**SERGHEI (-SWE) v1.0: a performance portable HPC shallow water solver for hydrology and environmental hydraulics**

Daniel Caviedes-Voullième\(^1,2\), Mario Morales-Hernández\(^3,4\), Matthew R. Norman\(^4\), and Ilhan Özgen-Xian\(^5,6\)

\(^1\)Simulation and Data Lab Terrestrial Systems, Jülich Supercomputing Centre, Forschungszentrum Jülich (Germany)
\(^2\)Institute of Bio- and Geosciences: Agrosphere (IBG-3), Forschungszentrum Jülich (Germany)
\(^3\)Fluid Mechanics, I3A-Universidad de Zaragoza (Spain)
\(^4\)Oak Ridge National Laboratory (USA)
\(^5\)Institute of Geocology, Technische Universität Braunschweig (Germany)
\(^6\)Earth & Environmental Sciences Area, Lawrence Berkeley National Laboratory (USA)

**Correspondence:** Daniel Caviedes-Voullième (d.caviedes.voullieme@fz-juelich.de)

**Abstract.** The Simulation Environment for Geomorphology, Hydrodynamics and Ecohydrology in Integrated form (SERGHEI) is a multi-dimensional, multi-domain and multi-physics model framework for environmental and landscape simulation, designed with an outlook towards Earth System Modelling. It aims to provide a modelling environment for hydrodynamics, ecohydrology, morphodynamics, and, most importantly, interactions and feedbacks among such processes at different levels of complexity and across spatiotemporal scales. The small scale feedbacks and interactions, which warrant high resolution, can result in emergent behaviours manifesting at larger scales, thus warranting large model domains. At the core of SERGHEI’s technical innovation is its HPC implementation, built from scratch on the Kokkos portability layer. Consequently, SERGHEI achieves performance-portability from personal computers to top HPC systems, including GPU-based and heterogeneous systems. SERGHEI relies Kokkos to handle memory spaces, thread management and execution policies for the required backend programming models. In this work we explore combinations of MPI and Kokkos using OpenMP and CUDA backends. In this contribution, we introduce the SERGHEI model framework, and present with detail its first operational module for solving shallow water equations (SERGHEI-SWE). This module is designed to be applicable to hydrological, environmental and consequently Earth System Modelling problems, but also to classical engineering problems such as fluvial or urban flood modelling. We also provide evidence of its applicability by testing it against several well-known benchmarks. We also evaluate its performance on several benchmarks and large scale problems. Finally, SERGHEI-SWE is evaluated in terms of scaling (on several TOP500 HPC systems).
1 Introduction

The upcoming exascale high-performance parallel computing (HPC) systems will enable physics-based geoscientific modelling with unprecedented detail (Alexander et al., 2020). Although the need for such HPC systems is traditionally driven by climate, ocean, and atmospheric modelling, hydrological models are progressively becoming as physical, sophisticated, and computationally intensive. physically-based, integrated hydrological models such as Parflow (Kuffour et al., 2020), Amanzi/ATS (Coon et al., 2019), and Hydrogeosphere (Brunner and Simmons, 2012) are becoming more prominent in hydrological research and Earth System Modelling (ESM) (Fatichi et al., 2016; Paniconi and Putti, 2015), making HPC more and more relevant for computational hydrology (Clark et al., 2017).

Hydrological models, as many other HPC applications, are currently facing challenges in exploiting available and future HPC systems. These challenges arise, not only because of the intrinsic complexity of maintaining complex codes over large periods of time, but because HPC and its hardware are undergoing a large paradigm change (Leiserson et al., 2020; Mann, 2020), which is strongly driven by the end of Moore’s law (Morales-Hernández et al., 2020). In order to gain higher processing capacity, computers will require heterogeneous and specialised hardware (Leiserson et al., 2020), potentially making high-performing code harder to develop and maintain, and demanding for developers to adapt and optimise code for an evolving hardware landscape. It has become clear that upcoming exascale systems will have heterogeneous architectures embedded in modular and reconfigurable architectures (Djemame and Carr, 2020; Suarez et al., 2019) that will consist of different types of CPUs and accelerators, possibly from multiple vendors requiring different programming models. This puts pressure on domain scientists to write portable code that performs efficiently on a range of existing and future HPC architectures (Bauer et al., 2021; Lawrence et al., 2018; Schulthess, 2015), and to ensure the sustainability of such code (Gan et al., 2020).

Different strategies are currently being developed to cope with this grand challenge. One strategy is to offload the architecture-dependent parallelisation tasks to the compiler—see, for example, (Vanderbauwhede and Takemi, 2013; Vanderbauwhede and Davidson, 2018; Vanderbauwhede, 2021). Another strategy is to use an abstraction layer that provides a unified programming interface to different computational backends—a so-called performance portability framework—that allows the same code to be compiled across different HPC architectures. Examples of this strategy include RAJA (Beckingsale et al., 2019) and Kokkos (Edwards et al., 2014; Trott et al., 2021), which are both very similar in their scope and their capability. Both RAJA and Kokkos are C++ libraries that implement a shared-memory programming model to maximise the amount of code that can be compiled across different hardware devices with nearly the same parallel performance. They allow access to several computational backends, in particular multi-GPU and heterogeneous HPC systems.

This paper introduces the Kokkos-based computational (eco)hydrology framework SERGHEI (Simulation Environment for Geomorphology, Hydrodynamics and Ecohydrology in Integrated form) and its surface hydrology module SERGHEI-SWE. The primary aim of SERGHEI’s implementation is scalability and performance-portability. In order to achieve this, SERGHEI is written in C++ and based from scratch on the Kokkos abstraction. Kokkos currently supports CUDA, OpenMP, Thrust, Syvl, and Pthreads as backends. We chose Kokkos over other alternatives, because it is actively engaged in securing the sustainability of its programming model, fostering its partial inclusion into ISO C++ standards (Trott et al., 2021). Indeed, there is an increasing
number of applications in multiple domains leveraging on Kokkos, for example (Bertagna et al., 2019; Demeshko et al., 2018; Grete et al., 2021; Halver et al., 2020; Watkins et al., 2020). Thus, among other similar solutions, Kokkos has been identified as advantageous in terms of performance portability and project sustainability, although perhaps somewhat more invasive and less clear on the resulting code (Artigues et al., 2019). We present the full implementation of the SERGHEI-SWE module, the shallow water equations (SWE) solver for free surface hydrodynamics at the heart of SERGHEI.

SERGHEI-SWE enables the simulation of surface hydrodynamics of overland and stream flow seamlessly and across scales. Historically, hydrological models featuring surface flow have relied on kinematic or zero-inertia (diffusive) approximations due to their apparent simplicity (Caviedes-Voullième et al., 2018; Kollet et al., 2017) and because until the last decade, robust SWE solvers were not available (Caviedes-Voullième et al., 2020a; García-Navarro et al., 2019; Simons et al., 2014; Özgen-Xian et al., 2021). However, the current capabilities of SWE solvers, the increase in computational capabilities, and the need to better exploit parallelism—easier to achieve with explicit solvers than implicit solvers as usually required by diffusive equations (Caviedes-Voullième et al., 2018; Fernández-Pato and García-Navarro, 2016)—has been pushing to replace simplified surface flow models with fully dynamic SWE solvers. There is an increasing number of studies using SWE solvers for rainfall-runoff and overland flow simulations from hillslope to catchment scales—for example, (Bellos and Tsakiris, 2016; Bout and Jetten, 2018; Caviedes-Voullième et al., 2012, 2020a; Costabile and Costanzo, 2021; Costabile et al., 2021; David and Schmalz, 2021; Dullo et al., 2021a, b; Fernández-Pato et al., 2020; García-Alén et al., 2022; Simons et al., 2014; Xia and Liang, 2018). This trend contributes to the transition from engineering hydrology towards Earth System science (Sivapalan, 2018), a shift that motivated by necessity and opportunity, as continental (and larger) ESM will progressively require fully dynamic SWE solvers to cope with increased resolution digital terrain models and the dynamics which respond to them, improved spatiotemporal rainfall data and simulations, as well as increasingly more sophisticated process interactions across scales, from patch, to hillslope to catchments (Fan et al., 2019).

