling on the finest mesh showed that computing the adaptive Δt accounts for approximately 3 % of the total runtime (3.85 s out of 142.1 s total for the Gaussian bump run). In contrast, imposing a fixed conservative time-step to ensure stability increased the total runtime to 148.3 s. Thus, the computational overhead of adaptive time-step control is negligible compared to the efficiency gains from maximizing the stable time-step size.

The performance analysis presented in this subsection characterizes how solver efficiency varies with problem size on a fixed single-GPU device. Unlike formal strong or weak scaling studies, which assess performance as the number of processing units or available memory varies, this analysis captures the transition from latency-bound to throughput-bound execution as the workload grows to saturate the available hardware. A multi-device scaling study remains an important avenue for future work.

5.2 Real-life scenario 1: Malpasset Dam failure

To evaluate `SWEpy`'s performance in realistic inundation scenarios involving complex topography and moving wet/dry fronts, we reproduce the 1959 Malpasset dam failure on France's Reyran River. The event is characterized by rapid flooding over highly irregular terrain (Moulinec et al., 2011). This case is used to validate the model's positivity-preserving reconstruction schemes, treatment of bathymetric source terms, and semi-implicit friction formulation described in Sect. 3.

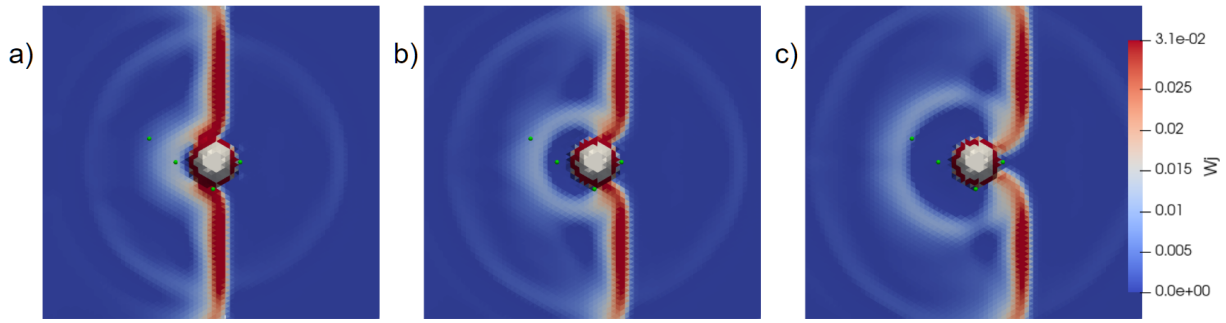

Figure 14. Wave-obstacle interaction study. Solution at times $t = 7$ s **(a)**, $t = 8$ s **(b)**, and $t = 9$ s **(c)**, using the WENO reconstruction operator. Green marks are the locations of gauges 1, 6, 16, and 22 (from left to right).

Table 5. Average execution time per time-step for idealized benchmarks (NVIDIA GTX 1650 vs. Serial Intel Core i5-10300H CPU). Speedups are relative to the single-core CPU baseline.

Test	Resolution	#Elements	GPU Time/step (s)	CPU Time/step (s)	Speedup
C. Island	Coarse	2958	1.03	0.076	0.07×
	Medium	11 682	1.03	0.411	0.40×
	Fine	45 924	1.04	2.579	2.47×
G. Bump	Coarse	18 992	0.96	0.73	0.76×
	Medium	74 848	1.22	8.39	6.90×
	Fine	1 184 128	8.37	219.29	26.20×

The computational domain is discretized using an unstructured triangular grid adapted from the TELEMAC-2D validation dataset. The mesh contains 26 000 elements, with characteristic triangle heights Δr (measured from a vertex to the opposing side) ranging from 4.01 to 401.95 m, and an average value of 40.28 m.

The bathymetric and topographic data are derived from the 1931 IGN Saint-Tropez map, with additional refinement upstream of the dam to better resolve steep gradients (Fig. 15a, inset). The initial condition sets the reservoir water surface to an elevation of 100 m upstream of the dam, represented as a vertical plane between coordinates (4701.18, 4143.10) and (4655.50, 4392.10), and 0 m elsewhere, with all cells above sea level initialized as dry (Fig. 15b). Boundary conditions are specified as impermeable walls, consistent with the TELEMAC reference configuration. The Manning roughness coefficient is set to $n = 0.03$ to match the TELEMAC reference setup. The simulation employs minmod reconstruction and adaptive time-stepping with a Courant–Friedrichs–Lewy (CFL) number of 0.33, and is run until $t_{\max} = 4000$ s.

Twelve virtual gauge locations are defined along the river valley to monitor the flood wave progression (Fig. 15b). Three gauges (transformers A, B, and C) are used to measure wave arrival times, while nine gauges (P6–P14) record maximum water heights from 1 : 400 scale laboratory experiments by Electricité de France. The simulated values of arrival times and peak heights are reported in Table 7 alongside the corresponding experimental data and results from TELEMAC's HLLC solver, widely regarded as the most accurate scheme for this benchmark.

This benchmark provides a rigorous test of SWEpy's ability to handle complex bathymetry, apply semi-implicit friction for roughness effects, and accurately treat wetting–drying fronts in initially dry cells. While SWEpy exhibits notable discrepancies at certain locations, its relative errors are, in most cases, nearly an order of magnitude smaller than those produced by TELEMAC's FV solver. As noted in TELEMAC's validation guide, these discrepancies may stem from several factors: (i) measurement uncertainty in the 1 : 400 scale physical model, (ii) omission of debris transport and sediment dynamics, and (iii) the fact that the dam breach was not truly instantaneous. In addition, the combination of rapidly varying topography and highly curved flow paths may violate the underlying assumptions of the SWE, further contributing to discrepancies between simulated and observed data.

TELEMAC's accuracy improves with increasing distance from the dam, suggesting that numerical diffusion is a significant factor in the model and may help explain its closer agreement in arrival-time estimates. Point P13 stands out as a pronounced outlier for both solvers. Its location within a poorly resolved section of the inner riverbank likely contributes to the large discrepancy in the predicted maximum height. To support this hypothesis, nearby points located outside the riverbank record simulated water heights between 4 and 8 m, values that align more closely with the observed

Table 6. GPU performance metrics derived from system traces for the Conical Island and Gaussian Bump benchmarks.

Test	Resolution	#Elements	GPU Time (s)	GPU Ops	GPU Throughput	VRAM Throughput
C. Island	Coarse	2958	155.764	3 103 522	4.0 %	2.51 %
	Medium	11 682	337.416	6 477 322	7.2 %	3.76 %
	Fine	45 924	549.277	12 982 607	25.0 %	7.05 %
G. Bump	Coarse	74 848	36.38	524 831	22.9 %	4.8 %
	Medium	297 152	50.11	564 031	65.5 %	11.8 %
	Fine	1 184 128	148.30	564 031	91.1 %	34.3 %

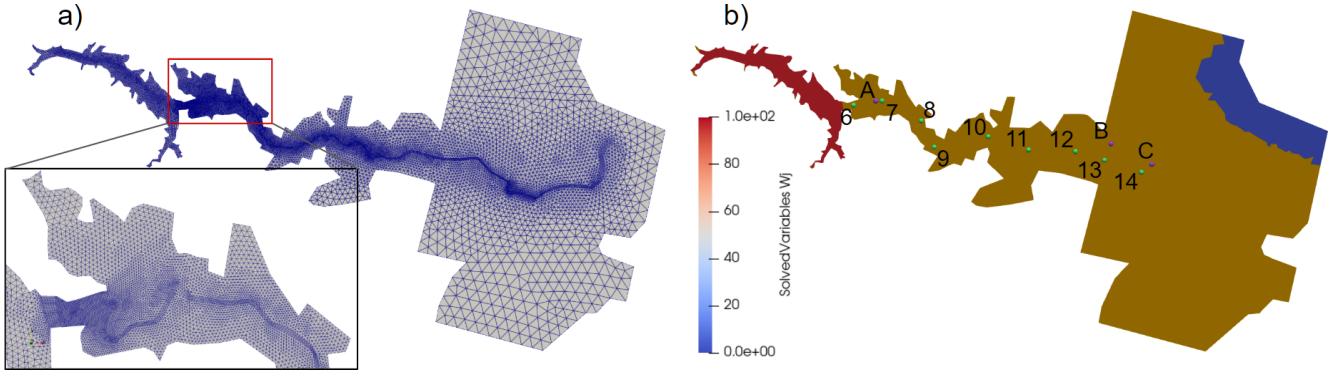

Figure 15. Grid used in the simulation **(a)** with zoomed view of the upstream refined part, and initial water height **(b)**. Points are the locations of measured data.

data. This poor resolution stems primarily from the coarse bathymetry dataset lifted from the original maps, therefore, finer remeshing would not necessarily resolve this discrepancy.

Given the complex bathymetry and the tightly spaced unstructured grid, some numerical artifacts arose in our first solutions from the interaction between the wet/dry treatment and the impermeable-wall ghost-cell boundary conditions. These occur in some border cells where the bathymetry outward normal points away from the domain, i.e., locations where water would naturally exit the computational area. Such cells can act as an artificial source of inflow, as observed in the animations provided in the supplement. In the present case, these artifacts do not significantly affect the solutions presented, since the water height introduced (~ 0.01 m) is negligible compared to the wave arrival heights (~ 10 m). However, careful grid construction can help prevent such situations. Figure 16 illustrates the original and corrected bathymetry near domain boundaries, showing a noticeable reduction in spurious water influx after applying the correction.

