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 Catalan[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)

Abstract. We present SWEpy, a new Python GPU-accelerated open-source finite volume (FV) software designed to solve the Saint-Venant system of shallow water equations (SWE) on unstructured triangular grids. SWEpy is designed for flexibility and performance, considering a well-balanced, positivity-preserving, and higher-order central-upwind FV scheme, intended to solve tsunami wave propagation, flooding, and dam-break scenarios, among others.

In this regard, we enhance the minimization of numerical diffusion, a phenomenon frequently found in this sort of FV schemes, by using a second-order WENO reconstruction operator as well as a third-order strong stability-preserving Runge-Kutta time integrator. With this in mind, a modular software architecture is presented that can support a range of initial and boundary conditions and source terms.

SWEpy's performance, stability, and accuracy are verified using canonical benchmarks, including Synolakis' conical island and Bryson's flow over a Gaussian bump, and further demonstrated in large-scale simulations of the 1959 Malpasset Dam failure and the Mw8.8 2010 Maule tsunami. SWEpy delivers high-resolution results on consumer-grade hardware, offering a user-friendly platform for both research and operational forecasting.

1 Introduction

Accurate simulation of hazardous hydrological events, such as dam failures, tsunamis, and urban floods, plays a critical role in 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). Over the years, several numerical tools have been developed for these purposes, ranging from open-source models to commercial software (ANSYS, Inc., 2013; Jodhani et al., 2023), enabling stakeholders to generate reliable data for evaluating community vulnerabilities. However, the increase of global challenges, including rapid urbanization, climate change, and socio-economic uncertainties–particularly in developing regions (United Nations, 2019)–is imposing new demands on risk management frameworks. These evolving needs require advanced modeling

tools that are not only accurate and efficient but also flexible, scalable, and accessible to diverse users in research, planning, and operational contexts.

In order to address these emerging challenges, open-source modeling tools are invaluable in advancing research and operational applications. They promote collaborative development, enabling diverse contributors to improve and adapt models while ensuring broad accessibility without commercial barriers. To illustrate the current landscape, we have compiled ~~a representative overview of~~ a representative but not exhaustive set of freely available shallow water equation (SWE) solvers in Table 1, highlighting their key features, including numerical schemes, grid types, and parallelization capabilities among others. As shown, while many excel in specific domains such as rainfall-runoff or flooding, there remains a gap in flexible solvers that combine unstructured triangular grids with high-order reconstructions for improved accuracy in complex ~~scenarios~~geometries.

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)	No	MPI	Open-source (BSD)	General purpose	FV wave-propagation (Godunov-type)	Adaptive Cartesian
HEC-RAS 2D	Brunner (2021)	Partial*	–	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**	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.

Table 1. Overview of some of the openly available shallow water equation (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. *: GPU support in development (HEC-RAS 2025 Alpha). **: Generation, Propagation, Inundation.

These programs solve nonlinear SWEs, which have become the cornerstone of two-dimensional free-surface flow modeling across scales and applications (Delis and Nikolos, 2021). However, increasing demands for high spatial resolution and real-time performance have driven some efforts toward simplified formulations (Courty et al., 2017) or machine-learning surrogates (Kabir et al., 2020; Zhou et al., 2022; Shaeri Karimi et al., 2019) to mitigate computational costs ~~—approaches that~~ which may compromise physical accuracy (Fernández-Pato et al., 2018). ~~A~~ On the other hand, a more robust strategy involves parallelization, leveraging high-performance computing on CPUs/GPUs ~~(Caviedes-Voullième et al., 2023; Morales-Hernández et al., 2021)~~ (Caviedes-Voullième et al., 2023; Morales-Hernández et al., 2021; Reinarz et al., 2020), achieving speedups of orders of magnitude. ~~However~~

Several mature SWE solvers have demonstrated high performance and robustness in large-scale, real-world applications. Tools such as 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), to name a few, exploit parallelization on CPUs and GPUs via OpenMP, MPI, and hybrid approaches, using either fixed meshes or adaptive mesh refinement (AMR) to efficiently capture localized dynamics. Their deployment in tsunami early-warning systems, coastal hazard analysis, and continental-scale flooding highlights the importance of scalable high-performance computing for accurate SWE simulations.

Despite their performance, many of these solvers rely on low-level languages (e.g., FORTRAN, C++, C) and APIs (e.g., CUDA, OpenMP), limiting accessibility for users without specialized expertise. To overcome this barrier, high-level languages like Python offer a viable alternative, although traditional parallelization has been challenging due to issues such as the Global Interpreter Lock (Turner and Wouters, 2024). This is usually addressed using libraries like Numba (Lam et al., 2015), PyCUDA (Kloeckner et al., 2025), TensorFlow (Abadi et al., 2015), PyTorch (Ansel et al., 2024), and Dask (Rocklin, 2015), which enable GPU-based parallelization via Single Instruction Multiple Data (SIMD) techniques, or "compiled loop" parallelization. ~~These libraries, although powerful, primarily operate via decorators and wrappers, which can make achieving modularity a bit unclear. In addition, the last three options are oriented towards machine learning and big data. In contrast, CuPy (Okuta et al., 2017) serves as a drop-in replacement for NumPy (Harris et al., 2020), handling array calculations with CUDA kernels to provide seamless, user-friendly GPU acceleration—versatile enough for custom kernels and even multi-GPU setups—while democratizing advanced modeling without sacrificing efficiency.~~

~~Although almost all~~ In contrast, CuPy (Okuta et al., 2017) serves as a drop-in replacement for NumPy (Harris et al., 2020), executing array operations through NVIDIA CUDA kernels to deliver seamless and user-friendly GPU acceleration. By relying on the mature and stable CUDA ecosystem, CuPy enables reproducible high-performance execution, support for custom kernels, and scalable multi-GPU workflows without exposing users to 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 of the reviewed solvers are parallelized in some form, there is considerable variability in their scope, parallelization strategies (e.g., CUDA, Kokkos, MPI, or OpenMP), numerical schemes, grid geometries, and programming languages. ~~For instance, while some~~ Some solvers are specialized for ~~rainfall-runoff~~ rainfall–runoff processes (e.g., SERGHEI-SWE, HiPIMS) or tsunami ~~generation-propagation-inundation~~ generation–propagation–inundation (e.g., COMCOT, Tsunami-

HySEA, TsunAWI), ~~others serve~~ while others target more general-purpose applications ~~like~~ such as flooding or channel flows. This diversity reflects inherent design trade-offs~~in design~~ , but also ~~highlights gaps in tools that integrate high~~ reveals gaps in solvers that combine flexibility across these dimensions. ~~The majority of these models rely on structured Cartesian grids due to their computational and implementation simplicity. However, unstructured grids, particularly triangular ones, offer significant advantages by allowing seamless local refinement (Schubert et al., 2008) in regions with complex bathymetry or where higher resolution is needed, such as near coastlines or urban structures.~~

A particularly influential factor is grid discretization. Many SWE solvers rely on structured Cartesian grids due to their computational efficiency and ease of implementation. However, both structured and unstructured discretizations can benefit from adaptive refinement strategies, such as adaptive mesh refinement (AMR) on Cartesian grids or local h/p-adaptivity on unstructured meshes. Others focus on unstructured triangular grids because they offer intrinsic geometric flexibility, enabling seamless local refinement in regions of complex bathymetry, irregular coastlines, or urban features without requiring hierarchical grid structures or nested mesh management (Schubert et al., 2008). This feature is particularly advantageous for multi-scale problems, such as tsunami propagation coupled with near-shore inundation, where localized resolution enhancement is essential. This adaptability avoids the need for model coupling or nested grids in multi-scale simulations, like those involving flooding and tsunami propagation (Harig et al., 2008; Bomers et al., 2019).

~~A common feature among the reviewed solvers is the finite volume method (FVM), particularly Godunov-type schemes, valued for their inherently conservative derivation (LeVeque, 2002). These schemes excel at capturing shock-wave discontinuities (Toro, 2001), making them ideal for simulating flows across diverse regimes and complex phenomena such as bore propagation or dam-break events. However, a significant challenge in these Godunov-type approaches is their reliance on Riemann solvers, such as the Roe (Roe, 1981) or Harten-Lax-van Leer (Toro et al., 1994) formulations, which resolve discontinuities at cell interfaces by approximating solutions to the associated eigenvalue problem (local wave propagation speeds) to construct numerical fluxes. This dependency increases computational complexity and can limit extensibility to higher-order schemes. An alternative approach, though less common~~A common feature among many SWE solvers is the use of finite volume Godunov-type schemes, which rely on approximate Riemann solvers such as Roe, HLL, or HLLC formulations. These solvers are well-established, robust, and widely optimized for large-scale applications, offering accurate resolution of wave propagation and discontinuities. An alternative class of methods is provided by central-upwind schemes, which avoid explicit Riemann problem solutions by estimating local propagation speeds to construct numerical fluxes. While not replacing Riemann-solver-based approaches, central-upwind formulations offer a conceptually simpler flux construction and can be attractive in settings where implementation flexibility, avoidance of characteristic decompositions, or compatibility with high-order reconstructions on unstructured grids are desirable. In particular, though less widely adopted, is the ~~use~~ class of central-upwind (CU) schemes, ~~initially~~ first introduced for Cartesian grids ~~in the seminal work~~ by Kurganov and Tadmor (2000) and later extended to triangular grids by Kurganov and Petrova (2005). These schemes bypass Riemann solvers by integrating the equations over Riemann fans sized by estimated local propagation speeds. This offers a simpler, yet robust framework, extensible to higher orders via polynomial reconstruction operators. However, existing CU formulations for the SWE often suffer from numerical diffusion, with recent efforts to address this shifting toward Cartesian-grid finite-difference methods rather than enhancing triangular FV

schemes (Kurganov and Xin, 2023; Chu et al., 2025; Cui et al., 2025). Addressing this issue without diverging from the original FV CU formulation could be achieved by exploiting the mentioned high order polynomial extensibility, but such reconstruction operators–like those in the ENO/WENO family (Zhang and Shu, 2016)–demand a lot of computational power due to algorithms involving smoothness indicator calculations, and large stencils, possibly involving complex neighbor searching. In this regard, high performance computing becomes an attractive option. While the computational overhead of high-order WENO schemes can be prohibitive for serial CPU execution, their high arithmetic intensity makes them ideal candidates for GPU acceleration.

Furthermore, the nature of the problem is determined by the source terms in the balance equations, as well as domain characteristics including bathymetry and boundary conditions. Given the diversity of models represented by the SWEs, varied criteria are required for their formulation and analysis. In this context, alongside numerical flux structures, spatial discretization designs must satisfy both physical and numerical requirements, such as exact equilibrium (well-balancing) and positivity preservation, extensively discussed in the literature (e.g., Kurganov and Tadmor (2000); Kurganov and Petrova (2005); Toro et al. (1994) to name a few). Well-balancing ensures that steady states–like lake-at-rest over variable topography or geostrophic equilibrium where pressure gradients counter Coriolis forces–are preserved exactly, preventing spurious oscillations that could undermine simulation accuracy in long-term or large-scale flows. Positivity preservation maintains non-negative water depths at wet/dry fronts, crucial for realistic inundation modeling without nonphysical negative heights or instabilities. Moreover, defining how the computed values align with scheme evaluations necessitates reconstruction operators that effectively address these constraints.

~~In this study, we present SWEpy, a solver designed to overcome these challenges by enabling versatile near- and far-field modeling on unstructured triangular grids. To bridge these gaps, we present SWEpy, an open-source Python-based FV solver for SWEs on unstructured triangular grids with GPU capabilities. Our primary contributions include:~~ In this study, we introduce SWEpy, an open-source, Python-based finite-volume solver for the shallow water equations on unstructured triangular grids with GPU acceleration. SWEpy is designed to address the limitations of existing approaches by enabling efficient and flexible near- and far-field modeling within a unified framework. The primary contributions of this work are: (1) the implementation of a central-upwind scheme extended to higher-order via quadratic WENO spatial reconstruction and third-order Strong-Stability-Preserving (SSP) Runge-Kutta time integration, reducing diffusion for versatile near-field (shocks/wet-dry) and far-field (wave propagation) simulations; (2) ~~GPL licensing with modular architecture for reproducibility and community extensions~~ released under the GNU General Public License (GPL) to ensure that derivative works remain openly available and to promote transparent, community-driven scientific development (e.g., infiltration/transport); and (3) Python/CuPy implementation for accessible GPU acceleration on CUDA-compatible devices, overcoming low-level barriers while enabling rapid computations on consumer hardware. The solver is well-balanced in wet domains and positivity-preserving, validated across benchmarks, and real cases such as Malpasset dam-break for near-field inundation and the Maule 2010 tsunami across the Pacific basin for far-field propagation. This paper is structured as follows: Section 2 details governing equations and FV foundations; Section 3 describes the CU scheme and reconstructions; Section 4 details Python/CuPy architecture; Section 5 presents validation results and performance analysis; Section 6 discusses implications, limitations, and future directions.

2 Problem Setting

This section presents the theoretical foundation of the model, defines the physical problem, and introduces the semi-discrete form of the SWEs, which underpins the `SWEpy` framework's central-upwind scheme with WENO reconstructions on unstructured triangular grids.

2.1 Governing Equations

The Shallow Water Equations, first proposed in one dimension by Saint-Venant in 1871, are derived for the two-dimensional case by applying the hydrostatic assumption to the Reynolds-Averaged Navier-Stokes equations. Using a scaling argument and assuming fluid incompressibility, the vertical momentum equation yields a horizontal pressure gradient, enabling depth-averaging of the continuity and momentum equations from the solid bottom to the free surface (Castro-Orgaz et al., 2019; Chow, 1971; Hervouet, 2007; Vreugdenhil, 1994). These equations form the cornerstone of computational models for simulating free-surface flows in rivers, coastal areas, and urban floodplains, capturing phenomena like flood waves, tsunamis, and storm surges (Delis and Nikolos, 2021), as validated in our benchmarks (Section 5).

To ensure consistent notation, we define the state vector $\boldsymbol{q} = (h, hu, hv)^\top$, where $h(x,y,t)$ is the water depth from the bathymetry $B(x,y)$ to the free surface $w(x,y,t)$, measured relative to the $z = 0$ xy-plane, $u(x,y,t)$ and $v(x,y,t)$ are depth-averaged velocities in the x and y directions, and $hu(x,y,t)$ and $hv(x,y,t)$ are the corresponding flux-discharges. Scalar variables (e.g., h) use regular symbols, while vectors (e.g., $\boldsymbol{q}$) are bold. These variables are illustrated in Figure 1.

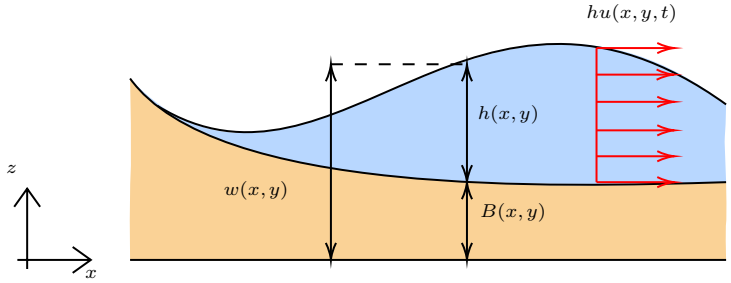

Figure 1. Model setting and physical variables used in the shallow water equations.