SERGHEI-SWE distinguishes itself from other HPC SWE solvers through a number of key novelties. Firstly, SERGHEI-SWE is open sourced under a permissive BSD license. While there are indeed many GPU-enabled SWE codes, many of these are research codes that are not openly available—for example, (Aureli et al., 2020; Buttinger-Kreuzhuber et al., 2022; Echeverribar et al., 2020; Hou et al., 2020; Lacasta et al., 2014, 2015; Liang et al., 2016; Vacondio et al., 2017)—or commercial codes, such as RiverFlow2D, TUFLOW, HydroAS_2D—see Jodhani et al. (2021) for a recent non-comprehensive review. Open source solvers are a fundamental need for the community, ensuring transparency, reproducibility, and providing a base for model (software) sustainability. We note that open source SWE solvers are becoming increasingly more available—see Table 1. However, only a handful of these freely available models are enabled for GPUs, mostly through CUDA. Fewer of them have multi-GPU capabilities and are capable of fully leveraging HPC hardware. All of these multi-GPU enabled codes are currently dependent on CUDA, and therefore somewhat limited to Nvidia hardware. This leads into the second novelty of SERGHEI-SWE: it is a performance-portable, highly scalable and GPU enabled solver. SERGHEI-SWE generalises hardware (CPU, GPU, accelerators) support to a performance-portability concept through Kokkos. This gives SERGHEI-SWE the key advantage to have a single code base for (currently) OpenMP and CUDA backends, but most importantly, keeps this code base relevant for other backends, such as HIP. This is particularly important, as there is a currently ongoing shift to AMD GPUs,
Table 1. Overview of openly available SWE solvers.

<table>
<thead>
<tr>
<th>Model</th>
<th>Reference</th>
<th>GPU</th>
<th>MPI</th>
<th>Availability</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERGHEI-SWE</td>
<td>this paper</td>
<td>Kokkos</td>
<td>yes</td>
<td>Open source (BSD)</td>
<td>Highly scalable</td>
</tr>
<tr>
<td>TRITON</td>
<td>Morales-Hernández et al. (2021)</td>
<td>CUDA</td>
<td>yes</td>
<td>Open source (BSD)</td>
<td>Highly scalable</td>
</tr>
<tr>
<td>PARFLOOD</td>
<td>Vacondio et al. (2014)</td>
<td>CUDA</td>
<td>yes</td>
<td>-</td>
<td>Highly scalable, source code can be requested, MPI parallelisation by Turchetto et al. (2019)</td>
</tr>
<tr>
<td>HiPIMS</td>
<td>Xia et al. (2019)</td>
<td>CUDA</td>
<td>-</td>
<td>Open source (GPLv3)</td>
<td>Multi-GPU support based on Thrust (on single node)</td>
</tr>
<tr>
<td>DRR/FI</td>
<td>Kobayashi et al. (2015)</td>
<td>-</td>
<td>yes</td>
<td>-</td>
<td>Highly scalable</td>
</tr>
<tr>
<td>SW2D-GPU</td>
<td>Carlotto et al. (2021)</td>
<td>CUDA</td>
<td>-</td>
<td>Open source</td>
<td>-</td>
</tr>
<tr>
<td>LisFlood-FP 8.0</td>
<td>Shaw et al. (2020)</td>
<td>CUDA</td>
<td>-</td>
<td>Open source (BSD)</td>
<td>SWE solver embedded into LisFlood (Bates and Roo, 2000), which originally did not solve SWE.</td>
</tr>
<tr>
<td>IBER</td>
<td>García-Feal et al. (2018)</td>
<td>CUDA</td>
<td>-</td>
<td>Freeware</td>
<td>-</td>
</tr>
<tr>
<td>SW2D-Lemon</td>
<td>Caldas Steintraesser et al. (2021)</td>
<td>-</td>
<td>-</td>
<td>Freeware</td>
<td>Source code can be requested</td>
</tr>
<tr>
<td>B-flood</td>
<td>Kirstetter et al. (2021)</td>
<td>-</td>
<td>-</td>
<td>Open source (GPL)</td>
<td>Adaptive mesh refinement</td>
</tr>
<tr>
<td>FullSWOF</td>
<td>Delestre et al. (2017)</td>
<td>-</td>
<td>yes</td>
<td>Open source (CeCILL)</td>
<td>MPI parallelisation by (Wittmann et al., 2017)</td>
</tr>
<tr>
<td>TELEMAC</td>
<td>Moulinec et al. (2011)</td>
<td>-</td>
<td>yes</td>
<td>Open source (GPLv3/GPLv2)</td>
<td>-</td>
</tr>
<tr>
<td>GeoClaw</td>
<td>Berger et al. (2011)</td>
<td>-</td>
<td>yes</td>
<td>Open source (BSD)</td>
<td>Adaptive mesh refinement</td>
</tr>
<tr>
<td>HEC-RAS2D</td>
<td>Brunner (2021)</td>
<td>-</td>
<td>-</td>
<td>Freeware</td>
<td>-</td>
</tr>
<tr>
<td>hms</td>
<td>Simons et al. (2014)</td>
<td>-</td>
<td>yes</td>
<td>Open source (GPL)</td>
<td>MPI parallelisation by Steffen et al. (2020)</td>
</tr>
</tbody>
</table>

with the most recent leading TOP 500 systems –Frontier and LUMI– and upcoming HPC systems (e.g., El Capitan) rely on AMD GPUs.

Another important novelty of SERGHEI-SWE is that it is specifically designed to cope with rainfall-runoff problems in both natural and urban environments, fluvial problems, and other flows of broad hydrological and environmental interest that occur on (eco)hydrological time scales. This separates it from some of the arguably more engineering-oriented established models in Table 1, which are more geared towards flood modelling.

SERGHEI-SWE has been developed harnessing the past 15 years of numerical advances in the solution of SWE, ranging from fundamental numerical formulations (García-Navarro et al., 2019; Morales-Hernández et al., 2020) to HPC GPU implementations (Brodtkorb et al., 2012; Hou et al., 2020; Lacasta et al., 2014, 2015; Liang et al., 2016; Vacondio et al., 2017; Sharif et al., 2020).
2 Mathematical and numerical model

SERGHEI-SWE is based on the resolution of the two-dimensional (2D) shallow water equations, that can be expressed in a compact differential conservative form as:

\[
\begin{align*}
\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} &= S_r + S_b + S_f, \\
U &= \begin{bmatrix} h \\ q_x \\ q_y \end{bmatrix}, \\
F &= \begin{bmatrix} q_x \\ \frac{q_x^2}{h} + \frac{1}{2}gh^2 \\ \frac{q_xq_y}{h} \end{bmatrix}, \\
G &= \begin{bmatrix} q_y \\ \frac{q_xq_y}{h} + \frac{1}{2}gh^2 \\ \frac{q_y^2}{h} + \frac{1}{2}gh^2 \end{bmatrix}, \\
S_r &= \begin{bmatrix} r_o - r_f \\ 0 \\ 0 \end{bmatrix}, \\
S_b &= \begin{bmatrix} 0 \\ -gh \frac{\partial z}{\partial x} \\ -gh \frac{\partial z}{\partial y} \end{bmatrix}, \\
S_f &= \begin{bmatrix} 0 \\ -\sigma_x \\ -\sigma_y \end{bmatrix}.
\end{align*}
\]

Here, \(t\) [T] is time, \(x\) [L] and \(y\) [L] are Cartesian coordinates, \(U\) is the vector of conserved variables (that is to say the unknowns of the system) containing the water depth, \(h\) [L], and the unit discharges in \(x\) and \(y\) directions, called \(q_x = hu\) [L^2 T^{-1}] and \(q_y = hv\) [L^2 T^{-1}] respectively. \(F\) and \(G\) are the fluxes of these conserved variables with gravitational acceleration \(g\) [LT^{-2}]. The mass source terms \(S_r\) account for rainfall, \(r_o\) [LT^{-1}], and infiltration/exfiltration, \(r_f\) [LT^{-1}]. The momentum source terms include gravitational bed slope terms, \(S_b\), expressed according to the gradient of the elevation \(z\) [L]; and friction terms, \(S_f\), as a function of the friction slope \(\sigma\). This friction slope is often modelled by means of Gauckler-Manning’s equation in terms of Manning’s roughness coefficient \(n\) [TL^{-1/3}], but also frequently with the Chezy and the Darcy-Weisbach formulations (Caviedes-Voullième et al., 2020a).