Unfortunately, some artificial inflows remain even after applying this correction procedure. Figure 17 shows the wave arrival at transformers A and C, where the inundation pattern is correctly reproduced but small residual mass contributions from these artifacts are still visible in some cells outside the measurement zones, still with negligible sizes; confirming that these artefacts do not affect the reported results.

These results confirm that SWEpy can reliably reproduce complex inundation dynamics over irregular terrain while also identifying clear avenues for improvement. Zones with limited topographic resolution would benefit from targeted mesh refinement, and the integration of additional physical processes, such as rheology, sediment transport, and the influence of steep bathymetric gradients or strong flow curvature, could further enhance predictive skill.

5.3 Real-life scenario 2: Maule 2010 tsunami

To evaluate SWEpy's capability for far-field tsunami simulation–specifically its ability to propagate long-period waves over transoceanic distances with minimal numerical dissipation – we reproduce the 2010 Maule tsunami, generated by an M_w 8.8 earthquake along the Chilean subduction zone (Benavente and Cummins, 2013). This event produced measurable signals across the Pacific basin, where Coriolis effects are dynamically relevant and numerical accuracy over very long propagation paths is essential for preserving wave amplitude and shape. This test also enables an assessment of the combined impact of WENO spatial reconstruction and RK4,3 time integration, compared with minmod and explicit Euler methods.

The computational mesh is an equilateral triangular grid generated from GEBCO bathymetry (GEBCO Bathy-

Table 7. Simulation results and relative errors, evaluated against recorded data, for both TELEMAC and SWEpy.

Point	Recorded Data		SWEpy			TELEMAC		
	h_{max} [m]	t_{ArrA} [s]	h_{max} [m]	t_{ArrA} [s]	Err [%]	h_{max} [m]	t_{ArrA} [s]	Err [%]
A	–	–	–	–	–	–	–	–
B	–	1140	–	1071	−6	–	1142.9	0
C	–	1320	–	1088	−18	–	1387.3	5
P6	40.3	–	37.72	–	−6	81.58	–	102
P7	14.6	–	18.13	–	24	55.88	–	283
P8	24.0	–	22.46	–	−6	53.21	–	122
P9	12.8	–	18.6	–	45	48.14	–	276
P10	11.8	–	15.34	–	30	36.88	–	213
P11	8.3	–	6.21	–	-25	25.41	–	206
P12	10.1	–	5.89	–	−42	19.29	–	91
P13	6.8	–	12.21	–	80	17.74	–	161
P14	−5.4	–	4.32	–	−20	12.71	–	135

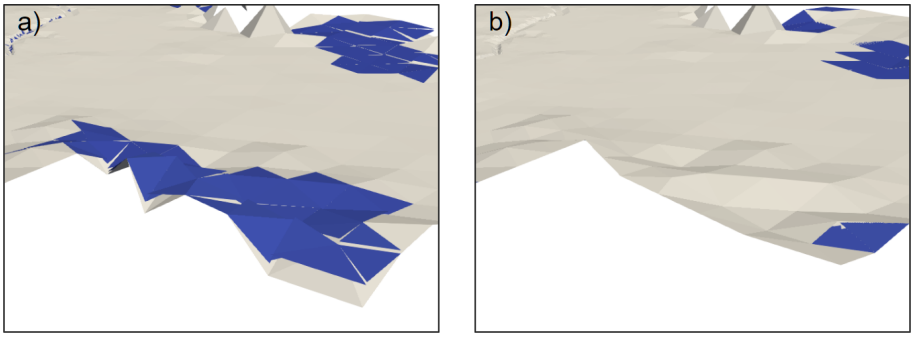

Figure 16. 3D View of solution before **(a)** and after **(b)** grid correction. Artificial water influx is reduced thanks to the bathymetry fix.

metric Compilation Group 2024, 2024) via a spherical–Cartesian transformation, containing approximately 10^6 cells ($\sim$ 536 000 nodes). Because the mesh was constrained to a rectangular bounding box for refinement, additional cells were created at the corners; the effective number of wet cells representing the domain is therefore about 860 000. The initial sea-surface displacement is prescribed from the inversion by Benavente and Cummins (2013) using the Okada fault-slip model, with zero initial velocity. Coriolis effects are included by setting $f = 10^{-4}\,\mathrm{s}^{-1}$, typical in mid-latitude regions (Kundu et al., 2012). Simulations span 24 h of physical time, using adaptive time-stepping with a Courant–Friedrichs–Lewy (CFL) number of 0.25 for explicit Euler integration and 0.5 for RK4,3 integration, applied to each reconstruction–integrator configuration.

Two virtual wave gauges are positioned at the locations of NOAA DART buoys 32 412 (southwest of Lima, Peru) and 21 413 (southeast of Kyoto, Japan), as indicated in Fig. 18. At each gauge, the water-column height is recorded every 60 s of simulated time, yielding continuous time series for the 24 h simulation period. This duration captures the primary tsunami signal and subsequent wave groups at both stations. The numerical results are compared with quality-controlled, rectified DART measurements (Mungov et al., 2005), enabling a direct evaluation of amplitude and phase preservation over basin-scale propagation.

Figure 19 compares the simulated tsunami waveforms at DART buoys 32 412 (panel a) and 21 413 (panel b) with quality-controlled observations and reference simulations from the HySEA model. At buoy 32 412, the WENO reconstruction with explicit Euler integration provides the closest match to the observed primary wave amplitude and timing, and retains secondary oscillations more effectively than the minmod and constant reconstructions. Minmod with a high limiter parameter ($\vartheta = 1.4$) reduces phase drift relative to the constant scheme, but still underestimates the amplitude of later wave groups. At buoy 21 413, located in the northwestern Pacific, all SWEpy configurations exhibit greater attenuation of the signal, reflecting the cumulative impact of propagation distance and coarse resolution; here, the WENO scheme again preserves amplitude better than the other reconstructions, with the minmod operator becoming too oscillatory, although the differences are less pronounced. Across both sites, HySEA results at 1.5 million cells show closer agreement with the DART data than SWEpy, consistent with the benefits of higher effective resolution. While WENO in-

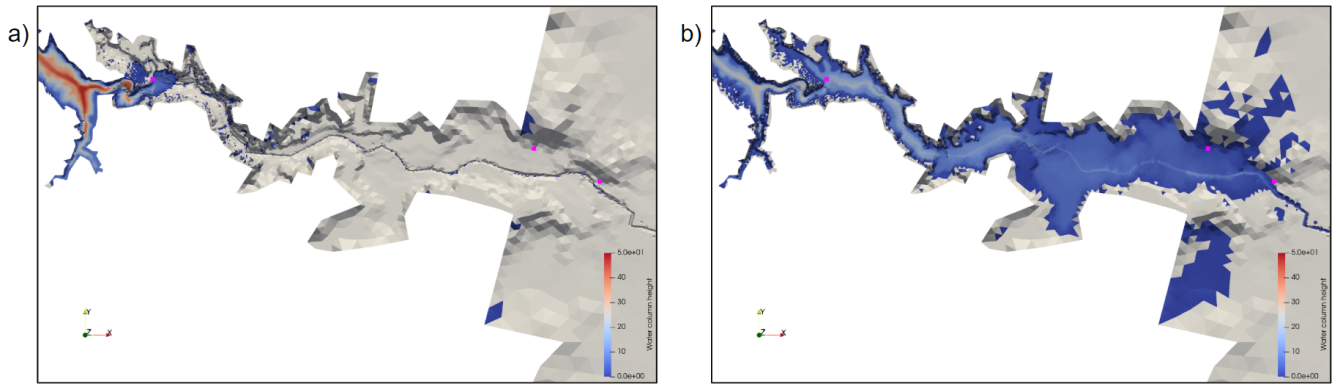

Figure 17. Top-down view of inundation wave arriving at transformer A **(a)** and C **(b)**. Transformer locations are marked with magenta boxes. Some water influx is present due to errors of the border-wet/dry interaction, but remains outside the measuring zones.

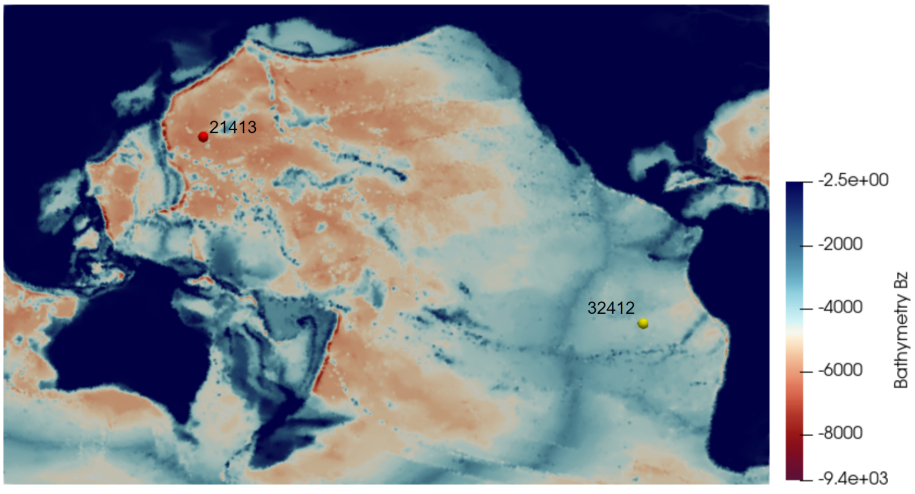

Figure 18. Bathymetry grid employed for the Maule tsunami simulation.

curs a modest computational cost increase over minmod (as explored later), it remains faster than real time for the domain and resolution tested, offering a consistent improvement in waveform fidelity.