With this notation, the SWEs are expressed in conserved vector form as:

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x + \boldsymbol{g}(\boldsymbol{q})_y = \boldsymbol{S_B}(\boldsymbol{q}) + \boldsymbol{S}(\boldsymbol{q}) \tag{1}$$

where the subscripts $(\cdot)_t$, $(\cdot)_x$, and $(\cdot)_y$ denote partial differentiation with respect to time t, and spatial coordinates x and y, respectively; furthermore, the vector components defined earlier are

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

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

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

$$\boldsymbol{S_B}(\boldsymbol{q}) = (0, -ghB_x, -ghB_y)^\top, \tag{5}$$

and $\boldsymbol{S}(\boldsymbol{q})$ represents additional source terms, such as Coriolis, friction, rheology, or turbulence. In this work, we focus on bottom friction $\boldsymbol{S_F}(\boldsymbol{q})$ and Coriolis effects $\boldsymbol{S_C}(\boldsymbol{q})$, critical for modeling dam-break scenarios and far-field tsunami propagation, respectively. Thus, the bottom friction–requiring semi-empirical closure laws, specifically the Manning-Strickler parameterization (Chow, 1971)– and coriolis terms are expressed as:

$$\boldsymbol{S_C}(\boldsymbol{q}) = (0, fhv, -fhu)^\top, \tag{6}$$

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

where n is 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 $\boldsymbol{S_B}$ and $\boldsymbol{S_C}$ are considered, while flooding scenarios use $\boldsymbol{S_B}$ and $\boldsymbol{S_F}$, as validated in Section 5. Solving these nonlinear hyperbolic equations requires robust numerical methods to capture shocks and ensure stability, particularly for complex geometries and wet/dry interfaces. `SWEpy`'s central-upwind scheme on unstructured triangular grids, detailed in Section 3, addresses these challenges to achieve high accuracy.

2.2 Semi-discrete formulation

To ensure conservation of mass and momentum while effectively handling discontinuities such as shocks or wet/dry fronts prevalent in shallow water flows, FVM offer a robust numerical framework for solving the SWEs (LeVeque, 2002; Moukalled et al., 2015; Toro, 2001). These methods integrate the governing equations over finite control volumes, approximating cell-averaged states and evolving them by computing fluxes across cell interfaces. This approach naturally captures shock waves, like hydraulic jumps or bore propagation, without requiring additional artificial viscosity (Stiernström et al., 2021), making it particularly suited for shallow water applications where abrupt changes in flow regime are common.

For enhanced well-balancing—ensuring exact preservation of steady states, especially over variable bathymetry—we reformulate the equations by substituting the water depth h with the surface elevation $w = h + B$ in the conserved variables vector $\boldsymbol{q}$, resulting in $\boldsymbol{q}(x, y, t) = (w, hu, hv)^T$. This substitution avoids numerical imbalances caused by topography gradients, transforming the system into:

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

with

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

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

We consider a triangular discretization of the polygonal spatial domain $\Omega = \bigcup_j \Omega_j$, including additional "ghost" cells for boundary conditions, as depicted in Figure 2 (bottom border).

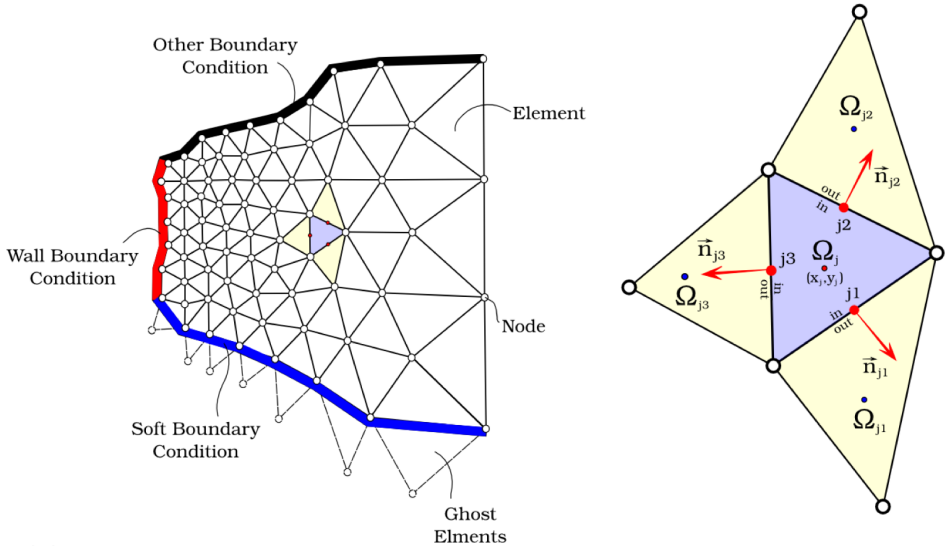

Figure 2. Illustration of a triangular unstructured grid. The figure shows on the left an example of a finite volume grid, while on the right a typical triangular cell with some attributes used in the semi-discrete formulation

Integrating (8) over each cell Ω_j and applying the Gauss divergence theorem yields:

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

Let Ω_{jk} $(k = 1,2,3)$ denote the neighboring cells of Ω_j, with (x_j, y_j) as the barycenter coordinates, Γ_{jk} the side toward Ω_{jk}, l_{jk} as its length, and $\hat{n}_{jk} = \cos(\theta_{jk})\hat{i} + \sin(\theta_{jk})\hat{j}$ as its outward normal vector, where θ_{jk} is its angle.

The semi-discrete formulation becomes:

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

where $\overline{\boldsymbol{q}}_j = \frac{1}{|\Omega j|}\int_{\Omega_j} \boldsymbol{q}(x,y,t)\, d\Omega$ is the average state over the cell, and $\mathcal{F}_{jk}$ is the numerical flux along segment Γ_{jk}, capturing interface interactions. This flux is derived from fields $\boldsymbol{F}$ and $\boldsymbol{G}$ in equation (9). The scheme relies on approximations of

$\boldsymbol{q}$ (and bathymetry B) at Γ_{jk}, denoted $\boldsymbol{q}_{jk}^{in}$ and $\boldsymbol{q}_{jk}^{out}$, and obtained using reconstruction operators. Integrations employ Gaussian quadrature, with the number of Gauss points determined by the degree of this reconstruction for the flow variables. As nodal values are time-dependent, equation (12) yields a system of ordinary differential equations, setting the stage for the central-upwind discretization detailed in Section 3. Terms $\overline{\boldsymbol{S}_{Bj}}$ and $\overline{\boldsymbol{S}_j}$ are each careful discretizations of the source terms, to be discussed in detail moving forward.

3 `SWEpy` Numerical Model

3.1 Central-Upwind Numerical Fluxes on Triangular Grids

Building on the semi-discrete formulation, `SWEpy` implements the central-upwind (CU) finite volume scheme for hyperbolic conservation laws on triangular grids, as originally proposed by Kurganov and Petrova (2005) and in parallel by Bryson and Levy (2005) who studied the well-balancing condition, and Xie et al. (2005) in the same direction, but with a modified flux formulation; and refined in later works. This Riemann-solver-free method offers a balance between computational simplicity and robustness, estimating local propagation speeds to stabilize fluxes without solving the full Riemann problem, making it well-suited for shallow water flows over unstructured grids with variable topography and wet/dry interfaces.

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

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

where M_{jk}^s are the scaled Gaussian quadrature points along the edge, and c_s are the associated weights. The number of points N_s depends on the reconstruction order, ensuring accurate integration of higher-degree polynomials. The terms a_{jk}^{in} and a_{jk}^{out} represent the inward and outward local propagation speeds, detailed below.

The projection components are:

$$F_{jk} = \frac{1}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \sum_{s=1}^{N_s} c_s \left[a_{jk}^{\text{in}} \boldsymbol{F}\left(\boldsymbol{q}_{jk}^{\text{in}}(M_{jk}^s), B(M_{jk}^s)\right) + a_{jk}^{\text{out}} \boldsymbol{F}\left(\boldsymbol{q}_{jk}^{\text{out}}(M_{jk}^s), B(M_{jk}^s)\right)\right], \tag{14}$$

$$G_{jk} = \frac{1}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \sum_{s=1}^{N_s} c_s \left[a_{jk}^{\text{in}} \boldsymbol{G}\left(\boldsymbol{q}_{jk}^{\text{in}}(M_{jk}^s), B(M_{jk}^s)\right) + a_{jk}^{\text{out}} \boldsymbol{G}\left(\boldsymbol{q}_{jk}^{\text{out}}(M_{jk}^s), B(M_{jk}^s)\right)\right]. \tag{15}$$

Here, $\boldsymbol{q}_{jk}^{\text{out}}$ represents the limit $\boldsymbol{q}(x,y) \to \boldsymbol{q}_{jk}^{\text{out}}(M_{jk}^s)$ as $(x,y) \in \Omega_j$, while $\boldsymbol{q}_{jk}^{\text{in}}$ is the limit $\boldsymbol{q}(x,y) \to \boldsymbol{q}_{jk}^{\text{in}}(M_{jk}^s)$ as $(x,y) \in \Omega_{jk}$. Furthermore, since divisions by zero may appear near wet/dry fronts given the form of F_{jk} and G_{jk}, an adequate treatment of the fluxes is required to avoid these singularities as addressed in the positivity-preserving reconstruction (Section 3.4). Then,

substituting (14) and (15) into (13) and integrating into the semi-discrete scheme (12) results in:

$$\begin{aligned}\frac{d\overline{\boldsymbol{q}}_j}{dt} =& -\frac{1}{|\Omega_j|}\sum_{k=1}^{3}\frac{l_{jk}\cos\theta_{jk}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\sum_{s=1}^{N_s}c_s\left[a_{jk}^{\text{in}}\boldsymbol{F}\left(\boldsymbol{q}_{jk}(M_{jk}^s),B(M_{jk}^s)\right)+a_{jk}^{\text{out}}\boldsymbol{F}\left(\boldsymbol{q}_j(M_{jk}^s),B(M_{jk}^s)\right)\right]\\ &-\frac{1}{|\Omega_j|}\sum_{k=1}^{3}\frac{l_{jk}\sin\theta_{jk}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\sum_{s=1}^{N_s}c_s\left[a_{jk}^{\text{in}}\boldsymbol{G}\left(\boldsymbol{q}_{jk}(M_{jk}^s),B(M_{jk}^s)\right)+a_{jk}^{\text{out}}\boldsymbol{G}\left(\boldsymbol{q}_j(M_{jk}^s),B(M_{jk}^s)\right)\right]\\ &+\frac{1}{|\Omega_j|}\sum_{k=1}^{3}l_{jk}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\sum_{s=1}^{N_s}c_s\left[\boldsymbol{q}_{jk}(M_{jk}^s)-\boldsymbol{q}_j(M_{jk}^s)\right]+\overline{\boldsymbol{S}_{\boldsymbol{B}}}_j+\overline{\boldsymbol{S}}_j,\end{aligned} \tag{16}$$

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

To define the one-sided local speeds a_{jk}^{in} and a_{jk}^{out}, which represent the maximum wave speeds at which information propagates inward or outward across the jk interface, we first compute desingularized velocities at the Gaussian points to avoid singularities near dry states:

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

where $h=w-B$ is the water depth, and ε is a small tolerance parameter to prevent singularities. These equations represent the structure to be used with the respective reconstruction operators of the variables evaluated at the necessary points. These velocities are then projected onto the edge normal:

$$u_j^{\theta}(M_{jk}^s)=\cos(\theta_{jk})u_j(M_{jk}^s)+\sin(\theta_{jk})v_j(M_{jk}^s)\ ;\ u_{jk}^{\theta}(M_{jk}^s)=\cos(\theta_{jk})u_{jk}(M_{jk}^s)+\sin(\theta_{jk})v_{jk}(M_{jk}^s)$$

using reconstructions from cells j and its neighbor jk. The local speeds are then determined as the extrema over the Gaussian points:

$$\begin{aligned}a_{jk}^{\text{out}}&=\max_s\left\{\max\left\{u_j^{\theta}(M_{jk}^s)+\sqrt{gh_j(M_{jk}^s)},\ u_{jk}^{\theta}(M_{jk}^s)+\sqrt{gh_{jk}(M_{jk}^s)},\ 0\right\}\right\},\\ a_{jk}^{\text{in}}&=-\min_s\left\{\min\left\{u_j^{\theta}(M_{jk}^s)-\sqrt{gh_j(M_{jk}^s)},\ u_{jk}^{\theta}(M_{jk}^s)-\sqrt{gh_{jk}(M_{jk}^s)},\ 0\right\}\right\}.\end{aligned} \tag{18}$$

We remark that points M_{jk}^s are a number of Gaussian points that depend on the degree of the reconstruction used. In our case, $N_s=1$ for the linear reconstruction and $N_s=2$ for the quadratic case.

3.2 Well-balancing of the source terms

A key requirement for robust SWE solvers, particularly in applications involving complex topography like dam-breaks and tsunamis, is well-balancing: the exact preservation of steady-state solutions without introducing artificial oscillations. This property is essential to maintain physical accuracy in lake-at-rest scenarios or geostrophic equilibria, where source terms must 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). In `SWEpy`, we achieve well-balancing through careful discretization of the source terms, ensuring numerical fluxes align with physical conditions across both fully wet domains and variable bathymetry.

3.2.1 Bathymetry gradient contribution

For the bathymetry source term, we derive a balanced discretization by assuming lake-at-rest conditions and equating it to the momentum flux contributions, as described in Bryson et al. (2011). Using the polynomial reconstructions of the variables, we integrate over the cell interior and its edges via Gaussian quadrature, yielding a general form applicable to arbitrary reconstruction orders:

$$\begin{aligned}
\overline{\boldsymbol{S}_{\boldsymbol{B}}}_j^{(l)} &= -g \sum_{s=1}^{N_s^{int}} c_s \frac{\partial w(M_j^s)}{\partial x} \big(w_j(M_j^s) - B(M_j^s)\big) + \frac{g}{2|\Omega_j|} \sum_{k=1}^{3} \sum_{s=1}^{N_s} c_s l_{jk} (w_j(M_{jk}^s) - B(M_{jk}^s))^2 \cos\theta_{jk}, \\
\overline{\boldsymbol{S}_{\boldsymbol{B}}}_j^{(3)} &= -g \sum_{s=1}^{N_s^{int}} c_s \frac{\partial w(M_j^s)}{\partial y} \big(w_j(M_j^s) - B(M_j^s)\big) + \frac{g}{2|\Omega_j|} \sum_{k=1}^{3} \sum_{s=1}^{N_s} c_s l_{jk} (w_j(M_{jk}^s) - B(M_{jk}^s))^2 \sin\theta_{jk},
\end{aligned} \tag{19}$$

where N_s^{int} is the number of Gaussian points M_j^s for quadrature over the cell interior Ω_j, B is the reconstructed bottom via a same-order operator as the one used for the variables, and c_s are the corresponding weights. This formulation ensures the source term discretization mirrors the flux contributions, preventing spurious flows over uneven topography and maintaining equilibrium states critical for realistic simulations, as demonstrated in our steady-state benchmarks (Section 5).

3.2.2 Manning friction

The Manning friction source term's structure makes it rather straightforward to discretize in a well-balanced manner, since it is proportional to the discharge components $\overline{hu}$ and $\overline{hv}$; thus, the term vanishes identically, preserving equilibrium without further modifications. However, near wet/dry fronts or in low-depth regions where $h \to 0$, desingularization is required to avoid division by zero and ensure numerical stability. Following (Chertock et al., 2015), we introduce the discrete friction coefficient:

$$\mathcal{G}(\overline{\boldsymbol{q}}_j) := -gn^2 \left(\frac{2\overline{h}_j}{\overline{h}_j^2 + \max\left(\overline{h}_j^2, \varepsilon^2\right)} \right)^{7/3} \sqrt{(\overline{hu})_j^2 + (\overline{hv})_j^2}, \tag{20}$$

where $\overline{h}_j = \overline{w}_j - \overline{B_j}$, and ε as indicated before, yielding the discretized friction term as:

$$\overline{\boldsymbol{S}_{\boldsymbol{F}}}_j = \mathcal{G}(\overline{\boldsymbol{q}}_j) \begin{bmatrix} 0 & (\overline{hu})_j & (\overline{hv})_j \end{bmatrix}^T. \tag{21}$$

This formulation is incorporated semi-implicitly in time integration (Section 3.5) to handle the stiffness of the friction source term (Chertock et al., 2015). In `SWEpy`, $\mathcal{G}$ is computed vectorially across all cells on the GPU, enabling efficient parallel evaluation even for large grids.

