SERGHEI (SERGHEI-SWE) v1.0: a performance-portable high-performance parallel-computing 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, Jülich, Germany
$^2$Institute of Bio- and Geosciences: Agrosphere (IBG-3), Forschungszentrum Jülich, Jülich, Germany
$^3$Fluid Mechanics, I3A, Universidad de Zaragoza, Zaragoza, Spain
$^4$Oak Ridge National Laboratory, Oak Ridge, USA
$^5$Institute of Geoecology, Technische Universität Braunschweig, Braunschweig, Germany
$^6$Earth & Environmental Sciences Area, Lawrence Berkeley National Laboratory, Berkeley, USA

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

Received: 22 August 2022 – Discussion started: 8 September 2022
Revised: 9 December 2022 – Accepted: 30 December 2022 – Published: 8 February 2023

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. At the core of SERGHEI’s innovation is its performance-portable high-performance parallel-computing (HPC) implementation, built from scratch on the Kokkos portability layer, allowing SERGHEI to be deployed, in a performance-portable fashion, in graphics processing unit (GPU)-based heterogeneous systems. 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) and its HPC implementation. This module is designed to be applicable to hydrological and environmental problems including flooding and runoff generation, with an outlook towards Earth system modelling. Its applicability is demonstrated by testing several well-known benchmarks and large-scale problems, for which SERGHEI-SWE achieves excellent results for the different types of shallow-water problems. Finally, SERGHEI-SWE scalability and performance portability is demonstrated and evaluated on several TOP500 HPC systems, with very good scaling in the range of over 20 000 CPUs and up to 256 state-of-the art GPUs.

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 with 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
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 (2018); 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-graphics processing unit (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, HIP, SYCL, 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 it is 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 (SWEs) solver for free-surface hydrodynamics at the heart of SERGHEI.

SERGHEI-SWE enables the simulation of surface hydrodynamics of overland flow and streamflow 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 with implicit solvers, as usually required by diffusive equations (Caviedes-Voullième et al., 2018; Fernández-Pato and García-Navarro, 2016) – have been pushing to replace simplified surface flow models (for hydrological purposes) 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 was 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 that respond to them, improved spatiotemporal rainfall data and simulations, and 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 they are 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
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. (2021)</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>K 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>Moullec et al. (2011)</td>
<td>–</td>
<td>Yes</td>
<td>Open-source (GPLv3/LGPL)</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>

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 are therefore somewhat limited to Nvidia hardware. This leads into the second and most relevant 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 of having a single code base for the currently fully operational OpenMP and CUDA backends, as well as HIP, which is currently experimental in SERGHEI but, most importantly, keeps this code base relevant for other backends, such as SYCL. This is particularly important, as the current HPC landscape features not only Nvidia GPUs but also a currently increased adoption of AMD GPUs, with the most recent leading TOP 500 systems – Frontier and LUMI – as well as upcoming systems (e.g. El Capitan) relying on AMD GPUs. In this way, SERGHEI is safely avoiding the vendor lock trap.

SERGHEI-SWE has been developed by harnessing the past 15 years’ worth of numerical advances in the solution of SWE, ranging from fundamental numerical formulations (Echeverribar 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). Most of this work was done in the context of developing solvers for flood modelling, with rather engineering-oriented applications, demanding high quantitative accuracy and predictive capability. Most of the established models in Table 1 were developed within such contexts, although many are currently also adopted for more hydrological applications. Leveraging on this technology, SERGHEI-SWE is designed to cope with the classical shallow-water applications of fluvial and urban flooding, as well as with the emerging rainfall runoff problems in both natural and urban environments (for which coupling to sewer system models is a longer-term objective) and with other flows of broad hydrological and environmental interest that occur on (eco)hydrological timescales, priming it for further uses in ecohydrology and geomorphology. Nevertheless, all shallow-water applications should benefit from the high performance and high scalability of SERGHEI-SWE. With an HPC-ready SWE solver, catchment-scale rainfall runoff applications around the 1 m² resolution are feasible. Similarly, large river and floodplain simulations can be enabled for operational flood forecasting, and flash floods in urban environments can be tackled with extremely high spatial resolution. Moreover, it is noteworthy that SERGHEI-SWE is not confined to HPC environments, and users with workstations can also benefit from improved performance.