The explicit Euler + WENO result is not shown for buoy 21 413 because the combination of explicit Euler and coarse resolution over the long propagation path generated high-frequency oscillations that obscured the primary tsunami signal (Fig. 20). Table 8 quantifies the performance of each configuration by comparing the maximum wave height H_{max} and arrival time T_{arr} of the first wave against the DART observations. These metrics complement the full time-series comparison by highlighting differences in amplitude attenuation and phase shift.

These metrics indicate that SWEpy underestimates the maximum crest height and predicts earlier arrivals at both stations, whereas HySEA exhibits smaller amplitude bias – particularly at Lima – and reduced phase error, consistent with the visual comparisons in Fig. 19. The differences likely reflect a combination of: (i) source smoothing introduced during interpolation to the computational grid, (ii) bathymetric resolution, with the HySEA configuration employing nearly twice as many cells, and (iii) projection-related errors from the spherical–Cartesian transformation, given that HySEA solves the SWEs directly on the sphere. This last point is important, since we only converted the grid to Cartesian coordinates via a haversine transformation; however, formulas for fluxes, cell side length, and cell area were maintained, when they should be calculated considering transformations. Quantifying the contribution of each factor is left for future work, with the aim of guiding targeted improvements to SWEpy's far-field performance through a rigorous treatment of the spherical case.

While initially producing the most accurate waveforms at the near-field gauge, the explicit Euler time integrator in combination with the WENO reconstruction develops spurious high-frequency oscillations after extended transoceanic propagation. These oscillations overwhelm the primary tsunami signal at the Kyoto gauge, making the solution unsuitable for quantitative analysis. Figure 20 illustrates

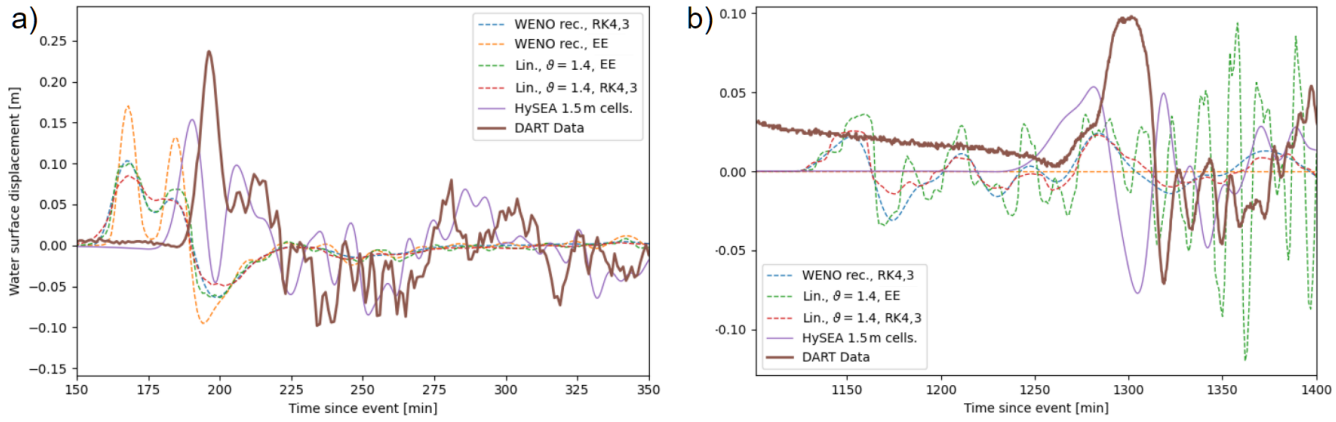

Figure 19. Tsunami profile comparison at DART buoy 32 412 **(a)** and 21 413 **(b)**. Results from the HySEA model are included for reference. The explicit Euler + WENO result for buoy 21 413 is omitted because of high-frequency oscillations that obscured the primary tsunami signal.

Table 8. Performance summary of SWEpy configurations and HySEA for the Maule 2010 tsunami benchmark. Maximum surface displacement H_{max} [m] and arrival time T_{arr} [min] are extracted at the first wave crest for DART buoys 32 412 (Lima) and 21 413 (Kyoto). Relative errors (%) are computed with respect to the DART observations.

Method	Lima				Kyoto			
	H_{max}	Err%	T_{arr}	Err%	H_{max}	Err%	T_{arr}	Err%
DART Data	0.237	–	196	–	0.098	–	1296	–
RK4,3 + WENO	0.1034	−56.4	167.263	−14.7	0.022	−77.6	1151.39	−11.2
explicit Euler + WENO	0.1703	−28.1	167.098	−14.7	–	–	–	–
RK4,3 + minmod	0.0851	−64.1	167.197	−14.7	0.02551	−74.0	1152	−11.1
explicit Euler + minmod	0.1005	−57.6	169.068	−13.7	0.03611	−63.2	1158.1	−10.7
HySEA	0.1537	−35.1	190.064	−3.02	0.0534	−45.5	1281	−1.2

the modeled free-surface elevation at the moments when the wave passes the Lima and Kyoto DART buoys, for both explicit Euler + WENO and RK4,3 + WENO configurations, highlighting the marked difference in solution smoothness.

Control of these oscillations could, in principle, be achieved by reducing the CFL number; however, this would further increase numerical diffusion. Even with a reduced value of CFL = 0.2, the oscillations remain excessive by the time the wave reaches Kyoto, producing spurious amplitudes of approximately 0.3 m, while simultaneously reducing the maximum height at Lima to 0.14 m.

Spurious oscillations in Fig. 20b may develop as a result of the interaction between the high-order reconstruction and the temporal discretization, due to the reduced level of numerical diffusivity and, consequently, the lack of an effective numerical dissipation mechanism to control the generated oscillations. In addition, the spatial mesh and, in particular, the way boundary conditions are imposed may also contribute to this behavior (see inset of Fig. 20a).

This phenomenon was investigated through an auxiliary numerical experiment in which a soliton propagating along a channel with periodic boundary conditions was considered; see Sect. S7.2 *Instability study: impact of temporal discretization* in the User Manual & Technical Reference (Meza et al., 2026). The results show that simulations employing a more diffusive reconstruction do not generate spurious oscillations, whereas the explicit Euler + WENO combination leads to the development of resonance-type oscillations. These oscillations may partially explain the spurious oscillatory patterns observed in the figure, ultimately giving rise to numerical convective instabilities in space.

To explore grid convergence, additional runs were performed with closer HySEA-SWEpy resolution agreement, using results from a HySEA run with a ∼ 250 000 element resolution as reference, and a SWEpy grid of ∼ 280 000 (effective) elements. Also, to assess the reduction of oscillatory behaviour in the explicit Euler + WENO case, a run with a CFL number of 0.15 was performed. Figure 21 shows the output at the DART buoys locations.

These results confirm two points discussed above: (1) The resolution of the mesh is indeed correlated to the arriving wave amplitude, with both SWEpy and HySEA achieving similar accuracy; and (2) control of the spurious oscillations via the CFL number is achievable with the trade-off of some