SERGHEI-SWE uses a first-order accurate upwind finite-volume scheme with a forward Euler time integration to solve the system of equations (1) on uniform Cartesian grids with grid spacing \(\Delta x\) [L]. The numerical scheme, presented in detail in (Morales-Hernández et al., 2021), harnesses many solutions that have been reported in the literature in the past decade, ensuring that all desirable properties of the scheme (well-balancing, depth-positivity, stability, robustness) are preserved under the complex conditions of realistic environmental problems. In particular, we require the numerical scheme to stay robust and accurate in the presence of arbitrary rough topography and shallow water depths with wetting and drying.

Well-balancing and water depth positivity are ensured by solving numerical fluxes at each cell edge \(k\) with augmented Riemann solvers (Murillo and García-Navarro, 2010, 2012) based on the Roe linearisation (Roe, 1981). In fluctuation form, the rule for updating the conserved variables in cell \(i\) from time step \(n\) to time step \(n + 1\) reads:

\[
U_i^n + 1 = U_i^n + (r_o - r_f)_i^n \Delta t,
\]

followed by

\[
U_i^n = U_i^n - \frac{\Delta t}{\Delta x} \sum_{k=1}^{4} \sum_{m=1}^{3} \frac{\lambda^-}{\lambda} \left[ (\hat{\lambda} \hat{a} - \hat{\beta}) \hat{e} \right]_{m,k}^n,
\]

with

\[
\left[ (\hat{\lambda} \hat{a} - \hat{\beta}) \hat{e} \right]_{m,k}^n = \frac{1}{2} \left( F_{i-1/2,m,k}^n + G_{i-1/2,m,k}^n - F_{i+1/2,m,k}^n - G_{i+1/2,m,k}^n \right),
\]

where \(F_{i-1/2,m,k}^n\) and \(G_{i-1/2,m,k}^n\) are the numerical fluxes at the cell edge \((i-1/2)\).
where $\tilde{\lambda}$ and $\tilde{e}$ are the eigenvalues and eigenvectors of the linearised system of equations, $\tilde{\alpha}$ and $\tilde{\beta}$ are the fluxes and bed slope and friction source term linearisations respectively, and the minus-sign accounts for the upwind discretisation. Note that all the tilde variables are defined at each computational edge. The time step $\Delta t$ is restricted to ensure stability, following the Courant-Friedrich-Lewy (CFL) condition:

$$\Delta t = \text{CFL} \min_i \left\{ \frac{\Delta x}{q_x h_i} + \sqrt{gh_i}, \frac{\Delta y}{q_y h_i} + \sqrt{gh_i} \right\}$$

Although the wave speed values are formally defined at the interfaces, the corresponding cell values are used instead for the CFL condition. As pointed in (Morales-Hernández et al., 2021), this approach does not compromise the stability of the system, but accelerates the computations and simplifies the implementation.

3 HPC implementation

3.1 Domain decomposition

The surface domain is a two-dimensional plane, discretised by a Cartesian grid with a total cell number of $N_t = N_x N_y$, where $N_x$ and $N_y$ are the number of cells in $x$ and $y$-directions, respectively. Operations are usually performed per subdomain, each one associated with an MPI rank. During initialisation, each MPI process constructs a local subdomain with $n_x$ cells in $x$-direction and $n_y$ cells in $y$-direction. The user specifies the number of subdomains in each Cartesian direction at runtime and SERGHEI determines the subdomain size from this information. Subdomains are the same size, except for correction due to non-integer-divisible decompositions. In order to communicate information across subdomains, SERGHEI uses so-called halo cells, non-physical cells on the boundaries of the subdomain that overlap with physical cells from neighbouring subdomains. The halo cells augment the number of cells in $x$- and $y$-direction by 1 at each boundary. Thus, the subdomain size is $n_t = (n_x + 2)(n_y + 2)$. The definitions are sketched—without loss of generality—for a square-shaped subdomain in Figure 1 and the way these subdomains overlap in the global domain is sketched in Figure 2 (left). Halo cells are not updated as part of the time stepping. Instead, they are updated by receiving data from the neighbouring subdomain, a process which naturally requires MPI communications.

Besides the global cell index that ranges from 0 to $N_t$, each subdomain uses two sets of local indices to access data stored in its cells. The first set spans over all physical cells inside the subdomain and the second index spans over both halo cells and physical cells—see Figure 1. The second set maps into memory position. For example, in order to access the physical cell 14 in Figure 1, one has to access memory position 27.

3.2 Data exchange between subdomains

The underlying methods for data exchange between subdomains are centered on the subdomains rather than on the interfaces. Data is exchanged through MPI-based send and receive calls (non-blocking) that aggregate data in the halo cells across the
Figure 1. Domain decomposition and indexing in SERGHEI: A subdomain consists of physical cells (white) and halo cells (gray). SERGHEI uses two sets of indices: an index for physical cells (left) and an index for all cells including the halo cells (right).

Figure 2. Data exchange between subdomains in SERGHEI: In the global surface domain, subdomains overlap with each other through their halo cells (left). These halo cells are used to exchange data between the subdomains (right).

Subdomains. Note that, by default, Kokkos implicitly assumes that the MPI library is GPU-aware, allowing GPU to GPU communication provided that the MPI libraries support this feature. Figure 2 (right) illustrates the concept of sending a halo buffer containing state variables from subdomain 1 to update halo cells of subdomain 0. The halo buffer contains state variables for \( n_y \) cells, grouped as water depth \( (h) \), unit discharge in \(-x\)-direction \( (hu) \), and unit discharge in \(-y\)-direction \( (hv) \).
3.3 Performance portable implementation

Further parallelism is achieved per subdomain through the Kokkos framework, which allows the user to choose between shared memory parallelism and GPU backends for further acceleration. SERGHEI’s implementation makes use of the Kokkos concept of Views, which are memory space aware abstractions. For example, for arrays of real numbers, SERGHEI defines a type realArr, based on View. This takes the form of Listing 1 for the shared (host) memory space and Listing 2 for the Unified Virtual Memory (UVM) GPU-device CUDA memory space.

**Listing 1. realArr definition based on View for CPU**
```cpp
typedef Kokkos::View<real*, Kokkos::LayoutRight> realArr;
```
and for a CUDA backend, making use of unified memory (CudaUVMSpace) is

**Listing 2. realArr definition based on View for GPU**
```cpp
typedef Kokkos::View<real*, Kokkos::LayoutRight,
    Kokkos::Device<Kokkos::Cuda, Kokkos::CudaUVMSpace>> realArr;
```

Similar definitions can be constructed for integer arrays. These arrays describe spatially distributed fields, such as conserved variables, model parameters, and forcing data. Deriving these arrays from View allows us to operate on them via Kokkos to achieve performance portability.

Conceptually, the SERGHEI-SWE solver consists of two computationally intensive kernels: (i) cell-spanning and (ii) edge-spanning kernels. The update of the conserved variables following Equation 2 results in a kernel around a cell-spanning loop. These cell-spanning loops are the most frequent ones in SERGHEI-SWE and are used for many processes, of different computational demand. The standard C++ implementation of such a kernel is illustrated in Listing 3, which spans indices $i$ and $j$ of a 2D cartesian grid. Here, the loops may be parallelised using, for example, OpenMP or CUDA. However, such a direct implementation of, for example an OpenMP parallelisation, would not automatically allow leveraging GPUs. That is to say, such an implementation is not portable.