1.1 The SERGHEI framework

SERGHEI is envisioned as a modular simulation framework around a physically based hydrodynamic core, which allows a variety of water-driven and water-limited processes to be represented in a flexible manner. In this sense, SERGHEI is based on the idea of water fluxes as a connecting thread among various components and processes within the Earth system (Giardino and Houser, 2015). As illustrated by the conceptual framework in Fig. 1, SERGHEI’s hydrodynamic
core will consist of a mechanistic surface (SERGHEI-SWE, the focus of this paper) and subsurface flow solvers (light and dark blue), around which a generalised transport framework for multi-species transport and reaction will be implemented (grey). 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 being coupled to the surface flow solver, thus making the hydrodynamic core of SERGHEI applicable to integrated surface–subsurface hydrology. The initial infrastructure for the three other transport-based frameworks is currently under development.

2 Mathematical and numerical model of SERGHEI-SWE

In this section we provide an overview of the underlying mathematical model and the numerical schemes implemented in SERGHEI-SWE. The implementation is based on well-established numerical schemes, and consequently, we limit this to a minimal presentation.

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

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

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 \( q_x = hu \) \([L^2T^{-1}]\) and \( q_y = hv \) \([L^2T^{-1}]\), respectively. \( F \) and \( G \) are the fluxes of these conserved variables with gravitational acceleration \( g \) \([LT^{-2}]\). The mass source terms \( S_t \) account for rainfall, \( \tau_o \) \([LT^{-1}]\), and infiltration or exfiltration, \( \tau_t \) \([LT^{-1}]\). 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). In addition, specialised formulations of the friction slope exist to consider the effect of microtopography and vegetation for small water depths, e.g. variable Manning’s coefficients (Jain and Kothiyari, 2004; Mügler et al., 2011) or generalised friction laws (Özgen et al., 2015b). A recent systematic comparison and in-depth discussion of several friction models with a focus on rainfall runoff simulations is given in Crompton et al. (2020). Implementing additional friction models is of course possible – and relevant, especially to address the multiscale nature of runoff in catchments – but not essential to the points in this paper. The observant reader will note that in Eq. (1), viscous and turbulent fluxes have been neglected. The focus here is on applications (rainfall runoff, dam breaks) where the influence of these can be safely neglected. Turbulent viscosity may become significant for ecohydraulic simulations of river flow, and turbulent fluxes of course play an important role in mixing in transport simulations. We will address these issues in future implementations of the transport solvers in SERGHEI.

SERGHEI-SWE uses a first-order accurate upwind finite-volume scheme with a forward Euler time integration to solve the system of Eq. (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 envi-
ronmental 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 as follows:

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

followed by

$$U_i^{n+1} = U_i^n + (r_o - r_l)^n \Delta t,$$

where $\lambda$ and $\tilde{\beta}$ 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–Friedrichs–Lewy (CFL) condition:

$$\Delta t = \text{CFL}_{\text{min}} \left\{ \frac{\Delta x}{\max|q_x h_i| + \sqrt{gh_i}|q_y h_i| + \sqrt{gh_i}} \right\}$$

where $CFL \leq 0.5$.

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.