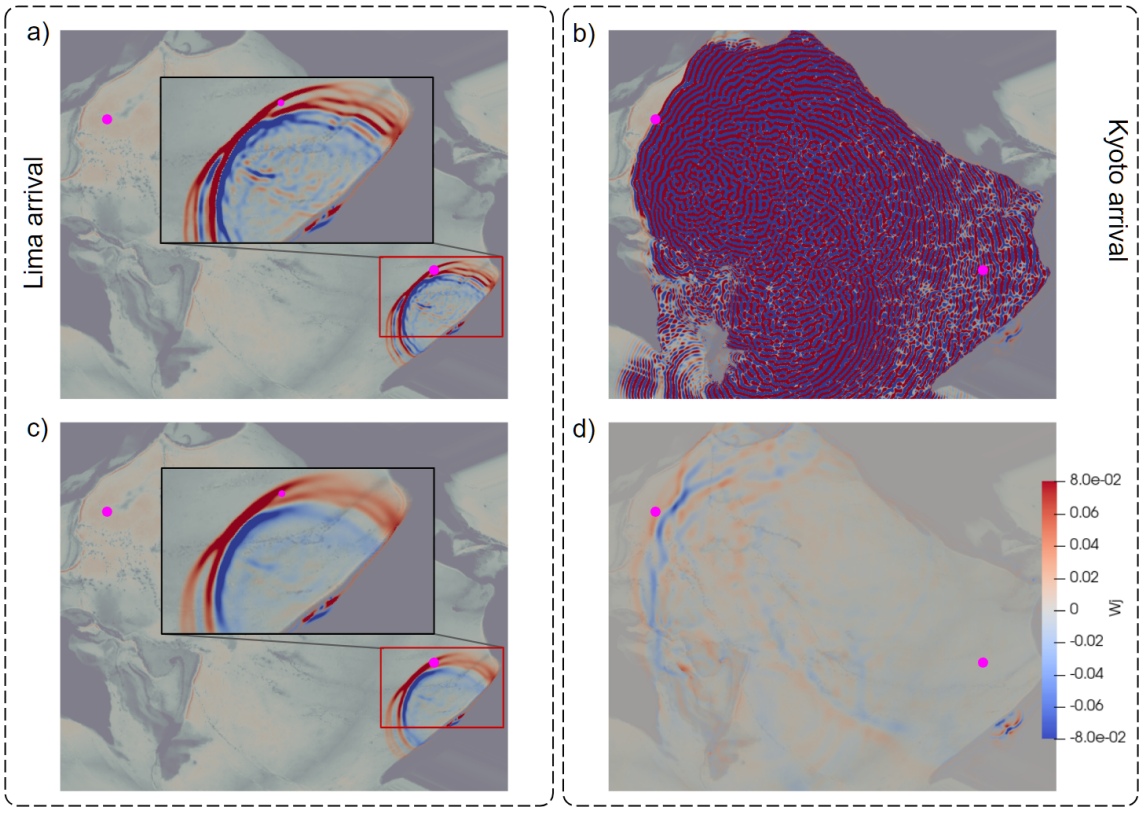

Figure 20. Snapshots of simulated free-surface elevation for the Maule 2010 tsunami at the times when the leading crest passes the locations of DART buoys 32 412 (Lima) and 21 413 (Kyoto). Insets show zoomed view of wave at Lima buoy. Panels **(a)** and **(b)** correspond to explicit Euler integration, and panels **(c)** and **(d)** to RK4,3 integration. Buoy locations are marked with magenta circles. These views provide basin-scale context for the time-series comparisons in Fig. 19.

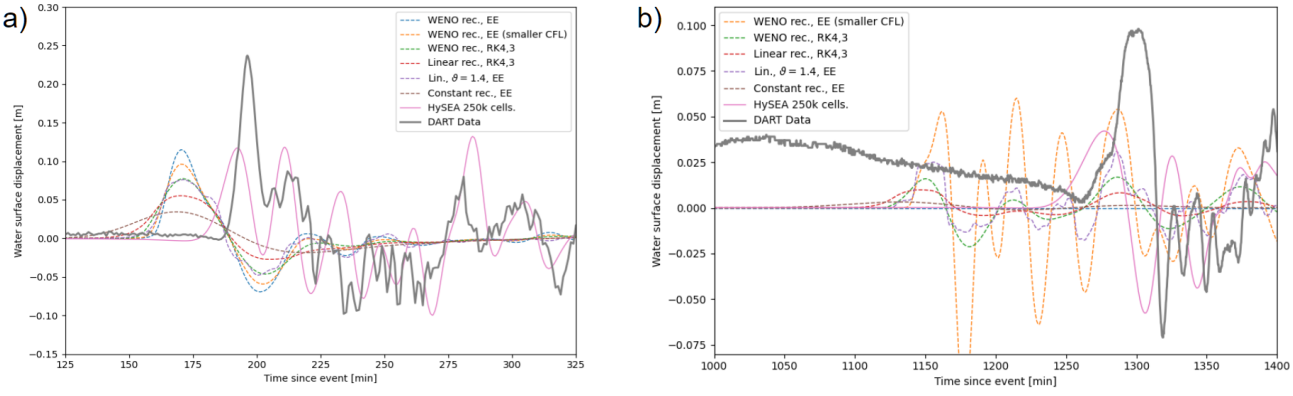

Figure 21. Tsunami profile comparison at DART buoy 32 412 **(a)** and 21 413 **(b)** for the coarser grids. Once again, results for the explicit Euler + WENO at the Kyoto buoy have been omitted.

quality degradation. This confirms that, with similar spatial resolution, SWEpy can attain comparable initial wave amplitudes to HySEA.

To illustrate computational efficiency and the benefits of GPU acceleration on a real-life problem, the Maule 2010 scenario was repeated on a different simplified, non linearized, mesh of approximately 2.5×10^5 elements, using different combinations of time integrators and spatial reconstructors. To test GPU-parallelization speedups, a serial, single core, CPU-only version of SWEpy was used as reference, where CuPy was replaced by the traditional NumPy. At this reduced resolution, wave-height errors ranged from $-98\,\%$ to $-13.4\,\%$ and arrival-time errors from $-2\,\%$ to $26.5\,\%$ relative to DART observations, reflecting the expected degrada-

Table 9. Comparison of methods explicit Euler + minmod (fastest) and RK4,3 + WENO (slowest). CPU times correspond to simulations ran on a serial CPU-only implementation. GPU times correspond to the present CuPy implementation.

Method	CPU (min)	GPU (min)	Speedup (×)
explicit Euler + minmod	37.32	2.87	13.0
WENO + RK4,3	360.49	17.31	20.8

tion in solution quality. Despite this, execution times for all configurations were faster than real time (Table 9).

Tests were performed on an Intel® Core™ i5-10300H CPU (10th generation) and an NVIDIA GeForce GTX 1650 GPU, both consumer-grade components released more than six years prior to writing. However, this performance comparison is intended to illustrate the benefits of GPU acceleration within the SWEpy framework using a standard single-core CPU baseline, rather than providing a hardware-optimized or scalability-focused performance study across architectures. These results underscore SWEpy's ability to deliver high-performance simulations on widely available, non-specialized hardware.

As a whole, the Maule 2010 benchmark results demonstrate SWEpy's ability to reproduce the main features of basin-scale tsunami propagation, including crest arrival timing, amplitude decay, and the modulation of subsequent wave groups. Among the tested schemes, the WENO reconstruction consistently yields the most accurate far-field waveforms, though in combination with explicit Euler integration it can develop basin-scale oscillations that obscure the signal at distant stations (Fig. 20). These oscillations persist even under reduced CFL numbers, with the trade-off of increased diffusion and degraded amplitudes at nearer gauges. The RK4,3 integrator mitigates this instability while preserving much of WENO's accuracy, making it the most balanced choice for long-range simulations. GPU acceleration enables faster-than-real-time performance even for the most demanding configuration, with speedups exceeding 20× on consumer-grade hardware with respect to the serial implementation. Together, these results confirm the model's applicability to large-domain tsunami scenarios, while highlighting clear paths for improvement in projection accuracy, bathymetric resolution, and source initialization to close the remaining gaps with higher-resolution reference models such as HySEA.

6 Conclusions

We presented SWEpy, an open-source GPU-accelerated (CUDA) Python package for the SWEs on unstructured triangular grids. The solver addresses hydrodynamic hazard scenarios across a wide range of spatial and temporal scales, from localized flooding to transoceanic tsunami propagation, by combining conservative numerical schemes with high-performance computing capabilities on consumer-grade hardware. SWEpy's modular design enables researchers to adapt or extend the code for specific applications while maintaining computational efficiency and numerical stability. Core features include robust wetting/drying treatment, low-diffusion reconstructions, and well-balanced source-term discretizations, which allow for reliable simulations in complex, high-risk environments.

To evaluate spatial accuracy, a grid-refinement study was conducted against a numerically converged reference solution computed with SWEpy on the finest mesh. The results demonstrated a monotonic reduction in L^2 error with decreasing Δx. Constant reconstruction achieved better-than-first-order accuracy, while both linear and quadratic variants exhibited second-order convergence. Although the quadratic WENO reconstruction did not reach full third-order accuracy due to flux–bathymetry coupling, it reduced errors by an average of $\sim 58\,\%$ relative to the linear reconstruction at matched resolution. Importantly, across all refinements, the well-balanced source-term discretization preserved steady-state conditions, providing a reliable numerical baseline for subsequent real-case validations.

Building on this, the accuracy and robustness of SWEpy, particularly its wet/dry treatment, were further assessed using the well-known Conical Island laboratory experiment, a benchmark widely used in coastal hydrodynamics. This configuration provides high-quality measurements of wave transformation, run-up, and diffraction around an isolated obstacle. SWEpy successfully reproduced the main free-surface patterns and wetting–drying transitions with a good agreement to experimental observations, confirming its ability to capture the fundamental physics of wave–structure interaction. Beyond serving as a verification benchmark, this experiment establishes confidence in the solver's applicability to more complex and operationally relevant scenarios, including large-scale dam-break inundations and basin-scale tsunami propagation over realistic bathymetry.

Real-case applications such as the Malpasset dam-break and the Maule 2010 tsunami highlight different aspects of the solver's performance. In the Malpasset case, characterized by a short-duration, high-gradient flood in a confined valley, SWEpy maintained physically realistic wetting–drying behavior, conserved mass across advancing fronts, and achieved substantially lower water level errors than TELEMAC's FV solver, with arrival times generally within $\sim 18\,\%$ of observations. In contrast, the Maule tsunami case exposed current limitations in large-scale basin-propagation.