**Listing 3. Conserved variable update in standard C++**
```cpp
inline void computeNewState(State &state, const Domain &dom, const SourceSinkData &ss){
    for (int j=0; j<dom.ny; j++){
        for (int i=0; i<dom.nx; i++){
            // computationally intensive code to update cells
        }
    }
}
```

In order to achieve the desired portability, we replace the standard `for` by a `Kokkos::parallel_for`, which enables a lambda function, is minimally intrusive and reformulates this kernel to the code shown in Listing 4. As a result, this implementation can be compiled for both OpenMP applications and GPUs with Kokkos handling the low-level parallelism on different backends.
Edge-spanning loops are conceptually necessary to compute numerical fluxes (Equation 2). Although numerical fluxes can be computed in a cell-centered fashion, this would lead to inefficiencies due to duplicated computations. In Listing 5 we illustrate the edge-spanning kernel solving the numerical fluxes in SERGHEI-SWE. Notably, Listing 5 is indexed by cells, and the construction of edge-wise tuples occurs inside of the kernel. This bypasses the need for additional memory structures to hold edge-based information, but only for Cartesian meshes. Generalisation to adaptive or unstructured meshes would require explicitly an edge-based loop with an additional View of size equal to the number of edges.

### Listing 5. Flux computations

```cpp
inline void computeDeltaFluxXRoe(State &state, Domain const &dom, Parallel &par) {
    Kokkos::parallel_for(dom.nCells, KOKKOS_LAMBDA(int iGlob) {
        int i, j, nCells;
        int id1, id2;
        unpackIndices(iGlob, dom.ny+2*hc, dom.nx+2*hc, j, i);
        if (i>hc-2 && i<dom.nx+hc && j>hc-1 && j<dom.ny+hc) {
            nCells=dom.nCells;
            id1=iGlob;
            id2=j+(dom.nx+2*hc)+1;
            // computationally intensive code to compute fluxes at the edge between cells id1 and id2
        }
    })
}
```

4 Verification and validation

In this section we report evidence supporting the claim that SERGHEI-SWE is an accurate, robust and efficient shallow water solver. The formal accuracy testing strategy is based on several well-known benchmark cases with well-defined reference solutions. Herein, for brevity, we focus only on the results of these tests, while providing a minimal presentation of the setups. We refer the interested reader to the original publications (and to the many instances in which these tests have been used) for further details on the geometries, parametrisations and forcing.

We purposely report an extensive testing exercise to show the wide applicability of SERGHEI across hydraulic and hydrological problems, with a wide range of the available benchmark tests. Analytical, experimental and field-scale tests are included. The first are aimed at showing formal convergence and accuracy. The experimental cases are meant as validation of the capabilities of the model to reach physically meaningful solutions under a variety of conditions. The field-scale tests
showcase the applicability of the solver for real problems, and allow for strenuous computational tasks to show performance, efficiency and parallel-scaling. All solutions reported here were computed using double precision arithmetic.

4.1 Analytical steady flows

We test SERGHEI’s capability to capture moving equilibria in a number of steady flow test cases compiled in (Delestre et al., 2013). Details of the test cases for reproduction purposes can be retrieved from (Delestre et al., 2013) and the accompanying software SWASHES—in this work, we use SWASHES version 1.03. In the following test cases, the domain is always discretised using 1000 finite volumes.

4.1.1 Bumps

These tests feature a smooth bump in a one-dimensional, frictionless domain which can be used to validate the C-property, well-balancing, and the shock-capturing ability of the numerical solver (Morales-Hernández et al., 2012; Murillo and García-Navarro, 2012). Firstly, we demonstrate demonstrate that SERGHEI preserves a lake-at-rest in the presence of an immersed and emerged bump in Figure 3. The model predictions from SERGHEI matches the analytical solution obtained through SWASHES, which verifies the C-property of the implemented solver.

To show well-balancing under steady flow, we test SERGHEI for three steady state flow configurations: subcritical flow, transcritical flow without a shock, and transcritical flow with a shock wave. Figure 4 shows SERGHEI predictions plotted against analytical solutions (SWASHES), with very good agreement. The constant unit discharge is captured with machine accuracy without oscillations at the shock, which is an inherent feature of the augmented Roe solver (Murillo and García-Navarro, 2010).
4.1.2 Flumes

A series of test cases featuring one-dimensional flumes with varying geometry based on analytical solution by MacDonald et al. (1995) are studied. These tests are well-known and widely used as benchmark solutions (e.g., Caviedes-Voullième and Kesserwani, 2015; Delestre et al., 2013; Kesserwani et al., 2019; Morales-Hernández et al., 2012; Murillo and García-Navarro, 2012). At steady state, local acceleration terms and source terms balance each other out such that the free surface water elevation becomes a function of bed slope and friction source terms. Thus, these test cases can be used to validate the implementation of these source terms and the well-balanced nature of the complete numerical scheme. Since pluvial flow is usually dominated by these two terms, verifying well-balancing and proper source term discretisation and implementation is especially important for solvers targeting such applications.

Figure 4. Analytical steady flows: Bumps. SERGHEI captures moving equilibria solutions for subcritical (top left), transcritical without a shock (top right), and transcritical with a shock (bottom center) test cases.
Figure 5. Analytical steady flows: Flumes. SERGHEI captures moving equilibria solutions for subcritical (top left), transcritical without a shock (top right), supercritical (bottom left), and transcritical with a shock (bottom right) test cases. Note that the solution is stable (no oscillations) and well-balanced (discharge remains constant along the flume).

Figure 5 shows comparisons between SERGHEI and analytical solutions obtained through SWASHES. Overall, good agreement between SERGHEI and analytical solutions is obtained. Note that the unit discharge is captured with machine accuracy in the presence of friction and bottom changes, which is mainly due to the upwind friction discretization used in the SERGHEI solver. As reported by Burguete et al. (2008); Murillo et al. (2009), a centered friction discretization does not ensure a perfect balance between fluxes and source terms for steady states even if using the improved discretisation by Xia et al. (2017).

Finally, MacDonald-type solutions can be constructed for frictionless flumes to study the bed slope source term implementation in isolation. We present a frictionless test case with SERGHEI that is not part of the SWASHES benchmark compilation. We discretise the bed elevation of the flume as

$$z(x) = C_0 - \frac{1}{2} \exp(-0.001x) - \frac{2q_0^2 \exp(0.002x)}{g},$$

(5)
Figure 6. Analytical steady flows: Flumes. SERGHEI captures moving equilibrium solution for frictionless test case, with a stable and well-balanced solution.

<table>
<thead>
<tr>
<th>Case</th>
<th>$L_1$ (m)</th>
<th>$L_2$ (m)</th>
<th>$L_\infty$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1</td>
<td>0.2</td>
<td>0.01826</td>
<td>0.00249</td>
</tr>
<tr>
<td>3.1.2</td>
<td>0.1</td>
<td>0.01467</td>
<td>0.00249</td>
</tr>
<tr>
<td>3.1.3</td>
<td>0.293</td>
<td>0.02618</td>
<td>0.00332</td>
</tr>
<tr>
<td>3.1.4</td>
<td>0.693</td>
<td>0.0306</td>
<td>0.00356</td>
</tr>
<tr>
<td>3.1.5</td>
<td>0.371</td>
<td>0.07285</td>
<td>0.06984</td>
</tr>
<tr>
<td>3.2.1a</td>
<td>1.0389</td>
<td>0.03805</td>
<td>0.00191</td>
</tr>
<tr>
<td>3.2.1b</td>
<td>0.68584</td>
<td>0.01909</td>
<td>0.0015</td>
</tr>
<tr>
<td>3.2.1c</td>
<td>5.21459</td>
<td>0.12162</td>
<td>0.00435</td>
</tr>
<tr>
<td>3.2.1d</td>
<td>1.02096</td>
<td>0.06826</td>
<td>0.0622</td>
</tr>
<tr>
<td>w/o fr.</td>
<td>0.74571</td>
<td>0.02743</td>
<td>0.00178</td>
</tr>
</tbody>
</table>

Table 2. Analytical steady flows: Summary of $L$-norms for errors in water depth; $L$-norms for errors in unit discharge are in the range of machine accuracy and omitted here.

where $C_0$ is an arbitrary integration constant and $q_0$ is a specified unit discharge. The water depth for this topography is

$$h(x) = \frac{1}{2} \exp(-0.001x).$$

(6)

Using $C_0 = 1.0$ m and $q_0 = 1.0$ m$^2$/s, we obtain the solution plotted in Figure 6. SERGHEI’s prediction and the analytical solution show good agreement.