It is relevant to acknowledge that second (and higher)-order schemes for SWE are available (e.g. Buttinger-Kreuzhuber et al., 2019; Caviedes-Voullième et al., 2020b; Hou et al., 2015; Navas-Montilla and Murillo, 2018). However, first-order schemes are still a pragmatic choice (Ayog et al., 2021), especially when dealing with very high resolutions (as targeted with SERGHEI), which offsets their higher discretisation error and numerical diffusivity in comparison to higher-order schemes. Similarly, robust schemes for unstructured triangular meshes are well established together with their well-known advantages in reducing cell counts and numerical diffusion (Bomers et al., 2019; Caviedes-Voullième et al., 2012, 2020a). As these advantages are less relevant at very high resolutions, we opt for Cartesian grids to avoid issues with memory mapping, coalescence and cache misses in GPUs (Lacasta et al., 2014), and additional memory footprints while also making domain decomposition simpler. Both higher-order schemes and unstructured (and adaptive) meshes may also be implemented within SERGHEI.

3 HPC implementation of the SERGHEI framework

In this section we describe the key ingredients of the HPC implementation of SERGHEI. Conceptually, this requires, firstly, handling parallelism inside a computational device (multicore CPU or GPU) with shared memory and the related portability and corresponding backends (i.e. OpenMP, CUDA, HIP, etc.). On a higher level of parallelism, distributing computations across many devices requires domain decomposition and a distributed memory problem, implemented via MPI. The complete implementation of SERGHEI encompasses both, distributing parallel computations into many subdomains, each of which is mapped onto a computational device. Here we start the discussion from the higher level of domain decomposition and highlight that the novelty of SERGHEI lies with the multiple levels of parallelism together with the performance-portable shared-memory approach via Kokkos.

3.1 Domain decomposition

The surface domain is a two-dimensional plane, discretised by a Cartesian grid with a total cell number of $N_x \times N_y$, 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 $n_x = (n_x + 2)(n_y + 2)$. The definitions are sketched – without loss of generality – for a square-shaped subdomain in Fig. 2, and the way these subdomains overlap in the global domain is sketched in Fig. 3 (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 $N_x$, 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 Fig. 2. The second set maps into memory position. For example, in order to access the physical cell 14 in Fig. 2, one has to access memory position 27.
3.2 Data exchange between subdomains

The underlying methods for data exchange between subdomains are centred on the subdomains rather than on the interfaces. Data are exchanged through MPI-based send-and-receive calls (non-blocking) that aggregate data in the halo cells across the 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 3 (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

Intra-device 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 \texttt{realArr}, based on \texttt{View}. This takes the form of Listing 1 for the shared (host) memory.

---

Figure 2. Domain decomposition and indexing in SERGHEI: a subdomain consists of physical cells (white) and halo cells (grey). SERGHEI uses two sets of indices: an index for physical cells (a) and an index for all cells including the halo cells (b).

Figure 3. Data exchange between subdomains in SERGHEI: in the global surface domain, subdomains overlap with each other through their halo cells (a). These halo cells are used to exchange data between the subdomains (b).
space and Listing 2 for the unified virtual memory (UVM) GPU device CUDA memory space. The UVM significantly facilitates development while avoiding writing explicit host-to-device (and vice versa) memory movements.

For a CUDA backend, the use of unified memory (CudaUVMSpace) is shown in Listing 2.

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 Eq. (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.

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 (Eq. 2). Although numerical fluxes can be computed in a cell-centred 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.

**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 set-ups. 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 computational cells. A summary of \( L \) norms for all test cases is given in Table 2. The definition of the \( L \) norms is given in Appendix A.

**4.1.1 C property**

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). Figure 4 shows that SERGHEI-SWE satisfies the C property by preserving a lake at rest in the presence of an emerged bump (an immersed bump test is shown in Sect. A1) and matches the analytical solution provided by SWASHES.

**4.1.2 Well-balancing**

To show well-balancing under steady flow, we computed two transcritical flows based on the analytical bench-

<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>Fig. 4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Fig. 5 (left)</td>
<td>0.68584</td>
<td>0.01909</td>
<td>0.0015</td>
</tr>
<tr>
<td>Fig. 5 (right)</td>
<td>1.02096</td>
<td>0.06826</td>
<td>0.0622</td>
</tr>
</tbody>
</table>

https://doi.org/10.5194/gmd-16-977-2023
mark of a one-dimensional flume with varying geometry proposed by MacDonald et al. (1995). 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). Additional well-balancing tests can be found in Sect. A2. 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. This is particularly important to subcritical fluvial flows and rainfall runoff problems, since both are usually dominated by these two terms.