Although it may vary across the domain, in our experiments the n coefficient is set as a constant for the whole grid (cf. sect. 5.2)

3.2.3 Coriolis

The Coriolis source term vanishes identically under zero-discharge equilibria where $hu = hv = 0$, ensuring inherent well-balancing in friction-dominated regimes, requiring no additional discretization strategies beyond direct averaging:

$$\overline{(\boldsymbol{S_C})}_j = f \begin{bmatrix} 0 & (\overline{hv})_j & -(\overline{hu})_j \end{bmatrix}^\top, \tag{22}$$

where f is the Coriolis parameter, typically approximated as $10^{-4}, \mathrm{s}^{-1}$ in mid-latitude regions (Kundu et al., 2012), as used in our Maule 2010 tsunami validation (Section 5.2.2). However, for geophysical flows such as large-scale oceanic or atmospheric circulations–relevant to the far-field tsunami propagation modeled in `SWEpy`–a more subtle form of well-balancing, known as geostrophic balance, is often required. This non-static steady state equilibrates horizontal pressure gradients with Coriolis forces. Recent schemes have addressed this for rotating SWEs through distinct approaches (Desveaux and Masset, 2022; Chertock et al., 2018). However, `SWEpy`'s central-upwind framework employs standard balancing, with potential for extensions to enhance geostrophic fidelity in future developments.

3.3 Spatial Reconstruction Operators and Scheme Formulation

This section presents the methodologies for spatial reconstruction of flow variables within `SWEpy`'s central-upwind finite volume scheme, tailored to address the physical and numerical demands of shallow water equations (SWE) on unstructured triangular grids. The reconstruction approach is influenced by problem-specific features—such as variable bathymetry, roughness, and domain extent—which dictate the need for accurate approximations to capture gradients and discontinuities, particularly in near-field shocks and far-field wave propagation validated in Section 5. These approximations must satisfy critical properties: well-balancing, ensuring exact preservation of steady states (e.g., lake-at-rest or geostrophic equilibria over complex topography) to avoid spurious oscillations (Bryson et al., 2011), and positivity preservation, maintaining non-negative water depths at wet/dry fronts to ensure physical realism in inundation scenarios. Numerical experiments with long-range tsunami waves (Section 5) revealed that constant and linear reconstructions introduce excessive diffusion, compromising wave height accuracy, thus motivating the development of higher-order operators. The reconstruction operators are defined as piecewise polynomials over each cell Ω_j, expressed as:

$$\tilde{q}_j(x,y) = \overline{q}_j + p_j^q(x,y), \tag{23}$$

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.

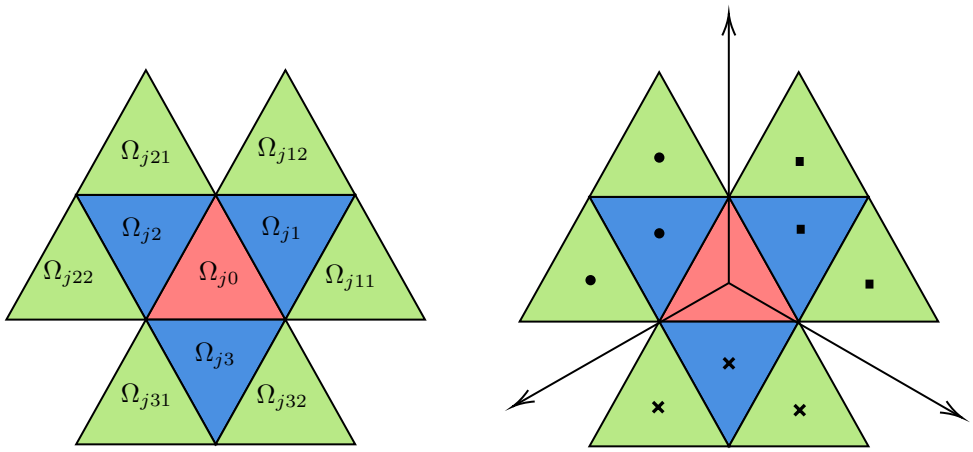

Figure 3. Stencil illustration for the j-th cell (left) and its sectorial division (right). The blue triangles represent first-order neighbors Ω_{jk}, while the green triangles denote second-order neighbors Ω_{jkl}.

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 utilize the first group Ω_{ji}, while quadratic reconstructions (employing two Gaussian points per edge) incorporate all cells of the big stencil Ω_{jkl}. The right panel of Figure 3 depicts the selection process for these sub-stencils, illustrating how barycenters of the chosen cells are constrained to lie within cones formed by lines connecting the reference cell's barycenter to its vertices.

3.3.1 Linear Piecewise Reconstruction with Minmod Gradient Limiter

In this context, the general form of the interpolator (23) is given by:

$$\tilde{q}_j(x,y) = \overline{q}_j + (q_x)_j(x-x_j) + (q_y)_j(y-y_j), \tag{24}$$

with $Dq_j = ((q_x)_j,(qy)_j)$ denoting the numerical gradient. The selection criterion for this gradient determines the reconstruction and responds to simulation needs. The variety of selection methods is extensive, as seen in classical approaches (Nessyahu and Tadmor, 1990; Sweby, 1984; Van Leer, 1997), finite volume treatments (Arminjon and St-Cyr, 2003; Christov and Popov, 2008; Jawahar and Kamath, 2000; LeVeque, 2002), and central-upwind schemes for Saint-Venant systems (Bryson et al., 2011; Kurganov and Petrova, 2005). In SWEpy's implementation, we follow Bryson et al. (2011) by constructing three conservative interpolating polynomials $L^j_{k,l}(x,y)$ over Ω_j and pairs $\Omega_{j,k}$, $\Omega_{j,l}$ (see Figure 3). With $\theta \in [1,2]$, define ${q'}_j = \theta\text{minmod}\{\nabla L^j_{kl}\}$. If substituting ${q'}_j$ in (24) causes midpoints to exceed local extrema, a constant plane through the cell's mean value is imposed; otherwise, $Dq_j = {q'}_j$. This ensures monotonicity and supports well-balancing by aligning with source term discretizations, though it may introduce diffusion in smooth regions, as observed in Section 5. Details of the procedure

are synthesized in Algorithm A1 in the appendices.

The minmod operator, adapted from Bryson et al. (2011), constructs a piecewise linear interpolant using the cell and two neighbors, minimizing the magnitude of the gradient unless the midpoints exceed the local extrema, in which case a constant value is imposed. The formulation is given as:

$$\tilde{u}_j = \overline{u}_j + \phi_x^{lin}(x - x_j) + \phi_y^{lin}(y - y_j), \tag{25}$$

where $(\phi_x^{lim}, \phi_y^{lim})$ are the regularization parameters (limiters) computed according to the employed method. This ensures monotonicity and supports well-balancing by aligning with source term discretizations, though it may introduce diffusion in smooth regions, as observed in Section 5.

3.3.2 Quadratic (WENO)

To mitigate numerical diffusion observed in lower-order reconstructions during long-range wave propagation (e.g., tsunami simulations in Section 5), we implement a quadratic weighted essentially non-oscillatory (WENO) reconstruction operator, adapted from (Zhu and Qiu, 2018) and applied to unstructured triangular grids like (Sunder et al., 2021), while adhering to the original spatial constraints for stability.

The reconstruction combines a least-squares quadratic polynomial $p_{q,j}$ over the full stencil with four linear polynomials $p_{k,j}$ ($k = 1, \ldots, 4$) over sub-stencils, expressed as:

$$\tilde{q}_j(x,y) = \frac{w_0}{\gamma_0}\left(p_0(x,y) - \sum_{k=1}^{4}\gamma_k p_{k,j}(x,y)\right) + \sum_{k=1}^{4} w_k p_{k,j}(x,y), \tag{26}$$

where p_0 is the quadratic polynomial and $p_{k,j}$ are the linear ones, with nonlinear weights w_0, w_k ($k = 1, \ldots, 4$) computed from smoothness indicators β_0 (quadratic) and β_k (linear) as $w_l = \overline{w}_l/(w_0 + \sum_{k=1}^{4}\overline{w}_k)$, where $\overline{w}_l = \gamma_l(1 + \tau/(\epsilon + \beta_l))$ and τ is a corrector parameter derived from the β_l values (Zhu and Qiu, 2018).

The quadratic interpolant $p_{q,j}$ is obtained via least-squares fitting to the cell-averaged states over Ω_j and its first- and second-order neighbors, ensuring exact reproduction of the mean in Ω_j. Details of this construction, leveraging only geometric information (e.g., barycenters and area moments) without numerical quadrature for efficiency, are provided in (Fuenzalida A. et al., 2025)–representing a key contribution to streamlined WENO implementations on CU schemes over unstructured grids.

For stencil selection, grids with sufficient structure enable a fast, loopless index-based search; However, an efficient geometric search algorithm is to be implemented to relax the requirements on the grid further. The procedure is summarized in Algorithm A2, highlighting `SWEpy`'s efficient, GPU-parallelizable design. This WENO adaptation guarantees high-order accuracy in smooth regions while minimizing errors near abrupt gradients, a crucial enhancement for `SWEpy`'s far-field applications where diffusion must be controlled without Riemann solvers (Kurganov and Petrova, 2005).

3.4 Wet/dry fronts reconstruction

High-order reconstructions, while effective for reducing diffusion in smooth regions, can produce unphysical negative water depths near wet/dry interfaces, where the water surface intersects the bathymetry. To preserve positivity—ensuring non-negative depths for numerical stability and physical realism—we conservatively modify the reconstruction following Bryson et al. (2011). This procedure replaces affected reconstructions with a linear polynomial that maintains the cell-averaged value, thus conserving mass, and handles cases with one or two dry vertices differently to align the surface with the bathymetry at those points.

For a cell where reconstructed depths at vertices yield negatives, the surface is redefined as a plane passing through points that enforce positivity. In the two-dry-vertex case, the plane connects the dry vertices at bathymetric levels and the barycenter at the mean surface elevation. For one dry vertex, it connects the dry vertex at bathymetry, a wet vertex at an adjusted elevation, and the barycenter. Mathematically, the plane equation is fitted to these points, ensuring $h = w - B \geq 0$ across the cell while preserving the mass. We acknowledge that in a hydrostatic framework, high-order accuracy at the immediate wet-dry front offers limited practical benefits compared to its stability risks. Consequently, reverting to a first-order positivity-preserving reconstruction at the front ensures numerical robustness without compromising the overall accuracy of the simulation, as the high-order scheme remains active in the far-field to preserve wave phase and amplitude during propagation. A schematic is provided in Figure 4, illustrating the geometric adjustment.

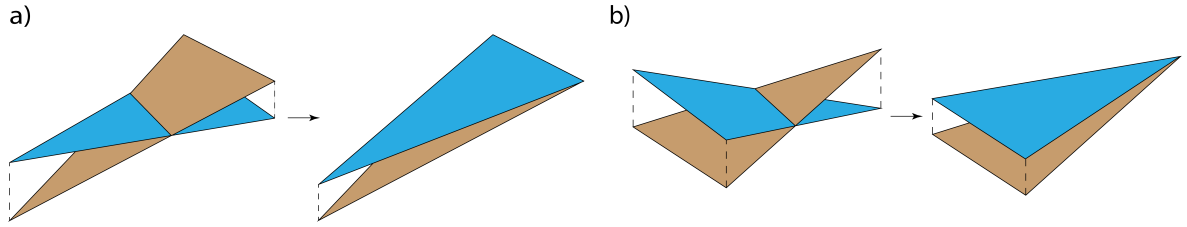

Figure 4. Schematic representation of the wet/dry treatment: (a) two dry points and (b) one dry point

This method guarantees positivity of the water column ($h \geq 0$) across the domain, essential for avoiding instabilities in inundation problems like in the Conical Island test, or the Malpasset Dam failure case (Section 5), with the correction algorithm summarized in Algorithm A3 and executed in parallel via GPU vectorization to identify and adjust interface cells efficiently ~~. However, it does not ensure well-balancing near fronts, where slight imbalances may occur (Liu et al., 2018), suggesting potential extensions with advanced reconstructions for future work.~~ However, it does not ensure well-balancing near fronts, where slight imbalances may occur (Liu et al., 2018). This aspect was investigated using the analytical solution proposed by Synolakis for wave run-up, as described in the Analytical Benchmarks section of the User Manual & Technical Reference. In these numerical experiments, the proposed approximations achieve good accuracy in the wet–dry region. Within this framework, a clear dependence of the results on the temporal discretization and on the Courant number is also observed. These results are currently being prepared for presentation in a dedicated study, primarily focused on the numerical and theoretical

properties of wet–dry reconstruction techniques, with particular attention to the suitability of the underlying model for the phenomenon under consideration.

3.5 Temporal discretization

Following the spatial discretizations detailed in previous subsections, the next challenge is to integrate the resulting system of ordinary differential equations (12) in time, ensuring stability and accuracy across varying flow regimes. We implement both the ~~Forward-Euler (FE)~~ Explicit-Euler (EE) and a four-stage, third-order strong stability-preserving Runge-Kutta scheme (SSP RK4,3, referred to as RK4,3 throughout our work) (Gottlieb et al., 2001) for time integration.

For problems involving Manning friction a semi-implicit treatment is incorporated with the objective to enhance stability without compromising efficiency (Chertock et al., 2015). We define the flux operator as:

$$\mathcal{H}_j(\overline{\boldsymbol{q}}_j, \tilde{\boldsymbol{q}}_j) = -\frac{1}{|\Omega_j|}\sum_{k\in\mathcal{N}_j}(\mathcal{F}_{jk})l_{jk} + \overline{\boldsymbol{SB}_j}, \tag{27}$$

where $\overline{\boldsymbol{q}}_j$ are the conserved variables, $\tilde{\boldsymbol{q}}_j$ denotes the reconstructions. In a Shu-Osher form, the semi-implicit RK update is then:

$$\left(\overline{\boldsymbol{q}}_j\right)^i = \sum_{l=0}^{i-1}\alpha_{i,l}\left(\overline{\boldsymbol{q}}_j\right)^l + \frac{\Delta t}{2}\sum_{l=0}^{i-1}\beta_{i,l}\mathcal{H}_j^l + \Delta t\left(\mathcal{G}(\overline{\boldsymbol{q}}_j)\right)^{i-1}\begin{bmatrix}0 & (\overline{hu}_j^i) & (\overline{hv}_j^i)\end{bmatrix}^T, \quad i = 1,2,3,4 \tag{28}$$

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

$$\begin{aligned}\begin{pmatrix}\alpha_{1,0} & \alpha_{2,1} & \alpha_{3,0} & \alpha_{3,2} & \alpha_{4,3}\end{pmatrix} &= \begin{pmatrix}1 & 1 & 2/3 & 1/3 & 1\end{pmatrix},\\ \begin{pmatrix}\beta_{1,0} & \beta_{2,1} & \beta_{3,2} & \beta_{4,3}\end{pmatrix} &= \begin{pmatrix}1 & 1 & 1/3 & 1\end{pmatrix}.\end{aligned} \tag{29}$$

while the ~~FE~~ EE formulation is identical with nonzero coefficients $\alpha_{1,0} = 1, \beta_{1,0} = 1$.