$L$-norms for errors in water depth are summarised in Table 2 for the sake of completeness. The $L$-norms for errors in unit discharge are in the range of machine accuracy for all cases and omitted here.
4.2 Analytical dam breaks

We verify SERGHEI’s capability to capture transient flow for a number of test cases compiled in (Delestre et al., 2013). Dam break problems are defined by an initial discontinuity in the water depth in the domain $h(x)$, such that

$$h(x) = \begin{cases} h_L & \text{if } x \leq x_0, \\ h_R & \text{otherwise,} \end{cases}$$

(7)

where $h_L$ denotes a specified water depth on the left hand-side of the location of the discontinuity $x_0$ and $h_R$ denotes the specified water depth on the right hand-side of $x_0$. Initial velocities are nil in the entire domain. In the following, we report empirical evidence of the numerical schemes mesh convergence property by comparing model predictions for test cases with 100, 1000, 10000, and 100000 elements, respectively.

4.2.1 Dam break over a wet bed without friction — A

The dam break on wet bed without friction test case is configured by setting water depths in the domain as $h_L = 0.005m$ and $h_R = 0.001m$. The domain is 10 m long, and the discontinuity is located at $x_0 = 5m$. The total run time is 6s. Figure 7 shows the model results obtained on successively refined grids, compared against the analytical solution by Stoker. Errors for this test case are reported in Table 3. We also report the observed convergence rate $R$, calculated on the basis of the $L_1$-norm. As the grid is refined, the model result converges to the analytical solution. Due to the discontinuities in the solution, the observed convergence rate is below the theoretical convergence rate of $R = 1$.

4.2.2 Dam break over a dry bed without friction — B

Flow featuring depth close to dry bed is a special case for the numerical solver, because regular wave speed estimations become invalid Toro (2001). In this test case, initial conditions are chosen to be $h_L = 0.005m$ and $h_R = 0m$, that is to say dry. As in the previous test case A, the domain is 10 m long, the discontinuity is located at $x_0 = 5m$, and the total run time is 6s. Model results are plotted against the analytical solution by Ritter for different grid resolutions in Figure 8. The model results converge to the analytical solution as the grid is refined. This is also seen in Table 3, where errors and convergence rates for this test case are summarised. Again, the observed convergence rate is below the theoretical convergence rate of $R = 1$, because of the increased complexity introduced by the discontinuity in the solution and the presence of dry bed.

4.3 Analytical oscillations

We present transient two-dimensional test cases with moving wet-dry fronts that consider the periodical movement of water in a parabolic bowl, so-called oscillations that have been studied by Thacker (1981). We replicate two cases from the SWASHES compilation (Delestre et al., 2013), using a mesh spacing of $\Delta x = 0.01m$. 

14
Figure 7. Dam break on wet bed without friction: Model predictions for different number of grid cells. SERGHEI converges to the analytical solution (Stoker’s solution) as the grid is refined.

<table>
<thead>
<tr>
<th>Case</th>
<th>n</th>
<th>$L_1(h)$ (m)</th>
<th>$L_2(h)$ (m)</th>
<th>$R(h)$ (m)</th>
<th>$L_1(u)$ (m/s)</th>
<th>$L_2(u)$ (m/s)</th>
<th>$R(u)$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>0.01623</td>
<td>0.03303</td>
<td>-</td>
<td>0.11194</td>
<td>0.14115</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>1000</td>
<td>0.00265</td>
<td>0.00932</td>
<td>0.79</td>
<td>0.01842</td>
<td>0.0424</td>
<td>0.78</td>
</tr>
<tr>
<td>A</td>
<td>10000</td>
<td>0.00041</td>
<td>0.00327</td>
<td>0.81</td>
<td>0.00272</td>
<td>0.01458</td>
<td>0.83</td>
</tr>
<tr>
<td>A</td>
<td>100000</td>
<td>$6 \times 10^{-5}$</td>
<td>0.00125</td>
<td>0.83</td>
<td>0.00037</td>
<td>0.00581</td>
<td>0.87</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>0.01566</td>
<td>0.02343</td>
<td>-</td>
<td>0.23</td>
<td>0.526</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>1000</td>
<td>0.00396</td>
<td>0.00645</td>
<td>0.6</td>
<td>0.138</td>
<td>0.4053</td>
<td>0.22</td>
</tr>
<tr>
<td>B</td>
<td>10000</td>
<td>0.00068</td>
<td>0.00137</td>
<td>0.76</td>
<td>0.08169</td>
<td>0.34</td>
<td>0.22</td>
</tr>
<tr>
<td>B</td>
<td>100000</td>
<td>0.0001</td>
<td>0.00026</td>
<td>0.83</td>
<td>0.04193</td>
<td>0.248</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Table 3. Analytical dam breaks: L-norms and empirical convergence rates ($R$) for water depth ($h$) and velocity ($h$)
4.3.1 Radially-symmetrical paraboloid

The first test case is a radially symmetrical oscillation in a frictionless paraboloid. The topography is defined as

\[
\begin{align*}
  z(r) &= -h_0 \left(1 - \frac{r^2}{a^2}\right), \\
  r &= \sqrt{(x-L/2)^2 + (y-L/2)^2},
\end{align*}
\]

where \( r \) is the radius, \( h_0 \) is the water depth at the centre of the paraboloid, \( a \) is the distance from the centre to the zero elevation shoreline, \( L \) is the length of the square-shaped domain, and \( x \) and \( y \) denote coordinates inside the domain. The analytical solution is derived in (Thacker, 1981). We use the same values as Delestre et al. (2013), that is \( h_0 = 1 \) m, \( a = 1 \) m, and \( L = 4 \) m. The simulation is run for 3 periods, with a spatial resolution of \( \delta x = 0.01 \) m. Figure 9 shows the numerical and analytical solution at four different times. Model results show good agreement with the analytical solution.

4.3.2 Planar surface in a paraboloid

The more established test case by (Thacker, 1981) is the periodic oscillation of a planar surface in a frictionless paraboloid. This case has been extensively used for validation of shallow water solvers, for example (Aureli et al., 2008; Dazzi et al., 2018; Liang et al., 2015; Murillo and García-Navarro, 2010; V connectio et al., 2014; Zhao et al., 2019), because of its rather complex
Figure 9. Radially-symmetrical paraboloid: Snapshots of water depth by the model compared to the analytical solution (contour lines). Period $T = 2.242851 \text{s}$

2D nature and the presence of moving wet/dry fronts. The topography is again given by Equation 8 with the same choice of parameters and discretisation as before. The analytical solution can be found in (Thacker, 1981; Delestre et al., 2013). The simulation is run for 3 full periods. Snapshots of the simulation are plotted in Figure 10 and compared to the analytical solution. The model results agree well with the analytical solution after three periods, with slightly growing phase error, as is commonly observed on this test case.
Govindaraju et al. (1990) presented an analytical solution to a variable rainfall over a sloping plane, which is commonly used (Caviedes-Voullième et al., 2020a; Gottardi and Venutelli, 2008; Singh et al., 2015). The plane is 21.945 m long, with a slope of 0.04. We select rainfall from Govindaraju et al. (1990), which has to distinct rainfall peaks. Friction is modeled with Chezy’s equation, with a roughness coefficient of $1.767 \text{ m}^{1/2}\text{s}^{-1}$. The computational domain was defined by a $200 \times 10$ grid, with $\Delta x = 0.109725 \text{ m}$.

Figure 10. Planar surface in a paraboloid: Snapshots of water depth by the model compared to the analytical solution (contour lines). Period $T = 2.242851 \text{ s}$

4.4 Variable rainfall over a sloping plane

Govindaraju et al. (1990) presented an analytical solution to a variable rainfall over a sloping plane, which is commonly used (Caviedes-Voullième et al., 2020a; Gottardi and Venutelli, 2008; Singh et al., 2015). The plane is 21.945 m long, with a slope of 0.04. We select rainfall from Govindaraju et al. (1990), which has to distinct rainfall peaks. Friction is modeled with Chezy’s equation, with a roughness coefficient of $1.767 \text{ m}^{1/2}\text{s}^{-1}$. The computational domain was defined by a $200 \times 10$ grid, with $\Delta x = 0.109725 \text{ m}$. 