Figure 5 shows comparisons between SERGHEI-SWE and analytical solutions (obtained through SWASHES) for two transcritical steady flows. Very good agreement 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 discretisation used in the SERGHEI-SWE solver. As reported by Burguete et al. (2008) and Murillo et al. (2009), a centred friction discretisation 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).

4.2 Analytical dam break

We verify SERGHEI-SWE’s capability to capture transient flow based on analytical dam breaks (Delestre et al., 2013). Dam break problems are defined by an initial discontinuity in the water depth in the domain \( h(x) \), such that

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

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 \( x_0 \). The domain is 10 m long, the discontinuity is located at \( x_0 = 5 \text{ m} \), and the total run time is 6 s. 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 100 000 elements, respectively.

A classical frictionless dam break over a wet bed is reported in Sect. A3. Here we focus on a frictionless dam break over a dry bed. Flow featuring depth close to dry bed is a special case for the numerical solver because regular wave speed estimations become invalid Toro (2001). Initial conditions are set as \( h_L = 0.005 \text{ m} \) and \( h_R = 0 \text{ m} \). Model results are plotted against the analytical solution by Ritter for different grid resolutions in Fig. 6. 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. Note that the norms definition can be found in Sect. A2. 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 oscillation: parabolic bowl

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 \( \Delta x = 0.01 \text{ m} \), one reported here and the other in Sect. A4.

The well-established test case by Thacker (1981) for a periodic oscillation of a planar surface in a frictionless paraboloid has been extensively used for validation of shallow-water solvers (e.g. Aureli et al., 2008; Dazzi et al., 2018; Liang et al., 2015; Murillo and García-Navarro, 2010; Vacondio et al., 2014; Zhao et al., 2019) because of its rather complex 2D nature and the presence of moving wet–dry fronts. The topography is defined as

\[
 z(r) = -h_0 \left( 1 - \frac{r^2}{a^2} \right),
\]

\[
 r = \sqrt{(x - L/2)^2 + (y - L/2)^2},
\]

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 \( h_0 = 0.1 \) m, \( a = 1 \) m, and \( L = 4 \) m. The simulation is run for three periods \( (T = 2.242851 \) s), with a spatial resolution of \( \delta x = 0.01 \) m. The analytical solution can be found in Thacker (1981); Delestre et al. (2013).

Snapshots of the simulation are plotted in Fig. 7 and compared to the analytical solution. The model results agree well with the analytical solution after three periods, with slight growing-phase error, as is commonly observed on this test case.

4.4 Variable rainfall over a sloping plane

Govindaraju et al. (1990) presented an analytical solution to a time-dependent 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 B from Govindaraju et al. (1990), a piecewise constant rainfall with two periods of alternating low and high intensities (50.8 and 101.6 mm h\(^{-1}\)) up until 2400 s. Friction is modelled with Chezy’s equation, with a roughness coefficient of 1.767 m\(^{1/2}\) s\(^{-1}\). The computational domain was defined by a 200 \times 10 grid, with \( \delta x = 0.109725 \) m.

The simulated discharge hydrograph at the outlet is compared against the analytical solution in Fig. 8. The numerical solutions matches 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.