All SWEpy configurations underestimated the first-wave amplitude and predicted earlier arrival times at both DART buoys. Among the tested schemes, WENO + EE performed best, with −28 % error in amplitude and −15 % error in arrival time, although it developed high-frequency oscillations over long propagation distances. The WENO + RK4,3 scheme remained oscillation-free and reduced dissipation and phase drift relative to lower-order schemes; however, notable amplitude and timing biases persisted. These discrepancies likely stem from source smoothing during interpolation, projection errors between spherical and Cartesian coordinates (in contrast to HySEA's spherical formulation), and coarser bathymetric resolution. Together, these findings identify clear directions for improvement, including enhanced projection accuracy, improved source initialization, and refined mesh resolution, which are critical for achieving stronger far-field predictive capability at ocean-basin scales.

From a computational perspective, SWEpy delivers high-performance even on modest hardware that is more than six years old at the time of writing. A performance and scalability study highlighted both the advantages of GPU acceleration and its limitations, particularly regarding kernel launch overhead. GPU and VRAM throughput were analyzed to identify which configurations most effectively utilize the hardware. Using a simplified unstructured mesh of approximately 2.5×10^5 elements, the 24 h Maule 2010 tsunami scenario (in its most demanding reconstruction and timestep configuration) was simulated in approximately 17 min on an NVIDIA GeForce GTX 1650 laptop GPU. This corresponds to a speedup of approximately ∼ 2100 % relative to the single-core CPU baseline, executed on an Intel® Core™ i5-10300H processor. These results, enabled by extensive CuPy-based vectorization and memory-efficient data structures, demonstrate that large-scale SWEs simulations on unstructured grids can be performed well within real time on widely accessible systems, significantly lowering the barrier to high-performance hydrodynamic modeling while maintaining scalability for more detailed studies.

SWEpy is built on the CuPy library, which is primarily developed for the NVIDIA CUDA ecosystem. While CuPy offers experimental support for AMD GPUs through the HIP/ROCm interface, all validation and performance results presented here rely exclusively on the stable CUDA backend to ensure numerical reproducibility, performance consistency, and software robustness in this initial release. As CuPy's multi-backend capabilities continue to mature, SWEpy is well positioned to benefit from broader GPU compatibility without requiring fundamental changes to its core implementation.

Despite these advances, several limitations remain, as revealed by the validation studies. In the Maule 2010 tsunami case, amplitude underestimation and arrival-time biases indicate the need for improved bathymetric interpolation, reduced projection error between spherical and Cartesian coordinates, and higher-resolution meshes for basin-scale applications. In the Malpasset case, complex gridding and bathymetry interactions suggest that the wet/dry treatment may interact with ghost-cell boundary conditions, potentially introducing artificial sources of mass influx, which warrants further investigation. Additionally, the sub-third-order convergence observed for the quadratic WENO scheme indicates that flux–bathymetry coupling remains a limiting factor for spatial accuracy. More broadly, the solver inherits the hydrostatic assumption of the SWEs, which may introduce inaccuracies in steep or highly curved flows where vertical accelerations are significant. Ongoing development efforts are therefore focused on improving bathymetric handling, coordinate projection, spatial accuracy, and physical modeling to enhance the robustness and applicability of SWEpy.

From a software development perspective, the roadmap for SWEpy emphasizes continued optimization of GPU efficiency and further refinement of its modular architecture. This design facilitates community contributions and allows individual components – such as source terms, numerical schemes, and physical models – to evolve independently. While the current implementation targets single-GPU execution, future work may explore improved utilization of multi-core CPUs through process-based parallelism using Python's native multiprocessing or CuPy-native kernel customization and fusion features. In contrast, implementing efficient MPI-based parallelization for unstructured grids entirely in Python remains challenging, as communication overhead can outweigh computational gains. Without hybrid C++/Python approaches – such as MPI wrappers around compiled kernels – pure Python MPI solutions often diminish the benefits of parallel execution. These considerations reinforce the current emphasis on GPU-centric acceleration, where high throughput can be achieved within a unified and accessible Python framework.

We anticipate that the flexibility of SWEpy provides a strong foundation for extending its capabilities beyond the current scope. Planned developments include additional source-term modules for rainfall and infiltration to support catchment-scale hydrology, as well as rheological models for non-Newtonian fluids to simulate landslides, tailings dam breaches, snow avalanches, and debris flows. The framework is also well suited for incorporating two-layer shallow water formulations (2LSWE), following recent advances in CU-type schemes (Cao et al., 2024; Liu, 2021a), to model stratified and two-phase flows such as water–mud avalanches or estuarine mixing. These enhancements, together with ongoing improvements in bathymetric representation, mesh refinement, and large-scale projection accuracy, will further strengthen the solver's predictive capabilities across a broad spectrum of hydrodynamic hazards. By maintaining its modular, open-source, and GPU-accelerated design, SWEpy is well positioned to evolve into a versatile and high-performance modeling framework that bridges the gap

between research prototypes and operational tools for flood and tsunami risk assessment worldwide.

Appendix A: Reconstruction Operators

A1 Linear (Minmod) Reconstruction

The minmod linear reconstruction is detailed in Algorithm A1.

Algorithm A1 Minmod linear reconstruction.

1: identify neighbors of each cell (cf. group Ω_{jk} in Fig. 3)
2: construct planes passing through the mean value of the variable at the barycenters of the cell and its neighbors.
3: calculate $((q_x)_j, (q_y)_j)$
4: select the plane whose gradient's magnitude is lowest among its three constructed planes
5: **for** `each cell` **do**
6: **for** `each side` **do**
7: **if not** `mean value at cell<reconstructed value at midpoint<mean value at side's neighbor` **then**
8: impose a constant plane passing through mean value of the variable over the cell
9: **end if**
10: **end for**
11: **end for**

This ensures monotonicity by selecting the least oscillatory interpolant. The algorithm is implemented in parallel for all cells in the `SWEpy` code by operating arrays containing the mentioned quantities.

A2 Quadratic WENO Reconstruction

The quadratic WENO reconstruction is summarized in Algorithm A2, which outlines the steps for computing the operator on unstructured triangular grids.

The detailed calculation procedure for the quadratic polynomial $p_{j_0}(x, y)$ relies solely on geometric information associated with the control cell Ω_j. Consider the quadratic form:

$$p_{j_0}(x, y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy, \quad \text{(A1)}$$

shown in Eq. (27), the associated integrals in the construction of the quadratic polynomials and smoothness indicators are calculated exactly.

This polynomial approximates the mean values over the stencil cells and exactly reproduces the mean in Ω_{j_0}. The associated integrals for smoothness indicators are computed exactly. Since the system is overdetermined (9 equations for 5 coefficients), it is solved via least-squares: given $M\boldsymbol{a} = \Delta\boldsymbol{q}$,

Algorithm A2 Quadratic WENO reconstruction.

1: identify neighbors of each cell (cf. group Ω_{jk} in Fig. 3)
2: identify neighbors of neighbors of each cell (cf. group Ω_{jkl} in Fig. 3)
3: **for** `each cell` **do**
4: Construct quadratic polynomial interpolator u_q (cf. Eq. A1)
5: Construct linear interpolators $p_{1,j}, p_{2,j}, p_{3,j}, p_{4,j}$
6: Set linear weights $\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4$
7: Calculate smoothness indicators $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ (Eq. 2.8 in Zhu and Qiu, 2018)
8: Calculate corrector τ for WENO-Z procedure (Eq. 2.13 in Zhu and Qiu, 2018)
9: Calculate and normalize nonlinear weights w_0, w_1, w_2, w_3, w_4
10: Construct reconstruction operator over cell as $\tilde{q}_j = \frac{w_0}{\gamma_0}\left(p_0 - \sum_{k=1}^{4}\gamma_k p_{k,j}\right) + \sum_{k=1}^{4} w_k p_{k,j}$
11: **end for**

$$\boldsymbol{a} \overset{\text{lsq}}{=} (M^t M)^{-1} M^t \Delta\boldsymbol{q}. \quad \text{(A2)}$$

This approach ensures high-order accuracy while preserving well-balancing in the CU scheme. A further detailed explanation of the reconstructor and its usage in the CU scheme will be in Fuenzalida Alarcón et al. (2025).

A3 Wet/dry treatment

The positivity-preserving correction for wet/dry fronts is outlined in Algorithm A3.

This algorithm corrects reconstructions yielding negative depths by fitting a mass-conserving linear plane, executed in parallel on the GPU for efficiency.

Algorithm A3 Positivity preserving wet/dry reconstruction.

find cells at the wet/dry front
for `each cell in front` **do**
  determines if the cell has one or two dry vertices
  if `cell has two dry vertices` **then**
    $v_1 := (x_{jk1}, y_{jk1}) \leftarrow$ `dry vertex 1`
    $v_2 := (x_{jk2}, y_{jk1}) \leftarrow$ `dry vertex 2`
    $v_3 := (x_j, y_j) \leftarrow$ `cell barycenter`
    $B_{jk1}, B_{jk2}, W \leftarrow$ `bathymetry at` v_1, `bat. at` v_2, `cell mean water level`
    replace reconstruction with plane passing through (v_1, B_{jk1}), (v_2, B_{jk2}), and (v_3, W)
  else if `cell has one dry vertex` **then**
    $v_1 := (x_{jk1}, y_{jk1}) \leftarrow$ `dry vertex`
    $v_2 := (x_{jk2}, y_{jk2}) \leftarrow$ `wet vertex`
    $v_3 := (x_j, y_j) \leftarrow$ `cell barycenter`
    $B_{jk1}, w, W \leftarrow$ `bat. at` v_1, `cell mean w.l.,` 3/2(w - cell mean bat.)+bat. at v_2
    replace reconstruction with plane passing through (v_1, B_{jk1}), (v_2, W), and (v_3, w)
  end if
end for

Code and data availability. SWEpy is available for CUDA-compatible devices through GitHub at https://github.com/joaquinmeza90/SWEpy.git (last access: 7 April 2026) (https://doi.org/10.5281/zenodo.19464287, Kusanovic et al., 2026), under a GPL license. To facilitate onboarding and ensure reproducibility, a comprehensive User Manual and Technical Reference (Meza et al., 2026) – covering installation, input data formats, and benchmark execution – can be downloaded from the project's official website: https://www.hydrology.cl/swepy (last access: 15 April 2026). A repository containing test cases showed in Sect. 5 is available at https://doi.org/10.5281/zenodo.16789889 (Kusanovic et al., 2025). This repository includes many of the cases reported here, except those for which data cannot be publicly released but can be obtained from the original sources (e.g., TELEMAC validation suite for Malpasset, GEBCO and NOAA for Maule).