Figure 10. Planar surface in a paraboloid: Snapshots of water depth by the model compared to the analytical solution (contour lines). Period $T = 2.242851 \text{ s}$
The simulated discharge hydrograph at the outlet is compared against the analytical solution in Figure 11. The numerical solutions match the analytical one very well. The only relevant difference occurs in the magnitude of the second discharge peak, which is slightly underestimated in the simulation.

![Discharge Hydrograph](image)

**Figure 11.** Simulated and analytical discharge for the analytical case of rainfall in a flume

5 Laboratory-scale experiments

5.1 Experimental dam-break over a triangular sill

Hiver (2000) presented a large flume experiment of a dam-break over a triangular sill, which is a standard benchmark in dam-break problems (Caviedes-Voullième and Kesserwani, 2015; Bruwier et al., 2016; Kesserwani and Liang, 2010; Loukili and Soulaïmani, 2007; Murillo and García-Navarro, 2012; Yu and Duan, 2017; Zhou et al., 2013), together with the reduced scale version (Soares-Frazão, 2007; Hou et al., 2013a, b; Yu and Duan, 2017).

The computational domain was discretised with a $380 \times 5$ grid, with a $\delta x = 0.1 \text{ m}$ resolution. Figure 12 shows simulated and experimental results for the triangular sill case. A very good agreement can be observed, both in terms of peak depths occurring whenever the shock wave passes through a gauge, and in the timing of the shock wave movement. The simulations tend to slightly overestimate the peaks of the shock wave, as well as overestimating the waves downstream of the sill (see 12(g)). Both behaviours are well documented in the literature.

5.2 Experimental idealised urban dam-break

A laboratory-scale experiment of a dam-break over an idealised urban area was reported by (Soares-Frazão and Zech, 2008) in a concrete channel including 25 obstacles representing buildings. It is widely used in the shallow-water community (Abderrezak et al., 2008; Caviedes-Voullième et al., 2020b; Ginting, 2019; Hartanto et al., 2011; Jeong et al., 2012; Özgen et al., 2015; Petaccia et al., 2010; Wang et al., 2017) because of its fundamental phenomenological interest and because it is demanding in
terms of numerical stability and model performance. The small buildings and streets in the geometry require sufficiently high resolution, both to capture the geometry, and to capture the complex flow phenomena which is triggered in the streets. Experimental measurements of transient water depth exist at different locations, including in between the buildings. A resolution of 2 cm was used for the simulated results in Figure 13, together with experimental data.

5.3 Experimental steady and dam-break flows over complex geometry

Martínez-Aranda et al. (2018) presented experimental results of steady and transient flows over several obstacles, while recording transient 3D water surface elevation in the region of interest. We selected the so-called G3 case, and simulated both a
Figure 13. Simulated (lines) and experimental (points) water depth profiles at $y = 0.2$ m, at four times, for the idealised urban dam-break case.

dam-break and steady flow. The experiment took place in a double-sloped plexiglass flume, 6 m long, and 24 cm wide. The obstacles in this case are a symmetric contraction and a rectangular obstacle on the centerline, downstream of the contraction. For both cases the flume (including the upstream wider reservoir) was discretised at a 5 mm resolution, resulting in a computational domain with 106887 cells. Manning’s roughness was set to $0.01$ m$^{-1/3}$. The steady simulation was run from an initial state with uniform depth $h = 5$ cm up to $t = 300$ s. The dam-break simulation duration was 40 s.

The steady flow case had a discharge of 2.5 L/s. Steady water surface results in the obstacle region are shown in Figure 14, for a centerline profile ($y = 0$) and a cross section at the rectangular obstacle, specifically at $x = 2.4$ (the coordinate system is set at the center of the flume inlet gate). The simulation results approximate experimental results well. The mismatches are similar to those analysed by Martínez-Aranda et al. (2018) and can be attributed to turbulent and 3D phenomena near the obstacles.

The dam-break case is triggered by a sudden opening of the gate followed by a wave advancing along the dry flume. Results for this case at three gauge points are shown in Figure 15. Again, the simulations approximate experiments well, capturing both the overall behaviour of the water depths and the arrival of the dam break wave, with local errors attributable to the violent dynamics (Martínez-Aranda et al., 2018).
5.4 Experimental unsteady flow over an island

Briggs et al. (1995) presented an experimental test of an unsteady flow over a conical island. This test has been extensively used for benchmarking (Bradford and Sanders, 2002; Choi et al., 2007; García-Navarro et al., 2019; Hou et al., 2013b; Liu et al., 1995; Lynett et al., 2002; Nikolos and Delis, 2009). A truncated cone of base diameter 7.2 m, top diameter 2.2 m and 0.625 m high, was placed at the centre of a 26 × 27.6 m smooth and flat domain. An initial hydrostatic water level of \( h + z = 0.32 \) m was set, and a wave was imposed on the boundary following

\[
h_b = h_0 + A \text{sech}^2 \left( \frac{B(t - T)}{C} \right)
\]

\[
B = \sqrt{gh_0 \left( 1 + \frac{A}{2h_0} \right)}
\]

\[
C = h_0 \sqrt{\frac{4h_0B}{3A\sqrt{gh_0}}}
\]
Figure 16 shows results for a simulation with a 2.5 cm resolution, resulting in 1.2 million cells. A roughness coefficient of 0.013sm$^{-1/3}$ was used for the concrete surface. The results are comparable to previous solutions in the literature, in general reproducing well the water surface surface, with some delay over experimental measurements.

**Figure 16.** Simulated and experimental results of unsteady flow over an island

### 5.5 Experimental laboratory scale tsunami

A 1:400 scale experiment of a tsunami run-up over the Monai valley was reported by (Matsuyama and Tanaka, 2001; third international workshop on long-wave run-up models., 2004), providing experimental data on the temporal evolution of the water surface at three locations, and of the maximum run-up. A laboratory basin of 2.05 × 3.4 m was used to create a physical scale model of the Monai coastline. A tsunami was simulated by appropriate forcing of the boundary conditions. This experiment has been extensively used to benchmark SWE solvers (Arpaia and Ricchiuto, 2018; Caviedes-Voullième et al., 2020b; Hou et al., 2015, 2018; Kesserwani and Liang, 2020; Morales-Hernández et al., 2014; Murillo et al., 2009; Murillo and García-Navarro, 2012; Nikolos and Delis, 2009; Serrano-Pacheco et al., 2009; Vater et al., 2019). The domain has dimensions and was discretised with a resolution of 1.4 cm, producing 95892 elements. Simulated water surface elevations are shown together with the experimental measurements in Figure 17 at three gauge locations. The results agree well with experimental measurements, both in the water surface elevations and the arrival times of the waves.

### 5.6 Experimental rainfall-runoff over an idealised urban area

Cea et al. (2010a) presented experimental and numerical results for a range of laboratory scale rainfall-runoff experiments on an impervious surface with different arrangements of buildings, which have been frequently used for model validation (Caviedes-Voullième et al., 2020a; Cea et al., 2010b; Cea and Bladé, 2015; Fernández-Pato et al., 2016; Su et al., 2017; Xia et al., 2017). This laboratory scale test includes non-trivial topographies, small water layers and wetting/drying fronts, making it a good benchmark for realistic rainfall-runoff conditions.

The dimensions of the experimental flume are 2 × 2.5 m. Here, we select one building arrangement named A12 by Cea et al. (2010a). The original DEM is available (from Cea et al. (2010a)) at a resolution of 1 cm. The buildings are 20 cm high, and are...
Figure 17. Simulated and experimental results for the laboratory scale tsunami case.

represented as topographical features on the domain. All boundaries are closed, except for the free outflow at the outlet. The domain was discretised with a \( \delta x = 1 \) cm resolution, resulting in 54600 cells. The domain was forced by two constant pulses of rain of 85 mm h\(^{-1}\) and 300 mm h\(^{-1}\) (lowest and highest intensities in the experiments) with a duration of 60 s and 20 s. The simulation was run up to \( t = 200 \) s. Friction was modelled by Manning’s equation, with a constant roughness coefficient of 0.010 m\(^{-1/3}\) for steel (Cea et al., 2010a).