This semi-implicit approach tackles numerical stability challenges from stiff, strongly coupled, or problematic source terms, particularly in friction-dominated flows near wet/dry fronts, as validated in our dam-break cases (Section 5).

Finally, as a quantifier of the control process between the model evolution, the grid, and the information transport speed, the Courant–Friedrichs–Lewy condition can be forced so the time step Δt is adaptively computed as:

$$\Delta t = CFL \cdot \min_{jk}\frac{r_{jk}}{\max\{a_{jk}^{\text{in}}, a_{jk}^{\text{out}}\}}. \tag{30}$$

where r_{jk} is the perpendicular height from edge jk to the opposite vertex, and CFL is user-defined. Per ~~Bryson et al. (2011)~~ Kurganov and Petrova (2005), CFL $\leq 1/3$ ~~is recommended,~~ is the theoretical boundary of the scheme. Additionally, per Bryson et al. (2011), the theoretical boundary considering the wet/dry reconstruction is $CFL \leq 1/6$, which ensures the positiy preserving condition $h = w - B \geq 0$, though our scheme supports larger values empirically~~, which ensures the positivity preserving condition $h = w - B \geq 0$. For SSP~~ . For RK4,3, Δt is calculated in the first stage and scaled for subsequent ones, balancing stability and efficiency.

4 Architecture & parallel structure

Having established the central-upwind fluxes, reconstructions, and source term discretizations in Section 3, we now describe their ~~GPU-optimized~~ GPU implementation in `SWEpy`. This implementation is designed for modularity, extensibility, and high-performance parallel execution. It also accelerates computations on unstructured grids while allowing users to customize ~~models–such as adding new source terms for phenomena such as rheology or infiltration.~~ models—such as incorporating additional source terms for phenomena like rheology or infiltration/rainfall under an appropriate time discretization. Within this design, the modular structure allows new source terms to be incorporated into the existing framework, provided they follow consistent spatial discretization and time-integration treatments aligned with those used for the current terms.

`SWEpy` follows a modular programming paradigm, partitioning the complex finite volume solver into independent, reusable components or modules (Parnas, 1972). Each module handles different tasks, including grid loading, analysis configuration, preprocessing, time-step integration, spatial reconstruction, numerical flux and source term computations, or 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. While the modular design supports extensibility, incorporating new physical processes requires careful numerical and performance considerations to remain consistent with the existing discretization and GPU-oriented implementation.

Figure 5 presents an overview of the parallel structure and architecture of the `SWEpy` software.

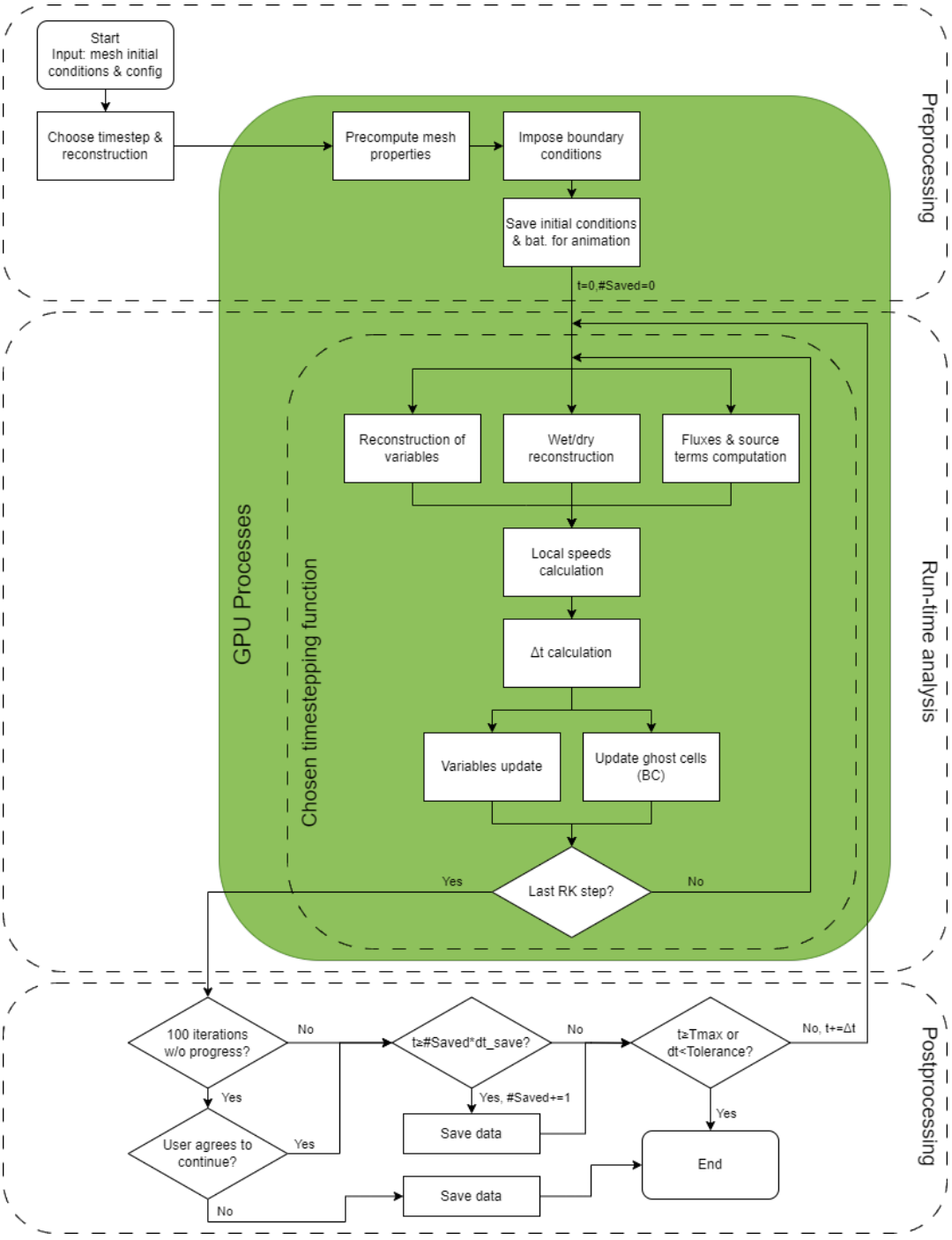

Figure 5. Overview of the `SWEpy` software parallel structure and architecture. The green box contains tasks performed on the GPU. ~~The inner segmented box contains tasks done by the chosen timestepping method. Outer segmented boxes indicate phases.~~ The innermost segmented box contains tasks done 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.

Developed in `Python`, `SWEpy` utilizes CuPy, a CUDA-accelerated counterpart to NumPy, to perform array-based operations on GPUs. This feature enables significant speedups in parallel computations without demanding expertise in low-level languages like C++ or CUDA (Okuta et al., 2017). By representing cell-based data (e.g., states and fluxes) as CuPy arrays indexed by cell ~~number—such~~ number–such as storing the mean water level of cell Ω_3 in `Wj[3]`–`SWEpy` exploits data parallelism inherent to finite volume methods. Operations such as reconstructions and flux evaluations are vectorized across the entire grid, reducing CPU-GPU transfers and leveraging the efficiency of single instruction, multiple data (SIMD) execution (Harris et al., 2020).

This architecture not only accelerates `SWEpy`'s central-upwind scheme on unstructured triangular grids (Section 3) but also scales efficiently for large-scale simulations like the Maule tsunami (Section 5), delivering high-resolution results on consumer-grade hardware, with potentially real-time/faster than real-time execution times. The runtime execution is organized into three phases: Preprocessing, Run-analysis, and Post-processing, as illustrated in the flowchart (Figure 5). Green boxes enclose GPU-accelerated tasks, segmented boxes indicate time-stepping operations and general phases, and arrows crossing over the green boxes highlight CPU-GPU synchronizations, providing a conceptual map of data flow~~and parallelism~~, parallelism, and modularity.

Below, each phase shown in Figure 5 is presented along with the functions of each module, highlighting in italics the function responsible for the specific tasks in the main body.

4.1 Preprocessing

4.1.1 Loading input data *(FileLoader.load_from_files)*

As illustrated in the uppermost part of Figure 5, the preprocessing phase initializes the simulation by preparing input data for efficient GPU execution, minimizing subsequent CPU-GPU transfers, and ensuring scalability for large unstructured grids. `SWEpy` accepts a user-generated triangular grid with vertex coordinates, connectivity arrays, and bathymetry values at vertices, along with initial conditions for the conserved variables, a list of boundary ghost cells with their type, and a configuration file specifying simulation parameters such as numerical tolerances, maximum runtime, gravitational acceleration, CFL number, and optional source term coefficients (e.g., Manning roughness or Coriolis parameter).

~~Key geometric properties required for the numerical scheme–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} (Fuenzalida A. et al., 2025)–are computed in parallel across all cells. Leveraging CuPy's array operations, these calculations exploit vectorized formulas based on vertex coordinates, enabling simultaneous evaluation on the GPU for the entire grid and accelerating setup for high-resolution domains typical in flood or tsunami modeling~~This data is loaded into the *mesh* dictionary, which serves as the main data structure across the runtime.

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

Bathymetry is then reconstructed using an interpolator chosen based on the selected reconstruction method. For example, a linear interpolator is used with the minmod scheme (Bryson et al., 2011)~~, while higher-order schemes like WENO are employed~~

~~for more accurate reconstructions~~. This process involves interpolating vertex values to both edge Gaussian points and the cell interiors, which are required for source term evaluations in equation (19). The interpolation is performed using algebraic expressions, such as calculating the gradient of a plane defined by three vertices. These operations are efficiently executed across the GPU, with precomputed values stored to optimize performance.

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

Key geometric properties required for the numerical scheme–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)* (Fuenzalida A. et al., 2025)–are computed in parallel across all cells. Leveraging CuPy's array operations, these calculations exploit vectorized formulas based on vertex coordinates, enabling simultaneous evaluation on the GPU for the entire grid and accelerating setup for high-resolution domains typical in flood or tsunami modeling.

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

For quadratic WENO reconstruction, a vectorized algorithm identifies first- and second-order neighbors without going through the data each time, flagging boundary cells lacking full stencils for fallback to minmod. Precomputation of least-squares matrices (right-hand side of Eq. (A2))–dependent solely on grid geometry–is performed using CuPy's `linalg` module to invert and multiply stacked matrices in parallel, storing local coefficients as arrays for rapid reuse in the run-analysis phase and reducing overhead in time-critical loops. By offloading these computations to the GPU, 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, as represented in the outer central segmented box of Figure 5, constitutes the core of `SWEpy`'s time evolution, iteratively advancing the solution through spatial reconstructions, source term evaluations, flux computations, and state updates~~—~~; all optimized for GPU parallelism to exploit the data-local nature of finite volume operations on unstructured grids. This phase leverages CuPy's array-based processing to perform calculations simultaneously across the entire domain, minimizing serial bottlenecks and enabling efficient simulation of large-scale flows, such as those validated in Section 5.

4.2.1 ~~Reconstruction~~

Before entering the main loop, the program chooses the "timestep" function to be used according to user selection *(ShallowWater.choose_timestep)*. This timestep function is central as it contains the sequence of reconstructions, corrections, and calculations to be done in order to update the flux variables on each iteration according to the central upwind method and the temporal discetization used. In other words, it defines what an iteration is. This means that designing timestep functions, using the elements listed in the following, allows for straightforward switching between reconstructions, source terms, time discretizations, and even numerical fluxes definitions.

4.2.1 Reconstruction *(PieceWiseReconstruction.minmod/weno2)*

For minmod reconstruction, SWEpy computes gradients for each cell using neighboring average states, interpolating conserved variables $\boldsymbol{q}_j(M_{jk})$ at edge midpoints for flux calculations in equation (16) and water surface values at vertices for wet/dry handling. Users can optionally include a diffusion coefficient, which is commonly used in minmod reconstruction, $\vartheta \in [1,2]$ to control dissipation, where higher values reduce diffusion but may introduce oscillations. CuPy's Single Instruction Multiple Data (SIMD) capabilities enable parallel gradient computation and interpolation across all cells, replacing sequential iterations in Algorithm A1 with vectorized operations on the GPU for accelerated performance. This means that "for-loops" in alg. A1 are performed in parallel over all cells.

In the case of the WENO reconstruction, the model solves the LSQ linear systems A2 using the precomputed matrices to construct the reconstruction polynomials. These are used to reconstruct the values at the two gaussian midpoints of the sides of each cell (for use in the numerical flux calculation 16), the values in the three gaussian points inside the cell (for use in the source terms calculation 19), and the values of the water surface at the vertices (for use in the wet/dry reconstruction). Since we saved the stencils needed for each cell in the preprocessing phase, we can perform these reconstructions in parallel accelerated by the GPU processing. Once again, this means that iteration in alg. A2 is parallelized over all cells.

Boundary cells lacking full second-order neighbors, identified during preprocessing, default to minmod reconstruction. Wet/dry fronts are corrected by replacing invalid reconstructions (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 alg. A3 ~~is~~ are replaced by SIMD commands for all cells, efficiently performed over the GPU. SWEpy's modularity supports user-defined reconstruction operators, requiring only interpolated values at specified points, facilitating extensions like hybrid schemes while preserving GPU efficiency.

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

Bathymetry source terms are computed for all cells using equation (19), leveraging preprocessed bathymetry reconstructions and the active spatial operator. Similarly, the Coriolis and friction terms are calculated using each cell's state and saved for use in the variables update stage.

These evaluations are vectorized across the grid on the GPU, ensuring parallel computation even for optional terms. ~~The scheme's flexibility accommodates multiple source terms, allowing users to define custom modules (e.g., for rainfall, rheology, and/or turbulence) integrated seamlessly into the parallel workflow.~~ The scheme's flexibility allows users to define additional source-term modules and integrate them into the existing workflow, provided that consistent spatial discretization, reconstruction order, and time-integration treatments are ensured.

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

Using reconstructed states, velocities are desingularized and projected onto edge normals to compute local propagation speeds (18) for all edges simultaneously on the GPU. If adaptive time stepping is enabled, Δt is determined enforcing the CFL stability

limit given the local propagation speeds, maximizing step size with the objective of enhancing efficiency and minimizing diffusion (cf. (30)).

~~Forward-Euler~~ Explicit-Euler (EE) integration applies (16) via array operations on the GPU, with CPU synchronization only for timestep advancement (one cycle in Figure 5). The ~~SSP~~ RK4,3 methods (Gottlieb et al., 2010), and the classical 4-step RK4 method both decompose into four scaled Forward-Euler steps, executed sequentially on the GPU to avoid iterative solvers and reduce overhead, optionally including friction corrections per stage.

~~Modularity enables user-defined integrators, such as implicit schemes for stiff problems, to be integrated into the GPU workflow for future extensions.~~ The modularity of the *timestep* function enables user-defined integration schemes to be incorporated into the GPU workflow for future extensions, such as predictor–corrector methods or the implementation of a modified Newton–Raphson subroutine for implicit formulations, although implicit formulations are currently less well suited to efficient GPU execution.

4.2.4 Variable update and correction (*timestep* function)

Fluxes are assembled using local speeds and reconstructions, updating cell states via the CU scheme in equation (16) in a single GPU-parallel operation, followed by ghost cell boundary imposition. For Manning friction, an intermediate semi-implicit correction adjusts the discharge vector-wise across the grid.

For multi-stage Runge-Kutta updates, intermediate states and fluxes are computed on the GPU and stored, with optional friction corrections applied per stage, minimizing synchronization and preserving third-order accuracy.

4.2.5 Boundary conditions (*BoundaryConditions.impose*)

The program imposes the user-defined boundary conditions on the ghost cells' states. Since each ghost cell has its definition, mixed boundary conditions at border cells can be defined and imposed all at once. The modularity of the program even allows for the definition of more complex functions that take care of the boundary conditions, like transport models for coupling other solvers' results, or higher-order extrapolations.