5 Laboratory-scale experiments

5.1 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 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 centreline, 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 106 887 cells. Manning’s roughness was set to 0.01 s 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\(^{-1}\). Steady water surface results in the obstacle region are shown in Fig. 9 for a centreline profile \((y = 0)\) and a cross-section at the rectangular obstacle, specifically at \( x = 2.40 \) m (the coordinate system is set at the centre 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 Fig. 10. Again, the simulations approximate experiments well, capturing both the overall behaviour of the water depths and the
5.2 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 and top diameter 2.2 m and with a height of 0.625 m was placed at the centre of a 26 m × 27.6 m smooth and flat domain. An initial hydrostatic water level of $h_0 = 0.32$ m was set, and a wave was imposed on the boundary following

$$h_b = h_0 + A \operatorname{sech}^2 \left( \frac{B(t - T)}{C} \right),$$

(7)

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

(8)

$$C = h_0 \sqrt{\frac{4h_0 B}{3A^2 gh_0}},$$

(9)

where $A = 0.032$ m is the wave amplitude, and $T = 2.84$ s is the time at which the peak of the wave enters the domain. Figure 11 shows results for a simulation with a 2.5 cm resolution, resulting in 1.2 million cells. A roughness coefficient of 0.013 s m$^{-1/3}$ was used for the concrete surface. The results are comparable to previous solutions in the literature, in general reproducing well the water surface, with some delay over experimental measurements.

5.3 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 m × 2.5 m. Here, we select one building arrangement named A12 by Cea et al. (2010a). The original digital elevation model (DEM) is available (from Cea et al., 2010a) at a resolution of 1 cm. The buildings are 20 cm high and are 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 \( \delta x = 1 \text{ cm} \) resolution, resulting in 54,600 cells. The domain was forced by two constant pulses of rain of 85 and 300 mm h\(^{-1}\) (lowest and highest intensities in the experiments) with durations of 60 and 20 s. The simulation was run up to \( t = 200 \text{ s} \). 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 12 shows the experimental and simulated outflow discharge for both rainfall pulses. There is a very good qualitative agreement, 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 7. Planar surface in a paraboloid: snapshots of water depth by the model compared to the analytical solution (contour lines). Period $T = 2.242851$ s.

Figure 8. Simulated and analytical discharge for the analytical case of rainfall in a flume.

6 Plot-scale to catchment-scale experiments

6.1 Plot-scale field rainfall runoff experiment

Tatard et al. (2008) presented a rainfall runoff plot-scale experiment performed in Thiès, Senegal. This test has been used often 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 0.01. A rainfall simulation with an intensity of 70 mm h$^{-1}$ during 180 s was performed. Steady velocity measurements were taken at 62 locations. The Gauckler–Manning roughness coefficient was set to $0.02 \text{ s m}^{-1/3}$, and a constant infiltration rate was set to $0.0041667 \text{ mm s}^{-1}$ (Mügler et al., 2011). The domain was discretised with $\Delta x = 0.02666$ m, resulting in 56,250 cells, with a single free-outflow boundary downslope.

Simulated velocities are compared to experimental velocities at the 62 gauged locations in Fig. 13. A good agreement between 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).

6.2 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; Brufau et al., 2004; Caviedes-Voullième et al., Geosci. Model Dev., 16, 977–1008, 2023 https://doi.org/10.5194/gmd-16-977-2023
Figure 9. Simulated and experimental steady water surface in the obstacle region of the G3 flume for the centreline profile \( y = 0 \) m (a) and a cross-section \( x = 2.40 \) m (b).

Figure 10. Simulated and experimental transient water depths at three gauge points (a) \( x = 2.25 \) m, \( y = 0 \) m; (b) \( x = 2.40 \) m, \( y = 0.08 \) m; (c) \( x = 2.60 \) m, \( y = 0 \) m) for the G3 flume dam break over several obstacles.

2020b; 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; Setra 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). 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 \( \delta x = 10 \) m (resulting in 83 137 and 515 262 cells, respectively). The Gauckler–Manning coefficient was set to a uniform value of 0.033 s m\(^{-1/3}\)), which has been shown to be a good approximation in the literature. Figure 14 shows a comparison of simulated water surface elevation (WSE) and arrival time for two resolutions against the reference experimental and field data. Figure 15 shows the geospatial distribution of the relative WSE error and the ratio of the simulated arrival time to the observed time. Overall, WSE shows a good agreement and somewhat smaller scatter for the higher resolution. Arrival time tends to be overestimated, somewhat more for coarser resolutions.