Figure 18 shows the experimental and simulated outflow discharge for both rainfall pulses. There is a very good qualitative behaviour, and peak flow is quantitatively well reproduced by the simulations. For the 300 mm h\(^{-1}\) intensity rainfall, the onset of runoff is earlier than in the experiments, and overall the hydrograph is shifted towards earlier times. Cea et al. (2010a) observed a similar behaviour, and pointed out that this is likely caused by surface tension during the early wetting of the surface, and it was most noticeable on the experiments with higher rainfall intensity.

Figure 18. Simulated hydrographs compared to experimental data from Cea et al. (2010a) for two rainfall pulses on the A12 building arrangement.
5.7 Experimental rainfall-runoff over a dense idealised urban area

Cea et al. (2010b) presented a laboratory scale experiment in a flume with a dense idealised urban area. The case elaborates on the setup of Cea et al. (2010a) (subsection 5.6), including 180 buildings (case L180), in contrast to the 12 buildings in subsection 5.6. This consequently requires a higher resolution to resolve the building (6.2 cm sides) and street width (~2 cm), and the flow in the streets. Rainfall is a single pulse of constant intensity. Two setups were used with intensities 85 mm h\(^{-1}\) and 300 mm h\(^{-1}\) and durations of 60 s and 20 s respectively. As Figure 19 shows, the hydrographs are well captured by the simulation, albeit with a delay. Analogously to subsection 5.6, this can be attributed to surface tension in the early wetting phase.

![Figure 19](image)

(a) Intensity 85 mm h\(^{-1}\), duration 60 s  
(b) Intensity 300 mm h\(^{-1}\), duration 20 s

Figure 19. Simulated hydrographs compared to experimental data from Cea et al. (2010b) for two rainfall pulses on the L180 building arrangement.

6 Plot-scale to catchment-scale experiments

6.1 Malpasset dam-break

The Malpasset dam-break event (Hervouet and Petitjean, 1999) is the most commonly used real-scale benchmark test in shallow water modelling (An et al., 2015; Brodtkorb et al., 2012; Bruflau et al., 2004; Caviedes-Voullième et al., 2002b; Duran et al., 2013; George, 2010; Hervouet and Petitjean, 1999; Hou et al., 2013a; Kesserwani and Liang, 2012; Kesserwani and Sharifian, 2020; Kim et al., 2014; Liang et al., 2007; Saëtra et al., 2015; Schwanenberg and Harms, 2004; Smith and Liang, 2013; Valiani et al., 2002; Xia et al., 2011; Yu and Duan, 2012; Wang et al., 2011; Zhou et al., 2013; Zhao et al., 2019). Although it may not be particularly challenging for current solvers, it remains an interesting case due to its scale, and the available field and experimental data (Aureli et al., 2021). The computational domain was discretised to \(\Delta x = 25 m\) and \(\Delta y = 10 m\) (resulting in
83137 and 515262 cells respectively). The Glaucker-Manning coefficient was set to a uniform value of $0.033 \text{sm}^{-1/3}$, which has been shown to be a good approximation in the literature.

Figure 20. Result comparison for the Malpasset dam-break test case.
6.2 Plot-scale field rainfall-runoff experiment

Tatard et al. (2008) presented a rainfall-runoff plot-scale experiment performed in Thies, Senegal. This test has been often used for benchmarking of rainfall-runoff models (Caviedes-Voullième et al., 2020a; Chang et al., 2016; Mügler et al., 2011; Özgen-Xian et al., 2020; Park et al., 2019; Simons et al., 2014; Yu and Duan, 2017; Weill, 2007). The domain is a field plot of $10 \times 4$ m, with an average slope of $1\%$. A rainfall simulation with an intensity of $70 \text{mmh}^{-1}$ during $180\text{s}$ was performed. Steady velocity measurements were taken at 62 locations. The Glaucker-Manning roughness coefficient was set to $0.02\text{sm}^{-1/3}$ and a constant infiltration rate was set to $0.0041667\text{mms}^{-1}$ (Mügler et al., 2011). The domain was discretised with $\delta x = 0.02666\text{m}$, resulting in 56250 cells, with a single free outflow boundary downslope.

Simulated velocities are compared to experimental velocities at the 62 gauged locations in Figure 21. A good agreement of simulated and experimental velocities exists, especially in the lower velocity range. The agreement is similar to previously reported results (e.g., Caviedes-Voullième et al., 2020a), and the differences between simulated and observed velocities have been shown to be a limitation of a depth-independent roughness and Manning’s model (Mügler et al., 2011).

![Figure 21. Comparison of simulated (line) and experimental (circles) steady velocities in the Thies field case.](https://doi.org/10.5194/gmd-2022-208)

7 Performance and scaling

In this section we report a first investigation of the computational performance and parallel scaling of SERGHEI-SWE for selected test cases. To demonstrate performance-portability, we show performance metrics for both OpenMP and CUDA backends enabled by Kokkos computed on CPU and GPU architectures respectively. For that, hybrid MPI-OpenMP and MPI-CUDA implementations are used, with one MPI task per node for MPI-OpenMP and one MPI task per GPU for MPI-CUDA. Most of the runs were performed on JUWELS at JSC (Jülich Supercomputing Centre). Additional HPC systems were also used.
for some cases. Properties of all systems are shown in Table 4. Additionally, we provide performance metrics on non-HPC systems including some consumer-grade GPUs.

It is important to highlight that no performance tuning or optimisation has been carried out for these tests, and that no system-specific porting efforts were done. All runs relied entirely on Kokkos for portability. The code was simply compiled with the available software stacks in the HPC systems and executed. All results reported here were computed using double precision arithmetic.

Table 4. HPC systems in which SERGHEI has been tested

<table>
<thead>
<tr>
<th>Name</th>
<th>Centre</th>
<th>Institution</th>
<th>Country</th>
<th>Devices</th>
<th>Vendor</th>
<th>Device/node</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUWELS</td>
<td>JSC</td>
<td>FZJ</td>
<td>Germany</td>
<td>Xeon Platinum 8168 CPU</td>
<td>Intel</td>
<td>2x(2x24)</td>
<td>2567</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Volta V100 GPU</td>
<td>Nvidia</td>
<td>4</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ampere A100 GPU</td>
<td>Nvidia</td>
<td>4</td>
<td>936</td>
</tr>
<tr>
<td>JURECA-DC</td>
<td>JSC</td>
<td>FZJ</td>
<td>Germany</td>
<td>EPYC 7742 2.25 GHz</td>
<td>AMD</td>
<td>2x(2x64)</td>
<td>480</td>
</tr>
<tr>
<td>SUMMIT</td>
<td>OLCF</td>
<td>ORNL</td>
<td>USA</td>
<td>Volta V100 GPU</td>
<td>Nvidia</td>
<td>6</td>
<td>4608</td>
</tr>
<tr>
<td>Cori</td>
<td>NERSC</td>
<td>LBNL</td>
<td>USA</td>
<td>Xeon E5-2698 v3 CPU</td>
<td>Intel</td>
<td>32</td>
<td>2388</td>
</tr>
</tbody>
</table>

JSC: Jülich Supercomputing Centre; FZJ: Forschungszentrum Jülich
OLCF: Oak Ridge Leadership Computing Facility; ORNL: Oak Ridge National Laboratory
NERSC: National Energy Research Scientific Computing Center; LBNL: Lawrence Berkeley National Laboratory

7.1 Single node scaling – Malpasset dam-break

The commonly used Malpasset dam-break test (introduced in subsection 6.1) was also tested for computational performance at a resolution of $\delta x = 10$ m. Results are shown in Figure 22. The case was computed on CPUs a single JUWELS node and a single JURECA-DC node. Three additional runs with single Nvidia GPUs were carried out: a commercial-grade GeForce RTX 3070, 8GB GPU (in a desktop computer) and two scientific-grade cards V100 and A100 respectively (in JUWELS). As Figure 22 shows, CPU runtime quickly approaches an asymptotic behaviour (therefore demonstrating that additional nodes are not useful in this case). Notably, all three GPUs outperform a single CPU node, and the performance gradient among the GPUs is evident. The A100 GPU is roughly 6.5 faster than a full JUWELS CPU node, and even for the consumer-grade RTX 3070 the speed-up compared to a single HPC node is 2.2. Although it is possible to scale up this case with significantly higher resolution and test it with multiple GPUs, it is not a case well suited for such a scaling test. Multiple GPUs (as well as multiple nodes with either CPUs or GPUs) require a domain decomposition. The orientation of the Malpasset domain is roughly NW-SE, which makes both 1D decompositions (along x or y) and 2D decompositions (x and y) inefficient, as many regions have no computational load. Moreover, the dam-break nature of the case implies that a large part of the valley is dry for long periods of time, therefore load balancing among the different nodes/GPUs will be poor.
Figure 22. Scaling for the Malpasset case (δx = 10 m) on a single node and on single GPUs. GPU speed-ups relative to a full JUWELS node are 6.5 (A100), 3.4 (V100) and 2.2 (RTX 3070).