Video supplement. Animations of both the Malpasset (3D and topdown view) and Maule (explicit Euler + WENO and RK4,3 + WENO configurations) scenarios simulated are available for visualization and download at https://doi.org/10.5281/zenodo.16798435 (Fuenzalida Alarcón, 2025).

Supplement. The supplement related to this article is available online at https://doi.org/10.5194/gmd-19-3953-2026-supplement.

Author contributions. JF contributed to conceptualization, investigation, software development, model validation, visualization, and writing. DK contributed to conceptualization, software development, formal analysis, model validation, and writing. JM contributed to conceptualization, software development, formal analysis, model validation, and writing. RM contributed to conceptualization, formal analysis, model validation, and writing. PC contributed to formal analysis, model validation, and writing.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. The authors bear the ultimate responsibility for providing appropriate place names. Views expressed in the text are those of the authors and do not necessarily reflect the views of the publisher.

Acknowledgements. The authors acknowledge the contributions of reviewers in improving this manuscript. Joaquín Meza was supported by Universidad Técnica Federico Santa María through the project PI_LII_23_06. Patricio Catalan was supported by Centro de Investigación para la Gestión Integrada del Riesgo de Desastres, ANID CIGIDEN R+ CIN250023, and ANID CCTVal CIA2500027.

Financial support. This research has been supported by the Agencia Nacional de Investigación y Desarrollo, Chile (ANID CIN (grant no. CIN250023, CIGIDEN)), the Agencia Nacional de Investigación y Desarrollo, Chile (ANID CIA, grant no. CIA2500027, CCTVal), and the Universidad Técnica Federico Santa María (grant no. PI_LII_23_06).

Review statement. This paper was edited by James Kelly and reviewed by Kyle Mandli and one anonymous referee.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S.,

Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, Technical report, Google, https://www.tensorflow.org/ (last access: 15 April 2026), 2015.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., and Chintala, S.: PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation, in: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Vol. 2, ASPLOS '24, 929–947, ACM, https://doi.org/10.1145/3620665.3640366, 2024.

ANSYS Inc.: ANSYS Fluent Theory Guide, ANSYS, Inc., 275 Technology Drive, Canonsburg, PA 15317, https://ansyshelp.ansys.com/public//Views/Secured/corp/v242/en/flu_th/flu_th.html (last access: 15 April 2026), 2013.

Arminjon, P. and St-Cyr, A.: Nessyahu–Tadmor-type central finite volume methods without predictor for 3D Cartesian and unstructured tetrahedral grids, Appl. Numer. Math., 46, 135–155, 2003.

Ayachit, U.: The paraview guide (full color version), Kitware, ISBN 978-1930934306, 2015.

Behrens, J.: TsunAWI – Unstructured Mesh Finite Element Model for the Computation of Tsunami Scenarios with Inundation, in: 5th NAFEMS CFD-Seminar: Simulation komplexer Strömungsvorgänge (CFD) – Anwendungen und Trends, ISBN 978-1-874376-33-0, https://epic.awi.de/id/eprint/18394/ (last access: 7 May 2026), 2008

Behrens, J., Androsov, A., Babeyko, A. Y., Harig, S., Klaschka, F., and Mentrup, L.: A new multi-sensor approach to simulation assisted tsunami early warning, Nat. Hazards Earth Syst. Sci., 10, 1085–1100, https://doi.org/10.5194/nhess-10-1085-2010, 2010.

Benavente, R. and Cummins, P. R.: Simple and reliable finite fault solutions for large earthquakes using the W-phase: The Maule ($M_w = 8.8$) and Tohoku ($M_w = 9.0$) earthquakes, Geophys. Res. Lett., 40, 3591–3595, https://doi.org/10.1002/grl.50648, 2013.

Berger, M. J., George, D. L., LeVeque, R. J., and Mandli, K. T.: The GeoClaw software for depth-averaged flows with adaptive refinement, Adv. Water Resour., 34, 1195–1206, https://doi.org/10.1016/j.advwatres.2011.02.016, 2011.

Bomers, A., Schielen, R. M. J., and Hulscher, S. J. M. H.: The influence of grid shape and grid size on hydraulic river modelling performance, Environ. Fluid Mech., 19, 1273–1294, https://doi.org/10.1007/s10652-019-09670-4, 2019.

Briggs, M. J., Synolakis, C. E., Harkins, G. S., and Green, D. R.: Laboratory experiments of tsunami runup on a circular island, Pure Appl. Geophys., 144, 569–593, https://doi.org/10.1007/BF00874384, 1995

Brunner, G.: HEC-RAS 2D User's Manual Version 6.0, US Army Corps of Engineers, Institute for Water Resources, Hydrologic Engineering Center (HEC), Davis, CA, USA, https://www.hec.usace.army.mil/confluence/rasdocs/r2dum/latest (last access: 15 April 2026), 2021.

Bryson, S. and Levy, D.: Balanced central schemes for the shallow water equations on unstructured grids, SIAM J. Sci. Comput., 27, 532–552, 2005.

Bryson, S., Epshteyn, Y., Kurganov, A., and Petrova, G.: Well-balanced positivity preserving central-upwind scheme on triangular grids for the Saint-Venant system, ESAIM: Mathematical Modelling and Numerical Analysis, 45, 423–446, https://doi.org/10.1051/m2an/2010060, 2011.

Cao, Y., Kurganov, A., Liu, Y., and Zeitlin, V.: Flux globalization-based well-balanced path-conservative central-upwind scheme for two-dimensional two-layer thermal rotating shallow water equations, J. Comput. Phys., 515, 113273, https://doi.org/10.1016/j.jcp.2024.113273, 2024.

Carlotto, T., Borges Chaffe, P. L., Innocente dos Santos, C., and Lee, S.: SW2D-GPU: A two-dimensional shallow water model accelerated by GPGPU, Environ. Modell. Softw., 145, 105205, https://doi.org/10.1016/j.envsoft.2021.105205, 2021.

Castro-Orgaz, O. and Hager, W. H.: Shallow water hydraulics, Springer, https://doi.org/10.1007/978-3-030-13073-2, 2019.

Catalán, P. A., Gubler, A., Cañas, J., Zúñiga, C., Zelaya, C., Pizarro, L., Valdés, C., Mena, R., Toledo, E., and Cienfuegos, R.: Design and operational implementation of the integrated tsunami forecast and warning system in Chile (SIPAT), Coast. Eng. J., 62, 373–388, https://doi.org/10.1080/21664250.2020.1727402, 2020.

Caviedes-Voullième, D., Morales-Hernández, M., Norman, M. R., and Özgen-Xian, I.: SERGHEI (SERGHEI-SWE) v1.0: a performance-portable high-performance parallel-computing shallow-water solver for hydrology and environmental hydraulics, Geosci. Model Dev., 16, 977–1008, https://doi.org/10.5194/gmd-16-977-2023, 2023.

Chertock, A., Cui, S., Kurganov, A., and Wu, T.: Well-balanced positivity preserving central-upwind scheme for the shallow water system with friction terms, Int. J. Numer. Meth. Fl., 78, 355–383, https://doi.org/10.1002/fld.4023, 2015.

Chertock, A., Dudzinski, M., Kurganov, A., and Lukáčová-Medvid'ová, M.: Well-balanced schemes for the shallow water equations with Coriolis forces, Numer. Math. (Heidelb.), 138, 939–973, 2018.

Chow, V. T., Maidment, D. R., and Mays, L. W.: Applied Hydrology, McGraw-Hill, New York, USA, ISBN 978-0-07-010810-3, 1988.

Christov, I. and Popov, B.: New non-oscillatory central schemes on unstructured triangulations for hyperbolic systems of conservation laws, J. Comput. Phys., 227, 5736–5757, 2008.

Chu, S., Kurganov, A., and Menshov, I.: New adaptive low-dissipation central-upwind schemes, Appl. Numer. Math., 209, 155–170, https://doi.org/10.1016/j.apnum.2024.11.010, 2025.

Courty, L. G., Pedrozo-Acuña, A., and Bates, P. D.: Itzï (version 17.1): an open-source, distributed GIS model for dynamic flood simulation, Geosci. Model Dev., 10, 1835–1847, https://doi.org/10.5194/gmd-10-1835-2017, 2017.