~~Boundary conditions currently implemented via ghost cells are: zero-order fully permeable (soft) border, by replicating the border cell's state at the neighboring ghost cell; and impermeable (wall) boundary, by replicating the border cell's water height, but inverting the flow direction, resulting in a zero-flow interface.~~ Boundary conditions currently implemented via ghost cells are: zero-order extrapolation permeable (soft) border, by replicating the border cell's state at the neighboring ghost cell; and impermeable (wall) boundary, by replicating the border cell's water height, but inverting the flow direction, effectively reflecting it, resulting in a zero-flow interface.

We highlight that periodic boundary conditions may also be naturally modeled by setting boundary cells as neighbors of other boundary cells. For example, if we want the top and bottom borders of a channel to be periodic, the grid should consider the bottom cells as the upwards neighbors of the top border cells and vice versa. These kinds of boundary conditions are used in some of our experiments.

4.3 Postprocessing *(FileSaver module)*

The post-processing phase, as shown in the lower portion of Figure 5, finalizes the simulation by managing data output and termination criteria, leveraging `SWEpy`'s modularity to enable customizable workflows that support both research analysis and operational monitoring. Users can integrate ~~routines—predefined or custom—to~~ routines–predefined or custom–to export results at any runtime stage, facilitating immediate inspection and post-simulation processing.

For data saving, `SWEpy` offers built-in options to store initial conditions and bathymetry *(FileSaver.save_bathymetry)* in .vtk format before run-analysis begins, with subsequent snapshots saved at user-defined intervals Δt_{save} until completion *(FileSaver.save_animation)*. This format is optimized for visualization tools like ParaView (Ayachit, 2015), enabling rendering of snapshots or animations to track flow evolution over unstructured grids. Time series of states for selected cells (e.g., virtual gauges in tsunami validations, Section 5) are also supported at the same Δt_{save} *(FileSaver.save_TS)*. Integrated into the main runtime, this successive saving enables live visualization of emerging solutions, crucial for detecting anomalies in long-running simulations without halting progress. CuPy's GPU-accelerated NumPy equivalents (e.g., `save`, `savez`, `savetxt`) ensure efficient disk writes for large arrays, preserving performance even during high-frequency outputs.

Simulation termination occurs when the elapsed time reaches the user-specified target. Still, modularity allows for other tailored criteria, such as the available divergence detection (e.g., if adaptive Δt drops below a threshold) or stagnation monitoring (e.g., negligible progress after a set number of iterations) , with further options achievable by manipulation of the *run* functions, or definition of new ones. In our experiments, these controls prevented unproductive runs, complementing live visualization for on-the-fly adjustments. This flexibility extends `SWEpy`'s utility, allowing integration with external tools for automated error handling or real-time coupling in hybrid modeling setups.

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 efficacy of the central-upwind scheme with WENO reconstructions and GPU acceleration detailed in Sections 3 and 4. We begin with canonical tests assessing spatial and temporal ~~order~~accuracy, well-balancing, positivity preservation, and diffusion reduction, followed by synthetic scenarios highlighting versatility across reconstruction operators. Finally, large-scale simulations of the 1959 Malpasset dam failure and 2010 Maule tsunami evaluate real-world applicability, comparing results to historical data and established solvers like TELEMAC (Moulinec et al., 2011). These experiments underscore `SWEpy`'s robustness for inundation with complex topography and long-range wave propagation, achieving high-resolution outcomes on consumer hardware with computation times reduced by up to $21\times$ via CuPy parallelism.

5.1 Benchmark tests

5.1.1 Spatial convergence order study

To validate `SWEpy`'s spatial accuracy and verify the correct implementation of the bathymetry source term discretization (19), we replicate the convergence test of Bryson et al. (2011), focusing on scenarios where well-balancing is essential to preserve steady states over non-flat topography without introducing spurious oscillations. The test quantifies the order of convergence for different spatial reconstruction operators (Section 3.3) in the presence of a smooth Gaussian bump.

The computational domain is a 2×1 m rectangle discretized into a regular mesh of equilateral triangles, with triangles along the top and bottom boundaries halved for consistent boundary treatment. The bottom topography is defined as

$$B(x, y) = 0.5 \exp\big(-25(x-1)^2 - 50(y-0.5)^2\big). \tag{31}$$

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

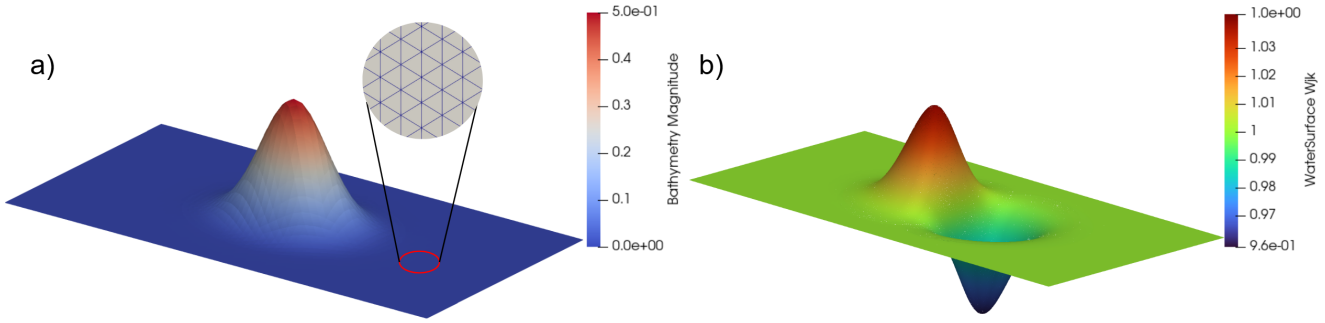

Figure 6. Bottom topography on a $n_x = 32$ point grid (left) and reference solution (right) for the Gauss bump. The zoomed-in circular window highlights the grid structure pattern.

The reference solution is computed at $t = 0.07$ s on a fine grid with $n_x = 512$ horizontal divisions, corresponding to approximately 1.18×10^6 cells. 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

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

and convergence orders are estimated via successive grid refinements. Table 2 reports errors and orders for grids $p = 0$ to 3 ($n_x = 32 \cdot 2^p$) across reconstruction–timestepper combinations, while figure 7 shows the log–log relationship between error and effective grid spacing Δx, with fitted power laws confirming the observed slopes.

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

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.

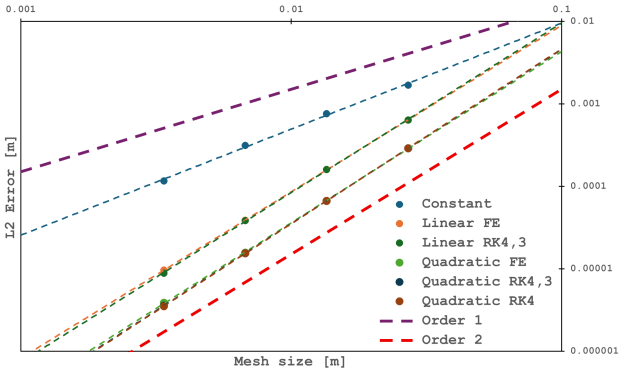

Figure 7. Log-log plot of L^2 error versus grid size Δx, with fitted power-law curves indicating convergence orders.

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 discretisation may be needed to fully realise higher-degree polynomial benefits. This degradation in order is an expected behaviour, ocurring in experiments such as those performed by Bryson et al. (2011), and more recently Nguyen (2023). However, a detailed convergence analysis of the extended scheme is 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 ~~forward Euler~~ Explicit-Euler time-stepping and linear reconstruction required approximately 5 minutes wall-clock time on consumer-grade GPU hardware, including high-frequency (0.01 s) output for animation purposes. This demonstrates that `SWEpy` can deliver high-resolution, well-balanced solutions for smooth-bottom flows at modest computational cost.

5.1.2 Well-balancing test

A simulation of a static flat water surface is performed over white-noise-generated bathymetry to verify the well-balanced property for all reconstruction operators. The domain is a 1×1 [m] box, with a random bathymetry ranging from $-1.1d$ to $-0.9d$ and average depth of $d = 2$, discretized using a right triangles grid with $\Delta x = \Delta y = 0.01$. The zeroing tolerance is set to 10^{-17} to test if the software is machine-precision accurate. Runs were performed using the explicit Euler method for time evolution, and an adaptive timestep with $CFL = 0.25$.

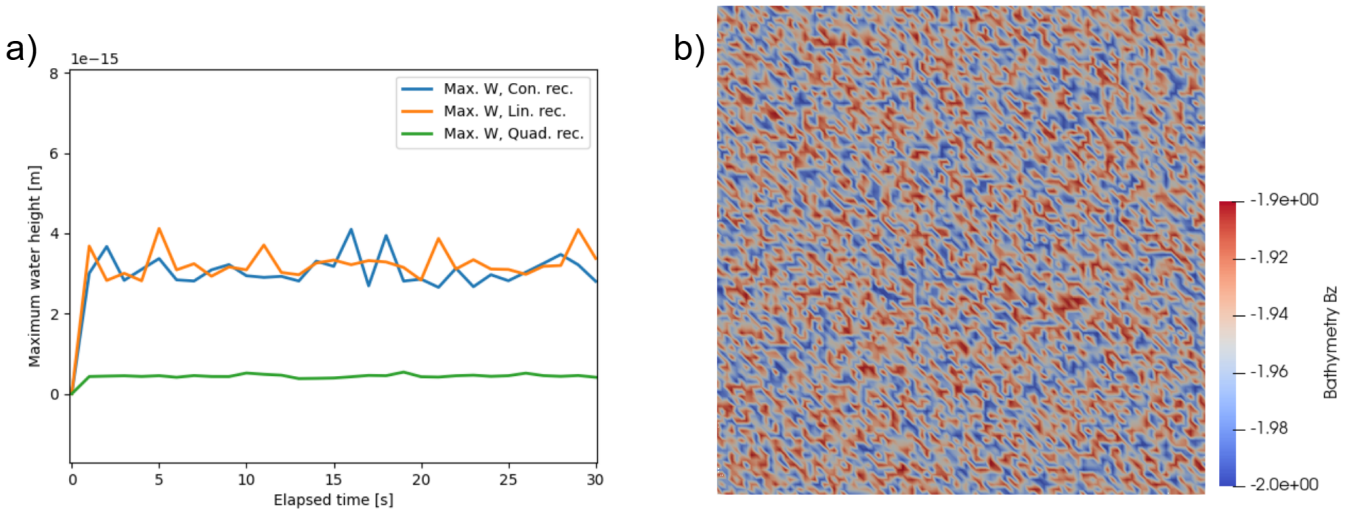

Figure 8. (a) Maximum water height over time. (b) Randomly generated bathymetry.

The initial water surface is set to 0 [m] and the velocity profile is null in both directions. Figure 8 shows the random bathymetry and the maximum water height over time for the three reconstructions, confirming no spurious oscillations arise. These results, while potentially surprising, are in fact anticipated since our bathymetry source term is exactly well balanced. This means that even in the presence of boundary conditions, unphysical oscillations or flows will not appear. As a note, this experiment also confirms the well-balancing property with the Runge Kutta scheme, since it's implemented as 4 successive EE steps.

5.1.3 Conical island wetting–drying benchmark

To evaluate `SWEpy`'s wet/dry handling and reconstruction detail during wave-obstacle interactions—essential for coastal inundation modeling—we simulate the conical island benchmark, a laboratory experiment by Briggs et al. (1995) recommended for SWE validation in Synolakis et al. (2008). This test assesses (i) positivity preservation on sloped topography, where runup/rundown induces dynamic wetting without unphysical negatives, and (ii) compares operator fidelity in capturing complex wavefronts. The domain is a 41×30 m tank with permeable (soft) boundaries to absorb reflections, minimizing boundary artifacts. Discretization uses an equilateral triangular grid with edge length 0.5 m (12,000 cells) for isotropy. Bathymetry features a flat bottom with a truncated cone (toe diameter 7.2 m, crest 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:

$$w(x,y,0) = \frac{H}{d}\,\mathrm{sech}^2\big(\gamma\,(x - X_1)\big), \tag{32}$$

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

$$u(x,y,0) = \frac{g}{d} w(x,y,0)\left(1 - 0.25\frac{w(x,y,0)}{d}\right), \quad v(x,y,0) = 0, \tag{33}$$

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 a ~~forward Euler~~ Explicit-Euler time integration scheme with a Courant-Friedrichs-Lewy (CFL) number of 0.25. ~~This choice isolates~~ The reason for this temporal integration setup is twofold: (1) it showcases the solver's capability of a higher effective CFL number, and (2) it's low enough so oscillations are controlled, highlighting the spatial-reconstruction effects, ~~allowing~~ and allowing for a focused assessment of how each operator handles wet/dry transitions and wavefront modelling as the soliton interacts with the cone, as visually depicted in the initial setup of Figure 9.

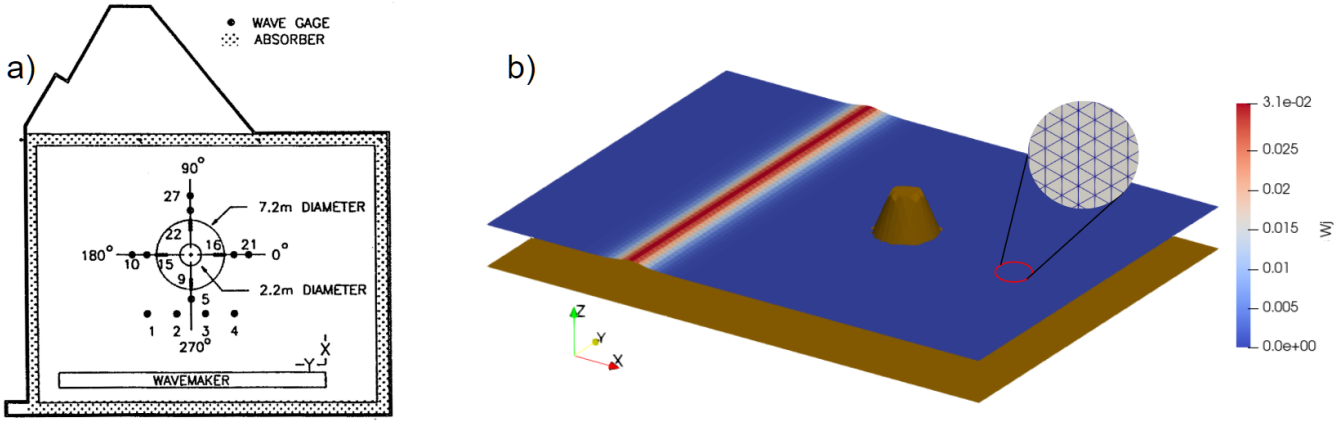

Figure 9. Configuration of original experiment (digitized from Briggs et al. (1995)) for the conical island benchmark, illustrating the solitary wave profile approaching the truncated cone.

Figure 10 shows the free-surface elevation at each element (Wj) for solutions using each reconstruction operator at $t = 7$ s, when the leading wave is passing the location of 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 exhibits 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 (respectively from left to right), aligning with the experimental setup for direct validation of runup dynamics.