7.2 HPC scaling – 2D circular dam break case

This is a simple analytical verification test in the shallow water literature, which generalises the 1D dam-break solution. This is a convenient test for scaling studies, as resolution can be increased at will, and the square and fully wet domain minimises load balancing issues. We take a 400 × 400 m flat domain with center at (0, 0) and initial conditions given by

\[ h(x, y, t = 0) = \begin{cases} 
4 & \text{if } \sqrt{x^2 + y^2} \leq 50 \\
1 & \text{otherwise}
\end{cases} \] (12)

We generated three computational grids, with \( \delta x = 0.05, 0.025, 0.0175 \text{ m} \), which correspond to 64, 256 and 552 million cells respectively. Figure 23 shows the strong scaling results for the 64 and 256 million cells cases, computed in the JUWELS-Booster system, on A100 Nvidia GPUs. The 64 million does not scale well beyond 4 GPUs. However, the 256 cells problem scales well up to 64 GPUs (and shows inefficiencies with 128), showing that the first case simply is too small for significant gains.

For the 552 million cell grid, only two runs were computed with 128 and 160 GPUs (corresponding to 32 and 40 nodes in JUWELS-Booster respectively). Runtime for these was 95.4 and 84.7 s respectively, implying a very good 89% scaling efficiency for this large number of GPUs. For this problem and these resources, the time required for inter-GPU communications is comparable to that used by kernels computing fluxes and updating cells, signalling scalability limits for this case on the current implementation.

7.3 HPC-scaling of rainfall-runoff in a large catchment

To demonstrate scaling under production conditions of real scenarios, we use an idealised rainfall-runoff simulation over the Lower Triangle region in the East River Watershed (Colorado, USA) (Carroll et al., 2018; Hubbard et al., 2018; Özgen-Xian...
Figure 23. Strong scaling behaviour for a circular dam break test case

For practical purposes, two configurations have been used for this test. A short rainfall of $T = 870\,s$, which was computed in NERSC Cori and JUWELS to assess CPU performance and scalability (results shown in Figure 24). A long rainfall event lasting $T = 12000\,s$ was simulated in SUMMIT and JUWELS to assess GPU performance and scalability, with results shown in Figure 25. CPU results (Figure 24) show that the strong scaling behaviour in NERSC Cori and JUWELS is very similar. Absolute runtimes are longer for NERSC Cori since the scaling study was carried starting from a single core, whereas in JUWELS it was with a full node (i.e., 48 cores). Most importantly, the GPU strong scaling behaviour overlaps almost completely between JUWELS and SUMMIT, though computations in SUMMIT were somewhat faster. CPU and GPU scaling is clearly highly efficient and with similar behaviour. These results demonstrate the performance-portability delivered via Kokkos to SERGHEI.

8 Vision and future work

Similar to Giardino and Houser (2015), we view water fluxes as the thread connecting various elements of our Earth’s system. Thus, SERGHEI is envisioned as a modular simulation framework around a physically-based hydrodynamic core, which allows to represent a variety of water-driven and water-limited processes in a flexible manner. As illustrated by the conceptual framework in Figure 26, SERGHEI’s hydrodynamic core will consist of mechanistic surface and subsurface flow solvers (light and dark blue), around which a generalised transport framework for multi-species transport and reaction will be implemented (gray). The transport framework will further enable the implementation of morphodynamics (gold) and vegetation dynamics (green) models. The transport framework will also include a Lagrangian particle-tracking module (currently also under development). At the time of the writing of this paper, the subsurface flow solver—based on the three-dimensional extension of the Richards solver by Li et al. (2021)—is experimentally operative and is underway to be coupled to the surface flow solver, thus,
The initial infrastructure for the transport-based three other frameworks is currently under development.

In contrast to many established codes, SERGHEI is conceptualised and designed with extendibility and software interoperability in mind. Design choices have been made to include foreseeable future developments on a wide range of topics: (i) numerics, e.g., the Discontinuous Galerkin discretisation strategies (Caviedes-Voullième and Kesserwani, 2015; Shaw et al., 2020) and multiresolution adaptive meshing (Caviedes-Voullième et al., 2020b; Kesserwani and Sharifian, 2020; Özgen-Xian, 2020; Kesserwani and Sharifian, 2020; Özgen-Xian, 2020).
et al., 2020); (ii) interfaces to mature geochemistry engines, e.g., CrunchFlow (Steefel, 2009) and PFLOTRAN (Lichtner et al., 2015), and (iii) vegetation models with varying degree of complexity, for example, Ecosys, EcH2O.

505 In the long term, SERGHEI’s Kokkos-based HPC capabilities will enable, for example, to run decadal morphological simulations, to better capture sediment connectivity across the landscape, and to run catchment-scale hydro-biogeochemical simulations with unprecedented high spatial resolution.

9 Conclusions

In this paper we present the SERGHEI framework, and in particular the SERGHEI-SWE module. SERGHEI-SWE implements a 2D fully dynamic shallow water solver, harnessing state-of-the-art numerics, and leveraging on Kokkos to facilitate portability across architectures. We show through empirical evidence with a large set of well established benchmarks that SERGHEI-SWE is accurate, numerically stable, and robust. Importantly, we show that SERGHEI-SWE’s parallel scaling is very good for CPU-based HPC systems, consumer-grade GPUs, and GPU-based HPC systems. Consequently, we claim that SERGHEI is indeed performance-portable, and approaching exascale-readiness, enabling its use as part of broader Earth System modelling efforts and a plausible community code for shallow water modelling.

Code and data availability. SERGHEI is available through GitLab, at https://gitlab.com/serghei-model/serghei, under a 3-clause BSD license. SERGHEI v1.0 was tagged as the first release at the time of submission of this paper. A static version of SERGHEI v1.0 is archived in Zenodo, DOI: 10.5281/zenodo.7041423

A repository containing test cases is available https://gitlab.com/serghei-model/serghei_testcases. This repository contains many of the cases reported here, except those for which we cannot publicly release data, but which can be obtained from the original authors of the
datasets. A static version of this datasets is archived in Zenodo, with DOI: 10.5281/zenodo.7041392. Additional convenient pre- and post-processing tools are also available at https://gitlab.com/serghei-model/sergheir.

Author contributions. DCV: Conceptualisation, Investigation, Software, Validation, Visualisation, Writing. MMH: Conceptualisation, Methodology, Software, Formal analysis, Writing. MN: Software. IOX: Formal Analysis, Software, Validation, Visualisation, Writing.

Competing interests. The authors declare no conflicts of interests.

Acknowledgements. The authors gratefully acknowledge the Earth System Modelling Project (ESM) for supporting this work by providing computing time on the ESM partition of the JUWELS supercomputer at the Jülich Supercomputing Centre (JSC) through the compute time project Runoff Generation and Surface Hydrodynamics across Scales with the SERGHEI model (RUGSHAS), Project Number 22686. This work used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy, Office of Science, user facility operated under Contract Number DE-AC02-05CH11231. This research was also supported by the US Air Force Numerical Weather Modeling Program and used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is a US Department of Energy (DOE) Office of Science User Facility. Some of the co-authors are employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy. Accordingly, the US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript or allow others to do so, for US Government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (https://www.energy.gov/downloads/doe-public-access-plan).
References