Cui, S., Gu, Y., Kurganov, A., Wu, K., and Xin, R.: Positivity-preserving new low-dissipation central-upwind schemes for

compressible Euler equations, J. Comput. Phys., 538, 114189, https://doi.org/10.1016/j.jcp.2025.114189, 2025.

Delestre, O., Darboux, F., James, F., Lucas, C., Laguerre, C., and Cordier, S.: FullSWOF: Full Shallow-Water equations for Overland Flow, Journal of Open Source Software, 2, 448, https://doi.org/10.21105/joss.00448, 2017.

Delis, A. I. and Nikolos, I. K.: Shallow Water Equations in Hydraulics: Modeling, Numerics and Applications, Water, https://doi.org/10.3390/w13243598, 2021.

Desveaux, V. and Masset, A.: A fully well-balanced scheme for shallow water equations with Coriolis force, Commun. Math. Sci., 20, 1875–1900, https://doi.org/10.4310/CMS.2022.v20.n7.a4, 2022.

Fernández-Nóvoa, D., González-Cao, J., and García-Feal, O.: Enhancing Flood Risk Management: A Comprehensive Review on Flood Early Warning Systems with Emphasis on Numerical Modeling, Water, 16, https://doi.org/10.3390/w16101408, 2024.

Fernández-Pato, J., Morales-Hernández, M., and García-Navarro, P.: Implicit 2D surface flow models performance assessment: Shallow Water Equations vs. Zero-Inertia Model, E3S Web Conf., 40, 05008, https://doi.org/10.1051/e3sconf/20184005008, 2018.

Fuenzalida Alarcón, J. A.: Animations of Malpasset and Maule simulations using SWEpy, Zenodo [video], https://doi.org/10.5281/zenodo.16798435, 2025.

Fuenzalida Alarcón, J. A., Meneses, R., Meza, J., and Kusanovic, D.: Adaptive Local Reconstruction for Solving Saint-Venant Equations in Irregular Domains, in preparation, 2025.

García-Feal, O., González-Cao, J., Gómez-Gesteira, M., Cea, L., Domínguez, J. M., and Formella, A.: An Accelerated Tool for Flood Modelling Based on Iber, Water, 10, 1459, https://doi.org/10.3390/w10101459, 2018.

GEBCO Bathymetric Compilation Group 2024: The GEBCO_2024 Grid – a continuous terrain model of the global oceans and land, NERC EDS British Oceanographic Data Centre NOC [data set], https://doi.org/10.5285/1C44CE99-0A0D-5F4F-E063-7086ABC0EA0F, 2024.

Gottlieb, S., Shu, C.-W., and Tadmor, E.: Strong Stability-Preserving High-Order Time Discretization Methods, SIAM Rev., 43, 89–112, https://doi.org/10.1137/S003614450036757X, 2001.

Gottlieb, S., Shu, C.-W., and Ketcheson, D.: Strong Stability Preserving Runge-kutta And Multistep Time Discretizations, World Scientific Publishing, Singapore, Singapore, https://doi.org/10.1142/7498, 2010.

Greenberg, J. M. and Leroux, A. Y.: A Well-Balanced Scheme for the Numerical Processing of Source Terms in Hyperbolic Equations, SIAM J. Numer. Anal., 33, 1–16, https://doi.org/10.1137/0733001, 1996.

Harig, S., Chaeroni, Pranowo, W. S., and Behrens, J.: Tsunami simulations on several scales: Comparison of approaches with unstructured meshes and nested grids, Ocean Dynam., 58, 429–440, https://doi.org/10.1007/s10236-008-0162-5, 2008.

Harig, S., Immerz, A., Weniza, Griffin, J., Weber, B., Babeyko, A., Rakowsky, N., Hartanto, D., Nurokhim, A., Handayani, T., and Weber, R.: The Tsunami Scenario Database of the Indonesia Tsunami Early Warning System (InaTEWS): Evolution of the Coverage and the Involved Modeling Approaches, Pure Appl. Geophys., 177, 1379–1401, https://doi.org/10.1007/s00024-019-02305-1, 2019.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E.: Array programming with NumPy, Nature, 585, 357–362, https://doi.org/10.1038/s41586-020-2649-2, 2020.

Hervouet, J.-M.: Hydrodynamics of free surface flows: modelling with the finite element method, John Wiley & Sons, https://doi.org/10.1002/9780470319628, 2007.

Jawahar, P. and Kamath, H.: A high-resolution procedure for Euler and Navier–Stokes computations on unstructured grids, J. Comput. Phys., 164, 165–203, 2000.

Jodhani, K. H., Patel, D., and Madhavan, N.: A review on analysis of flood modelling using different numerical models, Mater. Today-Proc., 80, 3867–3876, https://doi.org/10.1016/j.matpr.2021.07.405, 2023.

Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., and Pender, G.: A deep convolutional neural network model for rapid prediction of fluvial flood inundation, J. Hydrol., 590, 125481, https://doi.org/10.1016/j.jhydrol.2020.125481, 2020.

Kirstetter, G., Delestre, O., Lagrée, P.-Y., Popinet, S., and Josserand, C.: B-flood 1.0: an open-source Saint-Venant model for flash-flood simulation using adaptive refinement, Geosci. Model Dev., 14, 7117–7132, https://doi.org/10.5194/gmd-14-7117-2021, 2021.

Kloeckner, A., Wohlgemuth, G., Lee, G., Rybak, T., Nitz, A., Chiang, D., Seibert, S., Bergtholdt, M., Unterthiner, T., Markall, G., Kotak, M., Favre-Nicolin, V., Opanchuk, B., Merry, B., Pinto, N., Milo, F., Collignon, T., Rathgeber, F., Perkins, S., Rutsky, V., Catanzaro, B., Park, A., Witherden, F., E. Givon, L., Pfister, L., Brubaker, M., ZA, R., Hausammann, L., and Gohlke, C.: PyCUDA, Zenodo [code], https://doi.org/10.5281/ZENODO.15620354, 2025.

Kobayashi, K., Kitamura, D., Ando, K., and Ohi, N.: Parallel computing for high-resolution/large-scale flood simulation using the K supercomputer, Hydrological Research Letters, 9, 61–68, https://doi.org/10.3178/hrl.9.61, 2015.

Kundu, P., Cohen, I., and Dowling, D.: Fluid Mechanics, Science Direct e-books, Elsevier Science, ISBN 978-0-12-382100-3, https://books.google.cl/books?id=iUo_4tsHQYUC (last access: 15 April 2026), 2012.

Kurganov, A. and Petrova, G.: Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws, Numer. Meth. Part D E., 21, 536–552, https://doi.org/10.1002/num.20049, 2005.

Kurganov, A. and Petrova, G.: A Second-Order Well-Balanced Positivity Preserving Central-Upwind Scheme for the Saint-Venant System, Commun. Math. Sci., 5, 133–160, 2007.

Kurganov, A. and Tadmor, E.: New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection–Diffusion Equations, J. Comput. Phys., 160, 241–282, https://doi.org/10.1006/jcph.2000.6459, 2000.

Kurganov, A. and Xin, R.: New Low-Dissipation Central-Upwind Schemes, J. Sci. Comput., 96, 56, https://doi.org/10.1007/s10915-023-02281-8, 2023.

Kusanovic, D. S., Juan, F., Meza, J., Meneses, R., and Catalan, P.: SWEpy, Zenodo [code], https://doi.org/10.5281/zenodo.16789889, 2025.

Kusanovic, D. S., Juan, F., Meza, J., Meneses, R., and Catalan, P.: SWEpy, Zenodo [code], https://doi.org/10.5281/zenodo.19464287, 2026.

Lam, S. K., Pitrou, A., and Seibert, S.: Numba: a LLVM-based Python JIT compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15, Association for Computing Machinery, New York, NY, USA, ISBN 9781450340052, https://doi.org/10.1145/2833157.2833162, 2015.

LeVeque, R. J.: Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, ISBN 9780511791253, https://doi.org/10.1017/cbo9780511791253, 2002.

Lin, S. C., Wu, T.-R., Yen, E., Chen, H.-Y., Hsu, J., Tsai, Y.-L., Lee, C.-J., and Liu, P. L.-F.: Development of a tsunami early warning system for the South China Sea, Ocean Eng., 100, 1–18, https://doi.org/10.1016/j.oceaneng.2015.02.003, 2015.

Liu, X.: A new well-balanced finite-volume scheme on unstructured triangular grids for two-dimensional two-layer shallow water flows with wet-dry fronts, J. Comput. Phys., 438, 110380, https://doi.org/10.1016/j.jcp.2021.110380, 2021a.

Liu, X.: A new well-balanced finite-volume scheme on unstructured triangular grids for two-dimensional two-layer shallow water flows with wet-dry fronts, J. Comput. Phys., 438, 110380, https://doi.org/10.1016/j.jcp.2021.110380, 2021b.

Liu, X., Albright, J., Epshteyn, Y., and Kurganov, A.: Well-balanced positivity preserving central-upwind scheme with a novel wet/dry reconstruction on triangular grids for the Saint-Venant system, J. Comput. Phys., 374, 213–236, https://doi.org/10.1016/j.jcp.2018.07.038, 2018.