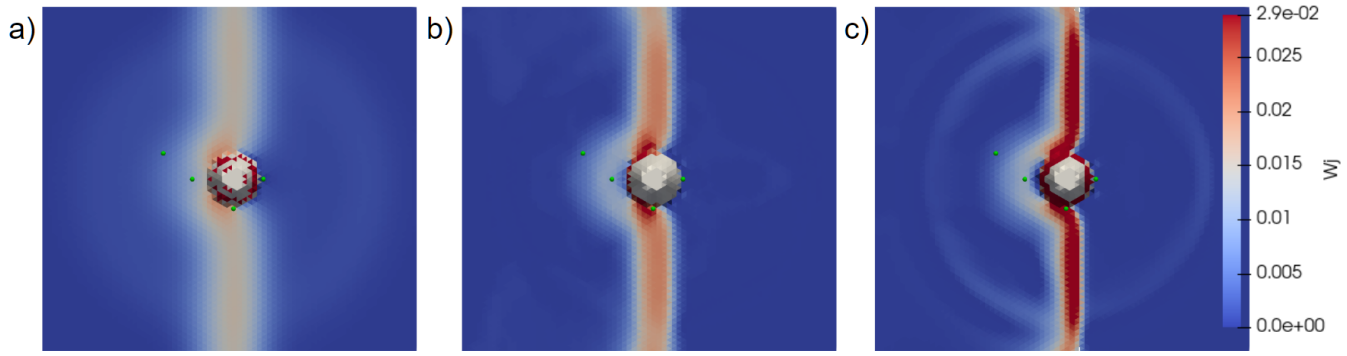

Figure 10. ~~Comparison of wavefront measurement at gauge 16.~~ Comparison of wave structure while passing through gauge 16. Free-surface elevation at $t = 7$ s for (a) constant, (b) minmod, and (c) WENO reconstruction operators. Green markers indicate the positions of gauges 1, 6, 16, and 22 (respectively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

We note that, due to the nature of the wet/dry reconstruction procedure, minimal artifacting occurs around the wet/dry interface producing "wet" triangles with tiny water column heights (approx. 10^{-5} to 10^{-4} m). This can be further reduced by tuning the dry tolerance parameter in the simulation configuration file, and by grid refinement, as shown in the following figure:

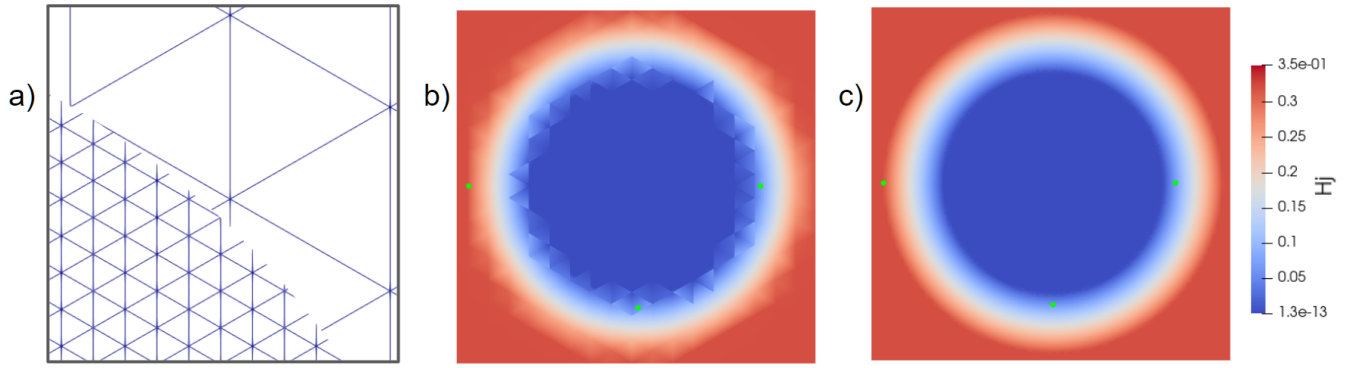

Figure 11. (a) Grid size comparison closeup. Water column heights (Hj) around protrusion of conical island diminish from the (b) coarse grid to the (c) fine grid.

It can be seen that the artifacting is much less noticeable in the finer grid. In this case, the water heights near the interface were of the order of 10^{-10}–10^{-9} m.

Figure 12 presents time-series results 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 incident wave arrival with simulations~~, corresponding~~. This value corresponds to the estimated paddle deceleration inferred from signal traces in Briggs et al. (1995).

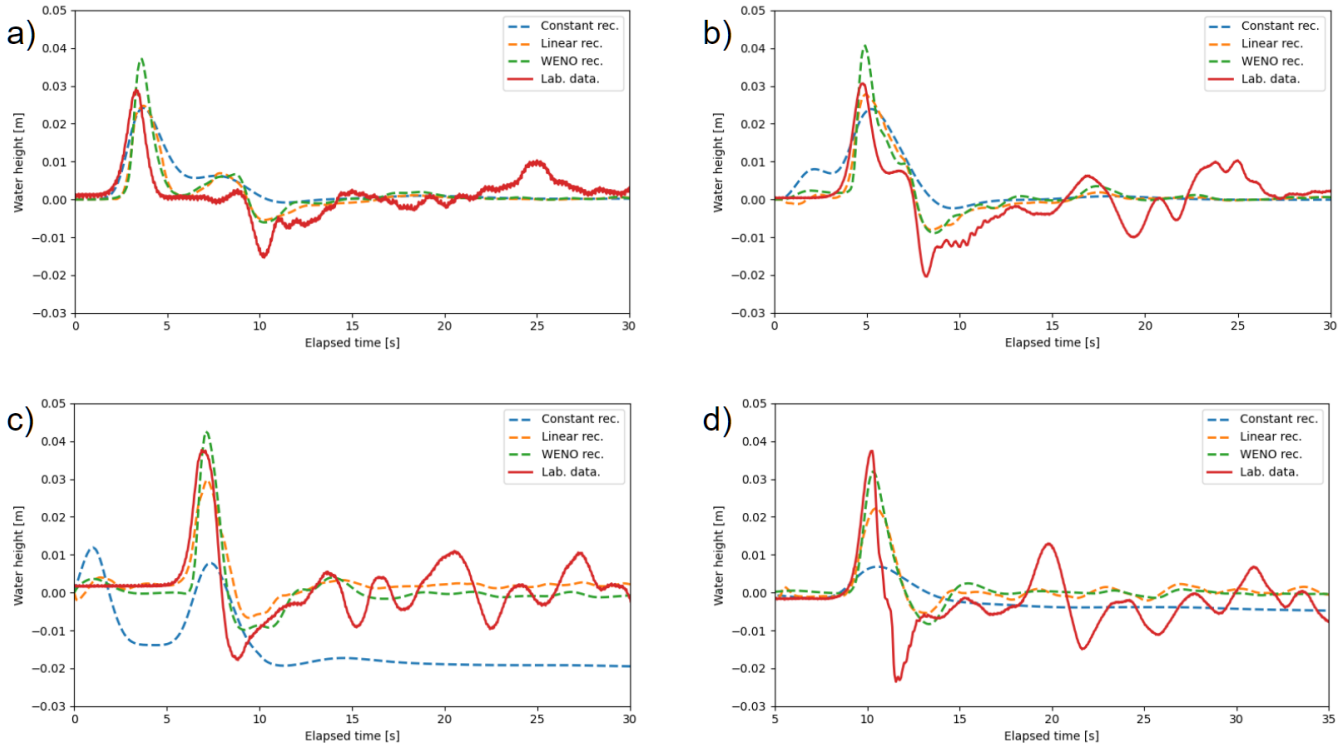

Figure 12. Time series of water heights at gauges 1 (a), 6 (b), 16 (c), and 22 (d). Solid line is laboratory data, and dashed lines are `SWEpy`'s solutions.

Across gauges, WENO reproduces main crest height and arrival with minimal phase error, while constant and minmod underestimate secondary oscillations and show 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% ($\overline{|e|}_2^{\text{constant}} = 0.0533$).

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

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

Since L^2 is phase-sensitive and the shift is estimated, we repeated analysis with uniform shifts from -0.10 to $+0.30$ s on numerical series (Table 4), confirming WENO's robustness (lowest scores across shifts), validating its capacity for dynamic wet/dry reconstruction in coastal flows.

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

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

Figure 13 illustrates spatial diffusion effects by comparing free-surface elevation at $t = 13$ s across the three reconstruction operators, 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 retaining 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 $\mathcal{E} = \rho g \int_{\Omega_{\text{strip}}} \eta^2, d\Omega$ over a vertical strip from $x_l = 9.5$ m to $x_r = 14.5$ m downstream, where $\eta = w$ given the zero still-water level 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 optimal balance between low numerical diffusion and faithful waveform reproduction, whereas constant reconstruction underperforms in both aspects.

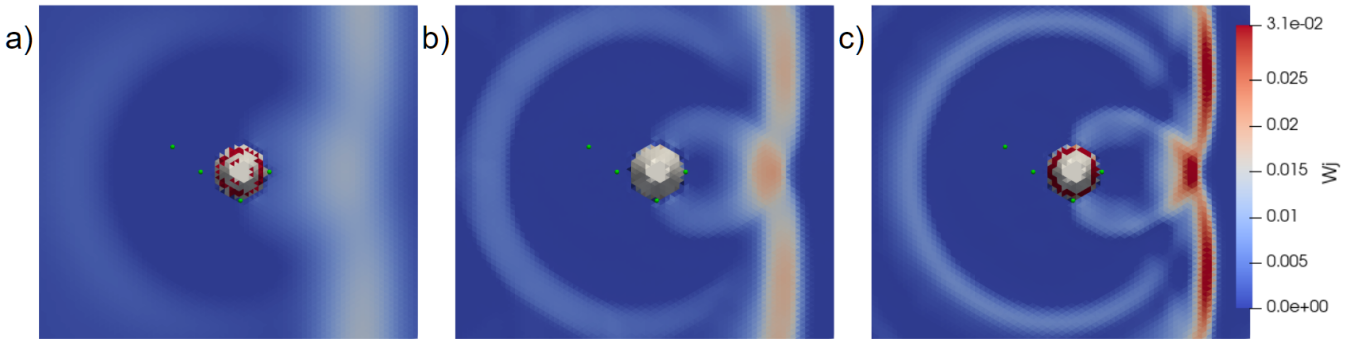

Figure 13. Diffusion study. Solution at time $t = 13$ s for constant (a), linear (b), and quadratic (c) renconstructors. Green marks are the locations of gauges 1, 6, 16, and 22 (from left to right).

From the previous comments, it can be appreciated that point measurements reflect only one aspect of the numerical result, which can be helpful for specific events such as wave front arrival time, zones of maximum amplitudes, or inundations, among

others. In addition, Figure 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 drawing on the conclusion that the WENO-based approach captures relevant information for global analysis, Figure 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, 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 through the interaction, even in the presence of zeroth-order boundary conditions. Thus, the wavefront sequence in Figure 14 illustrates that `SWEpy`'s implementations incorporate the appropriate numerical foundations, enabling detailed descriptions of more complex real-world phenomena.

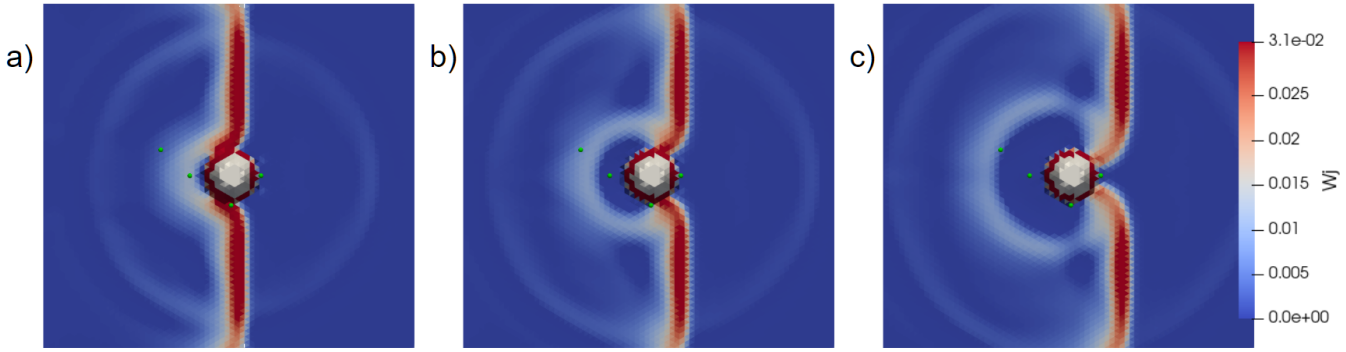

Figure 14. Wave-obstacle interaction study. Solution at times $t = 7$ s (a), $t = 8$ s (b), and $t = 9$ s (c), using the WENO reconstruction operator. Green marks are the locations of gauges 1, 6, 16, and 22 (from left to right).

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 height amplitude, phase, and post-interaction structure, while maintaining stability at wetting–drying fronts, 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 aligning 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 benchmarks scaling from coarse to fine resolutions using the previous Conical Island and Gaussian Bump test cases. 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 (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 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 shifts into a throughput-bound state, achieving a speedup of $26.2\times$. This confirms that SWEpy's architecture is optimized for high-resolution scenarios where the arithmetic intensity of WENO reconstruction saturates the GPU's CUDA cores.

Test	Resolution	#Elements	GPU Time/step (s)	CPU Time/step (s)	Speedup
C. Island	Coarse	2,958	1.03	0.076	$0.07\times$
	Medium	11,682	1.03	0.411	$0.40\times$
	Fine	45,924	1.04	2.579	$2.47\times$
G. Bump	Coarse	18,992	0.96	0.73	$0.76\times$
	Medium	74,848	1.22	8.39	$6.90\times$
	Fine	1,184,128	8.37	219.29	$26.20\times$

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.

System traces reveal the hardware utilization patterns across different scales. As shown in Table 6, particularly for the Gaussian Bump benchmark, GPU throughput (ratio of active kernel time to total GPU time) rises from 22.9% at coarse resolution to 91.1% at fine resolution, confirming effective saturation of the compute resources. Also, by looking at the Conical Island performance, we can see that the need for the wet/dry correction procedure plus the extended simulation duration significantly increase the number of GPU operations performed, without augmenting the GPU throughput. This indicates that the density of kernel launches is throttling the efficient usage of computation resources. This suggests the opportunity for CuPy optimizations like kernel fusing, or direct element-wise kernels for processes with numerous arithmetic operations.

Test	Resolution	#Elements	GPU Time (s)	GPU Ops	GPU Throughput	VRAM Throughput
C. Island	Coarse	2,958	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%

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

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

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. This benchmark event is characterized by rapid flooding over highly irregular terrain (Moulinec et al., 2011). The case serves to validate the model's positivity-preserving reconstruction schemes, treatment of bathymetric source terms, and semi-implicit friction formulation described in Section 3.

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 (measured from a vertex to the opposing side) ranging from $\Delta r_{\min} = 4.01$ m to $\Delta r_{\max} = 401.95$ m, and an average value of $\Delta r_{\text{avg}} = 40.28$ 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 (Figure 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 (Figure 15b). Boundary conditions are specified as impermeable walls; conditions also present in the TELEMAC example. The Manning roughness coefficient is set to $n = 0.03$ to match ~~the TELEMAC~~TELEMAC's configuration. 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.

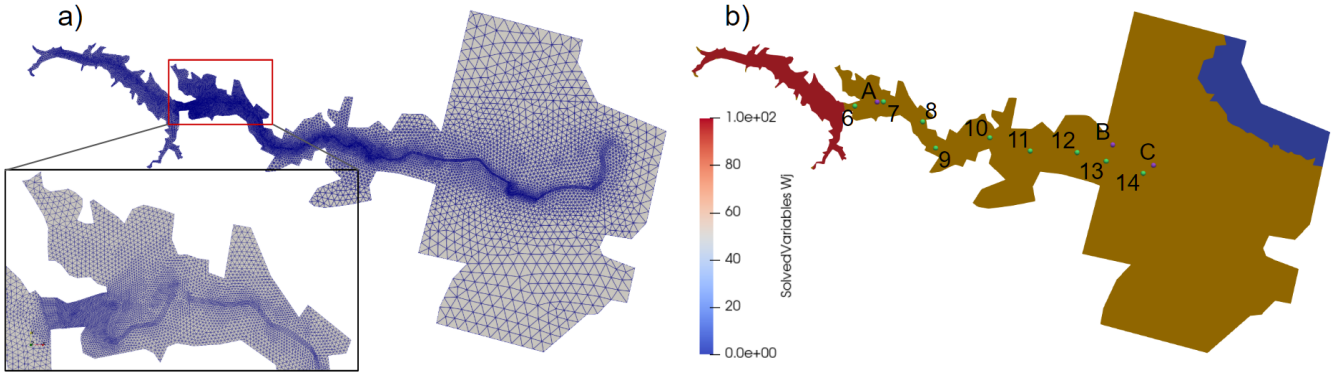

Figure 15. Grid used in the simulation (a) with zoomed view of the upstream refined part, and initial water height (b). Points are the locations of measured data.