Macías, J., Castro, M. J., Ortega, S., Escalante, C., and González-Vida, J. M.: Performance Benchmarking of Tsunami-HySEA Model for NTHMP's Inundation Mapping Activities, Pure Appl. Geophys., https://doi.org/10.1007/s00024-017-1583-1, 2017.

Meza, J., Kusanovic, D. S., Juan, F., and Meneses, R.: SWEpy: Shallow Water Equation Python solver – User Manual & Technical Reference, Zenodo, https://doi.org/10.5281/zenodo.19323355, 2026.

Morales-Hernández, M., Sharif, M. B., Kalyanapu, A., Ghafoor, S., Dullo, T., Gangrade, S., Kao, S.-C., Norman, M., and Evans, K.: TRITON: A Multi-GPU open source 2D hydrodynamic flood model, Environ. Modell. Softw., 141, 105034, https://doi.org/10.1016/j.envsoft.2021.105034, 2021.

Moukalled, F., Mangani, L., and Darwish, M.: The finite volume method, in: The finite volume method in computational fluid dynamics: An advanced introduction with OpenFOAM® and Matlab, Springer, 103–135, https://doi.org/10.1007/978-3-319-16874-6, 2015.

Moulinec, C., Denis, C., Pham, C.-T., Rougé, D., Hervouet, J.-M., Razafindrakoto, E., Barber, R., Emerson, D., and Gu, X.-J.: TELEMAC: An efficient hydrodynamics suite for massively parallel architectures, Comput. Fluid., 51, 30–34, https://doi.org/10.1016/j.compfluid.2011.07.003, 2011.

National Oceanic and Atmospheric Administration: Deep-Ocean Assessment and Reporting of Tsunamis (DART(R)), NOAA National Centers for Environmental Information [data set], https://doi.org/10.7289/V5F18WNS, 2005.

Nessyahu, H. and Tadmor, E.: Non-oscillatory central differencing for hyperbolic conservation laws, J. Comput. Phys., 87, 408–463, 1990.

Nguyen, T.: Adaptive Central-Upwind Scheme on Triangular Grids for the Shallow Water Model with Variable Density, Int. J. Numer. Anal. Mod., 20, 229–266, https://doi.org/10.4208/ijnam2023-1010, 2023.

Okuta, R., Unno, Y., Nishino, D., Hido, S., and Loomis, C.: CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations, in: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS), http://learningsys.org/nips17/assets/papers/paper_16.pdf (last access: 15 April 2026), 2017.

Parnas, D. L.: On the criteria to be used in decomposing systems into modules, Commun. ACM, 15, 1053–1058, https://doi.org/10.1145/361598.361623, 1972.

Reinarz, A., Charrier, D. E., Bader, M., Bovard, L., Dumbser, M., Duru, K., Fambri, F., Gabriel, A.-A., Gallard, J.-M., Köppel, S., Krenz, L., Rannabauer, L., Rezzolla, L., Samfass, P., Tavelli, M., and Weinzierl, T.: ExaHyPE: An engine for parallel dynamically adaptive simulations of wave problems, Comput. Phys. Commun., 254, 107251, https://doi.org/10.1016/j.cpc.2020.107251, 2020.

Rocklin, M.: Dask: Parallel computation with blocked algorithms and task scheduling, in: Proceedings of the 14th python in science conference, Citeseer, 130–136, https://doi.org/10.25080/Majora-7b98e3ed-013, 2015.

Schubert, J. E., Sanders, B. F., Smith, M. J., and Wright, N. G.: Unstructured mesh generation and landcover-based resistance for hydrodynamic modeling of urban flooding, Adv. Water Resour., 31, 1603–1621, https://doi.org/10.1016/j.advwatres.2008.07.012, 2008.

Shaeri Karimi, S., Saintilan, N., Wen, L., and Valavi, R.: Application of Machine Learning to Model Wetland Inundation Patterns Across a Large Semiarid Floodplain, Water Resour. Res., 55, 8765–8778, https://doi.org/10.1029/2019wr024884, 2019.

Shaw, J., Kesserwani, G., Neal, J., Bates, P., and Sharifian, M. K.: LISFLOOD-FP 8.0: the new discontinuous Galerkin shallow-water solver for multi-core CPUs and GPUs, Geosci. Model Dev., 14, 3577–3602, https://doi.org/10.5194/gmd-14-3577-2021, 2021.

Simons, F., Busse, T., Hou, J., Özgen, I., and Hinkelmann, R.: A model for overland flow and associated processes within the Hydroinformatics Modelling System, J. Hydroinform., 16, 375–391, https://doi.org/10.2166/hydro.2013.173, 2013.

Steinstraesser, J. G. C., Delenne, C., Finaud-Guyot, P., Guinot, V., Casapia, J. L. K., and Rousseau, A.: SW2D-Lemon: A New Software for Upscaled Shallow Water Modeling, Springer Nature Singapore, 23–40, ISBN 9789811916007, https://doi.org/10.1007/978-981-19-1600-7_2, 2022.

Stiernström, V., Lundgren, L., Nazarov, M., and Mattsson, K.: A residual-based artificial viscosity finite difference method for scalar conservation laws, J. Comput. Phys., 430, 110100, https://doi.org/10.1016/j.jcp.2020.110100, 2021.

Sunder, D., Vaghani, D., and Shukla, R.: Third-Order WENO Schemes on Unstructured Meshes, 215–223, ISBN 978-981-15-5182-6, https://doi.org/10.1007/978-981-15-5183-3_23, 2021.

Sweby, P. K.: High resolution schemes using flux limiters for hyperbolic conservation laws, SIAM J. Numer. Anal., 21, 995–1011, 1984.

Synolakis, C. E., Bernard, E. N., Titov, V. V., Kânoğlu, U., and González, F. I.: Validation and Verification of Tsunami Numerical Models, Pure Appl. Geophys., 165, 2197–2228, https://doi.org/10.1007/s00024-004-0427-y, 2008.

Tanaka, S., Bunya, S., Westerink, J. J., Dawson, C., and Luettich, R. A.: Scalability of an Unstructured Grid Continuous Galerkin Based Hurricane Storm Surge Model, J. Sci. Comput., 46, 329–358, https://doi.org/10.1007/s10915-010-9402-1, 2010.

Titov, V., Kânoğlu, U., and Synolakis, C.: Development of MOST for Real-Time Tsunami Forecasting, J. Waterw. Port C., 142, https://doi.org/10.1061/(asce)ww.1943-5460.0000357, 2016.

Toro, E. F.: Shock-capturing methods for free-surface shallow flows, John Wiley, Chichester, New York, ISBN 978-0-471-98766-6, 2001.

Toro, E. F., Spruce, M., and Speares, W.: Restoration of the contact surface in the HLL-Riemann solver, Shock Waves, 4, 25–34, https://doi.org/10.1007/bf01414629, 1994.

Turner, A. and Wouters, T.: What's new in python 3.13, Python Software Foundation, https://docs.python.org/3/whatsnew/3.13.html (last access: 15 April 2026), 2024.

United Nations: World Urbanization Prospects: The 2018 Revision, ST/ESA/SER.A/420, United Nations, New York, United Nations Publication, ISBN 978-92-1-148319-2, https://population.un.org/wup/assets/WUP2018-Report.pdf (last access: 15 April 2026), 2019.

Vacondio, R., Dal Palù, A., and Mignosa, P.: GPU-enhanced Finite Volume Shallow Water solver for fast flood simulations, Environ. Modell. Softw., 57, 60–75, https://doi.org/10.1016/j.envsoft.2014.02.003, 2014.

Van Leer, B.: Towards the ultimate conservative difference scheme, J. Comput. Phys., 135, 229–248, 1997.

Vreugdenhil, C. B.: Numerical Methods for Shallow-Water Flow, Springer Netherlands, ISBN 9789401583541, https://doi.org/10.1007/978-94-015-8354-1, 1994.

Wang, X. and Power, W.: COMCOT: a Tsunami Generation Propagation and Run-up Model, GNS Science Report 2011/43, GNS Science, https://shop.gns.cri.nz/sr_2011-043-pdf/ (last access: 15 April 2026), 2011.

Xia, X., Liang, Q., and Ming, X.: A full-scale fluvial flood modelling framework based on a high-performance integrated hydrodynamic modelling system (HiPIMS), Adv. Water Resour., 132, 103392, https://doi.org/10.1016/j.advwatres.2019.103392, 2019.

Xie, W.-X., Cai, L., Feng, J.-H., and Xu, W.: Computations of shallow water equations with high-order central-upwind schemes on triangular meshes, Appl. Math. Comput., 170, 296–313, 2005.

Zhang, Y.-T. and Shu, C.-W.: ENO and WENO Schemes, Elsevier, 103–122, https://doi.org/10.1016/bs.hna.2016.09.009, 2016.

Zhou, Y., Wu, W., Nathan, R., and Wang, Q. J.: Deep Learning-Based Rapid Flood Inundation Modeling for Flat Floodplains With Complex Flow Paths, Water Resour. Res., 58, https://doi.org/10.1029/2022wr033214, 2022.

Zhu, J. and Qiu, J.: New Finite Volume Weighted Essentially Nonoscillatory Schemes on Triangular Meshes, SIAM J. Sci. Comput., 40, A903–A928, https://doi.org/10.1137/17M1112790, 2018.