Twelve virtual gauge locations are defined along the river valley to monitor the flood wave progression (Figure 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, which is regarded as the most accurate scheme for this benchmark.

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

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

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 discrepan-

cies at certain locations, its relative errors are, in most cases, nearly an order of magnitude smaller than those produced by TELEMAC's finite volume 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 likelihood 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 shallow-water equations, further contributing to deviations between simulated and observed results.

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 m and 8 m, values that align more closely with the observed data. This poor resolution stems primarily from the coarse bathymetry dataset lifted from the original maps, therefore finer remeshing wouldn't necessarily solve this discrepancy.

Given the complex bathymetry and the tightly spaced unstructured grid, some numerical artifacts ~~arise in our~~ arised 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 normal points outward 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 ($\sim$ 0.01 m) is negligible when compared to the wave arrival heights ($\sim$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.

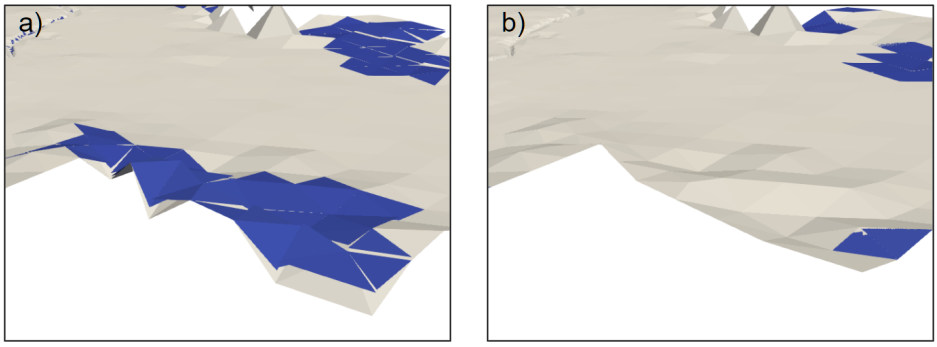

Figure 16. 3D View of solution before (a) and after (b) grid correction. Artificial water influx is reduced thanks to the bathymetry fix.

~~However~~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 neglegible sizes; thus confirming these artifacts do not pollute the results showed.

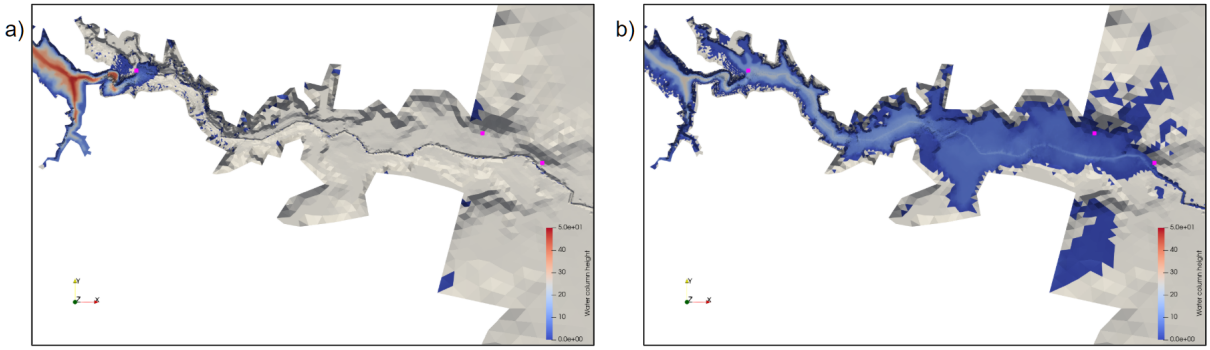

Figure 17. Top-down view of inundation wave arriving at transformer A (a) and C (b). Transformer locations are marked with magenta boxes. Some water influx is present due to errors of the border-wet/dry interaction, but remains outside the measuring zones.

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, like rheology, sediment transport, and the influence of steep bathymetric gradients or strong flow curvature, could further enhance predictive skill. The interaction between ghost-cell boundaries and the wet/dry treatment, identified here as a source of spurious inflow, will be studied in more detail in (Fuenzalida A. et al., 2025).

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 assessment of the combined impact of WENO spatial reconstruction and ~~SSP~~ RK4,3 time integration, in comparison with minmod and ~~FE~~ EE methods. The computational mesh is an equilateral triangular grid generated from GEBCO bathymetry (GEBCO Bathymetric 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 $f = 10^{-4}$ 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 ~~FE~~ Explicit-Euler (EE) 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 32412 (southwest of Lima, Peru) and 21413 (southeast of Kyoto, Japan), as indicated in figure 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.

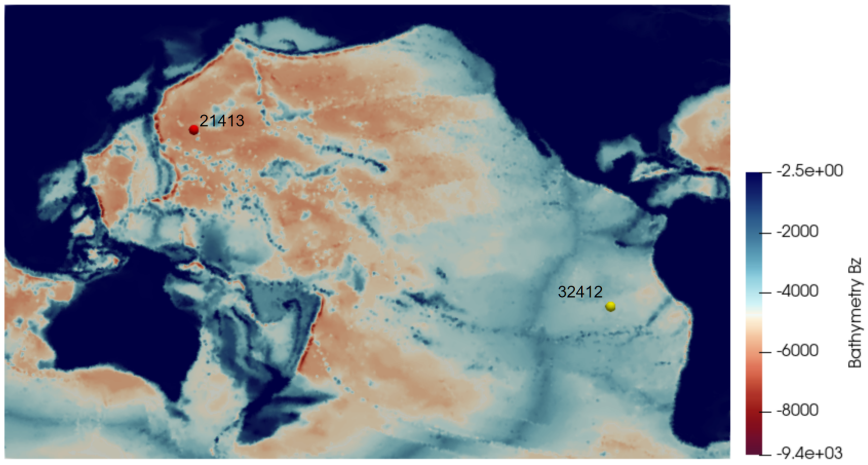

Figure 18. Bathymetry grid employed for the Maule tsunami simulation.

Figure 19 compares the simulated tsunami waveforms at DART buoys 32412 (panel a) and 21413 (panel b) with quality-controlled observations and reference simulations from the HySEA model. At buoy 32412, the WENO reconstruction with ~~FE~~ EE 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 21413, 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 minmood operator becoming too oscillatory, although the differences are less pronounced. Across both sites, HySEA results at 1.5 M cells show closer agreement with the DART data than `SWEpy`, consistent with the benefits of higher effective resolution. While WENO incurs a modest computational cost increase over minmod ~~(+14 minutes)~~(as explored later), it remains faster than real time for the domain and resolution tested, ~~and offers~~ offering a consistent improvement in waveform fidelity.

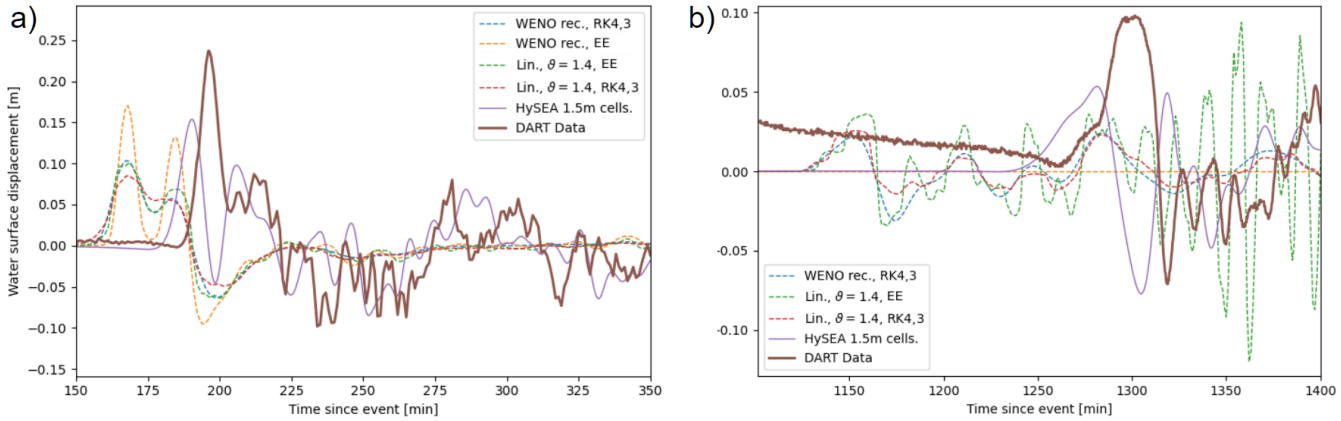

Figure 19. Tsunami profile comparison at DART buoy 32412 (a) and 21413 (b). Results from the HySEA model are included for reference. The ~~FE~~EE+WENO result for buoy 21413 is omitted because of high-frequency oscillations that obscured the primary tsunami signal.

The ~~FE~~EE+WENO result is not shown for buoy 21413 because the combination of ~~forward~~ Forward Euler time-stepping and coarse resolution over the long propagation path generated high-frequency oscillations that obscured the primary tsunami signal (figure 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.

~~Performance summary of `SWEpy` configurations and HySEA for the Maule 2010 tsunami benchmark. Maximum surface displacement $H_{\max}$ mand arrival time T_{arr} minare extracted at the first wave crest for DART buoys 32412 (Lima) and 21413 (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	-
WENO+RK4,3	0.1034	-56.4	167.263	-14.7	0.022	-77.6	1151.39	-11.2
WENO+~~FE~~ EE	0.1703	-28.1	167.098	-14.7	-	-	-	-
minmod+RK4,3	0.0851	-64.1	167.197	-14.7	0.02551	-74.0	1152	-11.1
minmod+~~FE~~ EE	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

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 32412 (Lima) and 21413 (Kyoto). Relative errors (%) are computed with respect to the DART observations.

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~~ bias–particularly at Lima–and reduced phase error, consistent with the visual comparisons in ~~figure~~Figure 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 shallow-water equations 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, cell area, and such 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 by a rigorous treatment of the spherical case.

While initially producing the most accurate waveforms at the near-field gauge, the ~~FE time integrator —particularly~~ EE 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 the modeled free-surface elevation at the moments when the wave passes the Lima and Kyoto DART buoys, for both ~~FE~~EE+WENO and RK4,3+WENO configurations, highlighting the marked difference in solution smoothness.

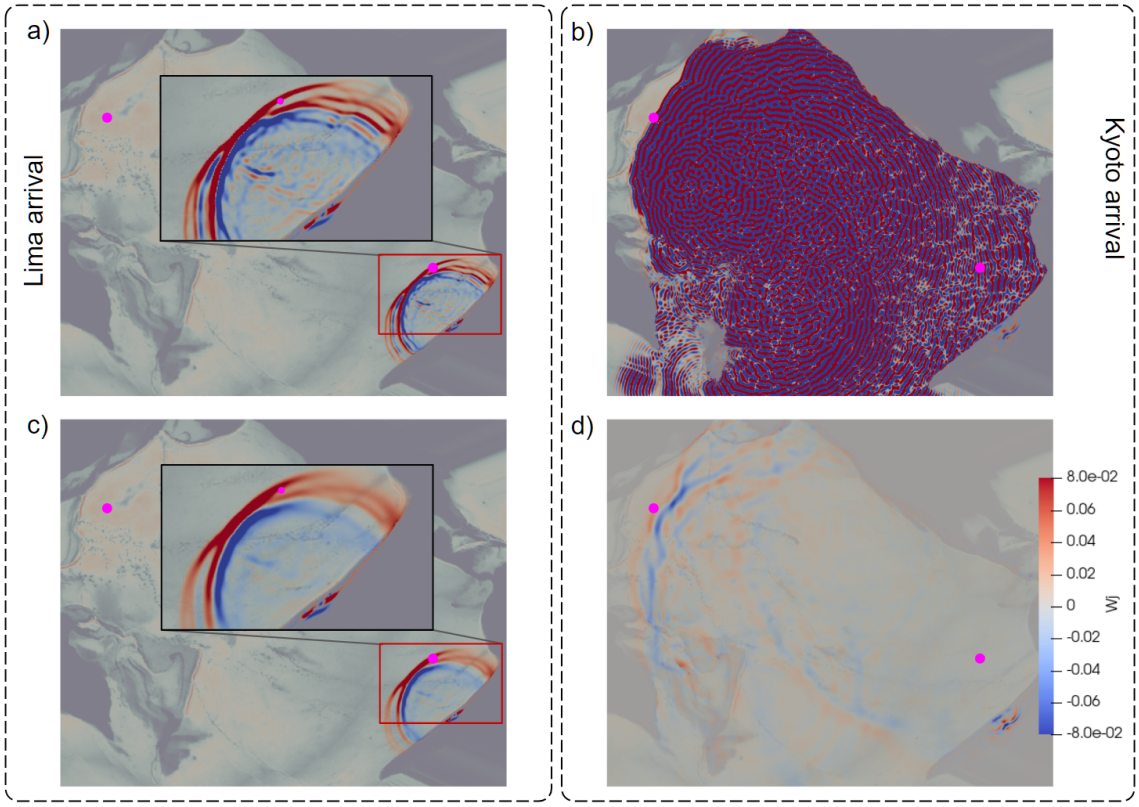

Figure 20. Snapshots of simulated free-surface elevation for the Maule 2010 tsunami at the times when the leading crest passes the locations of DART buoys 32412 (Lima) and 21413 (Kyoto). Insets show zoomed view of wave at Lima buoy. Panels (a) and (b) correspond to ~~forward~~ Explicit Euler (~~FE~~EE) integration, and panels (c) and (d) to ~~SSP~~ RK4,3 integration. Buoy locations are marked with magenta circles. These views provide basin-scale context for the time-series comparisons in figure 19.

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 ~~$CFL = 0.15$~~$CFL = 0.2$, 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.

~~To evaluate~~ Spurious oscillations in Figure 20 panel (b) 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 long channel with periodic boundary conditions was considered, see Section 7.2 *Instability study: impact of temporal discretization* in the User Manual & Technical Reference. The results show that simulations employing a more diffusive reconstruction do not generate spurious oscillations, whereas the EE+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.

With the objective of exploring grid convergence, additional runs were performed with closer HySEA-SWEpy resolution aggreement, using results from a HySEA run with a ∼250k element resolution as reference, and a SWEpy grid of ∼280k (effective) elements. Also, to evaluate reduction of oscillatory behaviour in the WENO+EE case, a run with a CFL number of 0.15 was also performed. Figure 21 shows the output at the DART buoys evaluated.

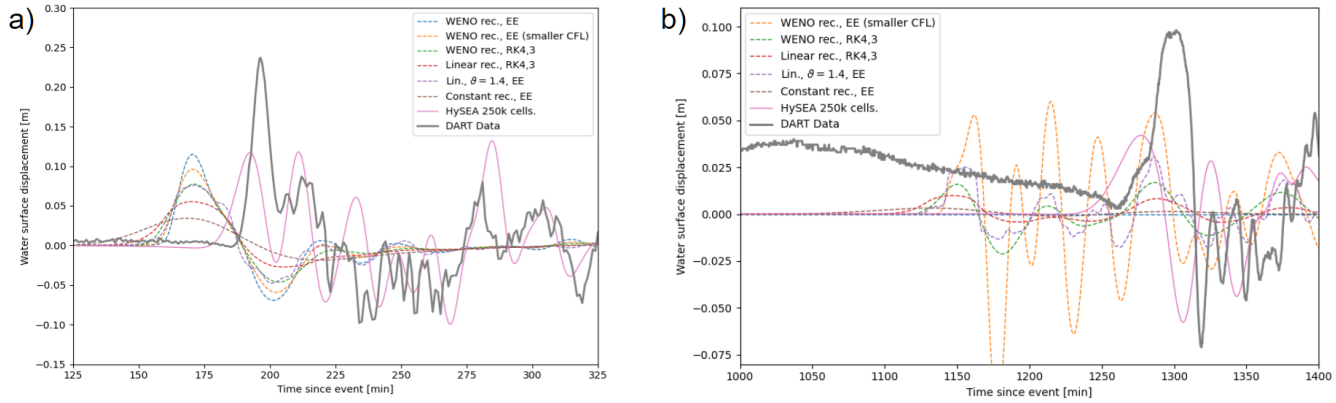

Figure 21. Tsunami profile comparison at DART buoy 32412 (a) and 21413 (b) for the coarser grids. Once again, results for the WENO+EE at the Kyoto buoy have been ommited.

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 via CFL number of the spurious oscillations is achievable with the tradeof of some 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 simulation, 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 degradation in solution quality. Despite this, execution times for all configurations were faster than real time (table 9).

~~Comparison of methods minmod+FE (fastest) and WENO+RK4,3 (slowest)~~

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

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

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, these performance comparison is intended to illustrate the benefits of GPU acceleration within the SWEpy framework using a standard single-core CPU baseline, rather than to provide 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 forward Euler integration it can develop basin-scale oscillations that obscure the signal at distant stations (figure 20). These oscillations persist even under reduced CFL numbers, with the trade-off of increased diffusion and degraded amplitudes at nearer gauges. The ~~SSP~~ 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 ~~serial implementations~~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 Python package implementing a (CUDA) GPU-accelerated central-upwind finite-volume solver for the shallow-water equations (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. Its modular design allows 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, enabling reliable simulations in complex, high-risk environments.

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

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

The real-case applications to the Malpasset dam-break and the Maule 2010 tsunami stress different aspects of the solver. In Malpasset, characterized by being a short-duration, high-gradient flood in a confined valley, SWEpy maintained physically realistic wetting–drying, conserved mass across advancing fronts, and achieved substantially lower height errors than TELEMAC's finite-volume solver, with arrival times generally within $\sim 18\%$ of observations. By contrast, the Maule tsunami exposed current limitations in basin-scale propagation: all SWEpy configurations underestimated the first-crest amplitude and arrived early at both DART buoys, with WENO+~~FE~~ EE performing best at -28% error in amplitude and -15% error in arrival times, but further developing high-frequency oscillations over long propagation paths. WENO+RK4,3 is oscillation free and reduced dissipation and phase drift relative to lower-order schemes, but appreciable amplitude and timing biases remained. Likely error sources include source smoothing during interpolation, spherical–Cartesian projection error (versus HySEA's spherical formulation), and coarser bathymetric resolution. These results point to clear targets for improvement: projection accuracy, source initialization, and mesh resolution. Addressing these would allow SWEpy to show stronger far-field skills at ocean-basin scale.

From a computational standpoint, SWEpy delivers high throughput even on modest hardware, that's more than 6-years-old at the time. Using a simplified unstructured mesh of approximately 2.5×10^5 elements for benchmarking purposes, the 24-hour Maule 2010 tsunami scenario was simulated in approximately 17 minutes on an NVIDIA GeForce GTX 1650 laptop GPU, representing a speedup of about $\sim 2100\%$ relative to the CPU baseline, ran on an Intel® Core™ i5-10300H processor. These results, achieved through extensive CuPy-based vectorization and memory-efficient data structures, demonstrate that large-scale, unstructured-grid SWEs simulations can be executed well within real-time on widely accessible systems, significantly lowering the entry barrier for high-performance hydrodynamic modeling while retaining scalability for more detailed configurations.

SWEpy is built on the CuPy library, which is primarily developed for the NVIDIA CUDA ecosystem. Although CuPy offers experimental support for AMD GPUs via the HIP/ROCm interface, all validation and performance results presented in this

work rely exclusively on the stable CUDA backend to ensure numerical reproducibility, performance consistency, and software robustness for this initial release. As CuPy's multi-backend support continues to mature, SWEpy is well positioned to benefit from broader GPU compatibility without requiring fundamental changes to its core implementation.

Despite these advances, the present implementation of SWEpy has clear limitations revealed by the validation exercises. In the Maule 2010 tsunami case, amplitude underestimation and arrival-time biases highlight the need for improved bathymetric interpolation, reduced projection error between spherical and Cartesian coordinates, and higher-resolution meshes for basin-scale simulations. In the Malpasset Dam case, complex gridding and bathymetry shows that the wet/dry treatment can interact with the ghost cell boundary conditions and form artificial sources of mass influx, requiring further study. The sub-third-order convergence observed for the quadratic WENO scheme suggests that flux–bathymetry coupling remains a limiting factor for spatial accuracy. Furthermore, the solver inherits the hydrostatic assumption of the SWEs, which can introduce inaccuracies in steep or highly curved flows where vertical accelerations are non-negligible. Ongoing development is therefore directed at enhancing bathymetric handling, coordinate projection, spatial accuracy, and physical modeling to make SWEpy an even more robust and attractive tool for a broad range of hydrodynamic applications.

From a software standpoint, the development roadmap for SWEpy emphasizes continued optimization of GPU efficiency and refinement of its modular architecture. This modular design supports community contributions and allows individual components—such as source terms, numerical schemes, and physical models—to evolve independently. Although the current implementation targets single-GPU execution, future work may explore improved use of multi-core CPUs through process-based parallelism using Python's native multiprocessing. In contrast, implementing an efficient MPI-based parallelization strategy for unstructured grids entirely in Python remains challenging, as communication overhead can quickly outweigh computational gains. Without adopting hybrid C++/Python approaches—such as MPI wrappers around compiled kernels—pure Python MPI solutions often negate the benefits of parallel execution. These limitations further justify the current focus on GPU-centric acceleration, where high throughput can be achieved within a unified and accessible Python framework

Looking ahead, the flexibility of SWEpy makes it a solid foundation for extending its scope well beyond the present applications. Planned developments include additional source-term modules for rainfall and infiltration to enable catchment-scale hydrology, and rheological models for non-Newtonian fluids to simulate landslides, tailings dam breaches, snow avalanches, and debris flows. The existing implementation is also well suited for integrating two-layer shallow-water formulations (2LSWE), following recent advances in central-upwind-type schemes (Cao et al., 2024; Liu, 2021a), to represent stratified and two-phase flows such as water–mud avalanches or estuarine mixing. These upgrades, combined with continued improvements in bathymetric representation, mesh refinement, and large-scale projection accuracy, will strengthen the solver's predictive skill across an even broader spectrum of hydrodynamic hazards. By retaining its modular, open-source, and GPU-accelerated design, SWEpy is poised to become a versatile and high-performance modeling environment, bridging the gap between research prototypes and operational tools for flood and tsunami risk assessment worldwide.

Code and data availability. SWEpy is available for CUDA-compatible devices through GitHub at https://github.com/joaquinmeza90/SWEpy.git (last access: 29 January 2026), under a GPL license. To facilitate onboarding and ensure reproducibility, a comprehensive User Manual and Technical Reference—covering installation, input data formats, and benchmark execution—can be downloaded from the project's official website: **www.hydrology.cl/swepy**. A repository containing test cases showed in Section 5 is available at https://doi.org/10.5281/zenodo.16789890 (last access: 29 January 2026). 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 (EE+WENO and RK4,3+WENO configurations) scenarios simulated are available for visualization and download at https://doi.org/10.5281/zenodo.16798435

Appendix A: Reconstruction Operators

A1 Linear (Minmod) Reconstruction

The minmod linear reconstruction is detailed in Algorithm A1.

Algorithm A1 Minmod linear reconstruction

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

Algorithm A2 Quadratic WENO reconstruction

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

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:

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

shown in 26, 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 $M\boldsymbol{a} = \Delta\boldsymbol{q}$,

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

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

A3 Wet/dry treatment

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

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

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

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 authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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, https://www.tensorflow.org/, software available from tensorflow.org, 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, Volume 2, ASPLOS '24, p. 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, version date: November 2013, 2013.

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

Ayachit, U.: The paraview guide (full color version), Kitware, 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, 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, Natural Hazards and Earth System Sciences, 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 (Mw = 8.8) and Tohoku (Mw = 9.0) earthquakes, Geophysical Research Letters, 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, Advances in Water Resources, 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, Environmental Fluid Mechanics, 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.: Benchmark Problem 2: Run-up of Solitary Waves on a Circular Island, in: International Workshop on Long Wave Modeling of Tsunami Runup, Vicksburg, MS, 1995.

Brunner, G.: HEC-RAS 2D User's Manual Version 6.0, https://www.hec.usace.army.mil/confluence/rasdocs/r2dum/latest, 2021.

Bryson, S. and Levy, D.: Balanced central schemes for the shallow water equations on unstructured grids, SIAM Journal on Scientific Computing, 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, number: 3 Publisher: EDP Sciences, 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, Journal of Computational Physics, 515, 113 273, 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, Environmental Modelling & Software, 145, 105 205, https://doi.org/10.1016/j.envsoft.2021.105205, 2021.

Castro-Orgaz, O., Hager, W. H., et al.: Shallow water hydraulics, Springer, 2019.

Catalan, P. A., Gubler, A., Cañas, J., Zuñiga, C., Zelaya, C., Pizarro, L., Valdes, C., Mena, R., Toledo, E., and and, R. C.: Design and operational implementation of the integrated tsunami forecast and warning system in Chile (SIPAT), Coastal Engineering Journal, 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, Geoscientific Model Development, 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, International Journal for Numerical Methods in Fluids, 78, 355–383, https://doi.org/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.: Applied hydrology, McGraw-hill, 1971.

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

Chu, S., Kurganov, A., and Menshov, I.: New adaptive low-dissipation central-upwind schemes, Applied Numerical Mathematics, 209, 155–170, https://doi.org/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, Geoscientific Model Development, 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, Journal of Computational Physics, 538, 114 189, 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, The 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, 2021.

Desveaux, V. and Masset, A.: A fully well-balanced scheme for shallow water equations with Coriolis force, Communications in Mathematical Sciences, 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 of Conferences, 40, 05 008, https://doi.org/10.1051/e3sconf/20184005008, 2018.

Fuenzalida A., J. A., Meneses, R., Meza, J., and Kusanovic, D.: Adaptive Local Reconstruction for Solving Saint-Venant Equations in Irregular Domains, manuscript 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., 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 Review, 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, 2010.

Greenberg, J. M. and Leroux, A. Y.: A Well-Balanced Scheme for the Numerical Processing of Source Terms in Hyperbolic Equations, SIAM Journal on Numerical Analysis, 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 Dynamics, 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 and Applied Geophysics, 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, 2007.

Jawahar, P. and Kamath, H.: A high-resolution procedure for Euler and Navier–Stokes computations on unstructured grids, Journal of Computational Physics, 164, 165–203, 2000.

Jodhani, K. H., Patel, D., and Madhavan, N.: A review on analysis of flood modelling using different numerical models, Materials Today: Proceedings, 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, Journal of Hydrology, 590, 125 481, 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, Geoscientific Model Development, 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, 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, 2012.

Kurganov, A. and Petrova, G.: Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws, Numerical Methods for Partial Differential Equations, 21, 536–552, https://doi.org/10.1002/num.20049, _eprint: https://onlinelibrary.wiley.com/doi/pdf/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, Communications in Mathematical Sciences, 5, 133 – 160, 2007.

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

Kurganov, A. and Xin, R.: New Low-Dissipation Central-Upwind Schemes, Journal of Scientific Computing, 96, 56, https://doi.org/10.1007/s10915-023-02281-8, 2023.

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, Philip, L.-F.: Development of a tsunami early warning system for the South China Sea, Ocean Engineering, 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, Journal of Computational Physics, 438, 110 380, 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, Journal of Computational Physics, 438, 110 380, https://doi.org/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, Journal of Computational Physics, 374, 213–236, https://doi.org/https://doi.org/10.1016/j.jcp.2018.07.038, 2018.

~~Macias~~

Macías, J., ~~Castro~~Castro, M. J., ~~González-Vida, J. M., de la Asunción, M~~Ortega, S., Escalante, C., and ~~Ortega, S.~~González-Vida, J. M.: ~~HySEA: An operational GPU-based model for Tsunami Early Warning Systems~~Performance Benchmarking of Tsunami-HySEA Model for NTHMP's Inundation Mapping Activities, ~~in: EGU General Assembly Conference Abstracts, EGU General Assembly Conference Abstracts, p. 14217, 2014.~~ Pure and Applied Geophysics, https://doi.org/10.1007/s00024-017-1583-1, 2017.

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, Environmental Modelling & Software, 141, 105 034, 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, pp. 103–135, Springer, 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, Computers & Fluids, 51, 30–34, https://doi.org/10.1016/j.compfluid.2011.07.003, 2011.

Mungov, G., DOC/NOAA/NWS/NDBC > National Data Buoy Center, National Weather Service, NOAA, U.S. Department Of Commerce, and DOC/NOAA/OAR/PMEL > Pacific Marine Environmental Laboratory, OAR, NOAA, U.S. Department Of Commerce: Deep-Ocean Assessment and Reporting of Tsunamis (DART(R)), https://doi.org/10.7289/V5F18WNS, 2005.

Nessyahu, H. and Tadmor, E.: Non-oscillatory central differencing for hyperbolic conservation laws, Journal of computational physics, 87, 408–463, 1990.

Nguyen, T.: Adaptive Central-Upwind Scheme on Triangular Grids for the Shallow Water Model with Variable Density, International Journal of Numerical Analysis and Modeling, 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, 2017.

Parnas, D. L.: On the criteria to be used in decomposing systems into modules, Communications of the 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, Computer Physics Communications, 254, 107 251, 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, 130-136, Citeseer, 2015.

~~Roe, P.: Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics, 43, 357–372, , 1981.~~

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, Advances in Water Resources, 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 Resources Research, 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, Geoscientific Model Development, 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, Journal of Hydroinformatics, 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, p. 23–40, Springer Nature Singapore, 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, Journal of Computational Physics, 430, 110 100, https://doi.org/10.1016/j.jcp.2020.110100, 2021.

Sunder, D., Vaghani, D., and Shukla, R.: Third-Order WENO Schemes on Unstructured Meshes, pp. 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 journal on numerical analysis, 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 and Applied Geophysics, 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, Journal of Scientific Computing, 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, Journal of Waterway, Port, Coastal, and Ocean Engineering, 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, https://docs.python.org/3/whatsnew/3.13.html, 2024.

United Nations: World Urbanization Prospects: The 2018 Revision, ST/ESA/SER.A/420, United Nations, New York, united Nations publication, 2019.

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

Van Leer, B.: Towards the ultimate conservative difference scheme, Journal of computational physics, 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, 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), Advances in Water Resources, 132, 103 392, 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, Applied mathematics and computation, 170, 296–313, 2005.

Zhang, Y.-T. and Shu, C.-W.: ENO and WENO Schemes, p. 103–122, Elsevier, 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 Resources Research, 58, https://doi.org/10.1029/2022wr033214, 2022.

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