7 Performance and scaling

In this section we report an 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.
990 D. Caviedes-Voullièmement et al.: SERGHEI-SWE v1.0

Figure 11. Simulated and experimental results of unsteady flow over an island for gauges G9 (a), G16 (b), and G22 (c).

Figure 12. Simulated hydrographs compared to experimental data from Cea et al. (2010a) for two rainfall pulses on the A12 building arrangement. (a) Rain intensity 85 mm h^{-1}, duration 60 s. (b) Rain intensity 300 mm h^{-1}, duration 20 s.

Figure 13. Comparison of simulated (line) and experimental (circles) steady velocities in the Thiès field case.

7.1 Single-node scaling – Malpasset dam break

The commonly used Malpasset dam break test (introduced in Sect. 6.2) was also tested for computational performance at a resolution of \( \delta x = 10 \) m. Results are shown in Fig. 16. 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, 8 GB GPU (in a desktop computer), and two scientific-grade cards V100 and A100, respectively (in JUWELS). As Fig. 16 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 to 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 and/or GPUs will be poor.

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 solu-
tion. We purposely select this case (instead of one of the many verification problems) for its convenience for scaling studies. Firstly, resolution can be increased at will. Additionally, the square domain allows for trivial domain decomposition, which together with the fully wet domain and the radially symmetric flow field minimises load-balancing issues. Essentially, it allows for a very clean scalability test with minimal interference from the problem topology, which facilitates scalability and performance analysis (in contrast to the limitations of the Malpasset domain discussed in Sect. 7.1). We take a 400 m × 400 m flat domain with the centre at (0, 0) and initial conditions given by

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

(10)

We generated three computational grids, with \( \delta x = 0.05, 0.025, 0.0175 \) m, which correspond to 64, 256, and 552 million cells, respectively. Figure 17 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-million-cells problem scales well up to 64 GPUs (and

https://doi.org/10.5194/gmd-16-977-2023 Geosci. Model Dev., 16, 977–1008, 2023
Table 4. HPC systems in which SERGHEI-SWE 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 or 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>2 × (2 × 24)</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>2 × (2 × 64)</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>


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 et al., 2020). The domain has an area of 14.82 km² and elevations ranging from 2759–3787 m. The computational problem is defined with a resolution of \( \delta x = 0.5 \) m (matching the highest-resolution DEM available), resulting in \( 122 \times 10^6 \) computational cells. Although this is not a particularly large catchment, the very-high-resolution DEM available makes it an interesting performance benchmark, which is our sole interest in this paper.

For practical purposes, two configurations have been used for this test: a short rainfall of \( T = 870 \) s, which was computed in Cori and JUWELS to assess CPU performance and scalability (results shown in Fig. 18), and a long rainfall event lasting \( T = 12 \times 10^3 \) s, which was simulated in Summit and JUWELS to assess GPU performance and scalability, with results shown in Fig. 19. CPU results (Fig. 18) show that the strong scaling behaviour in Cori and JUWELS is very similar. Absolute runtimes are longer for Cori, since the scaling study was carried out 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, although computations in Summit were somewhat faster. CPU and GPU scaling are clearly highly efficient, with similar behaviour. These results demonstrate the performance portability delivered via Kokkos to SERGHEI.

8 Conclusions and outlook

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. These features make SERGHEI-SWE a plausible community code for...
shallow-water modelling for a wide range of applications requiring large-scale, very-high-resolution simulations.

Exploiting increasingly better and highly resolved geospatial information (DEM, land use, vector data of structures) prompts the need for high-resolution solvers. At the same time, the push towards the study of multiscale systems and integrated management warrants increasingly larger domains. Together, these trends result in larger computational problems, motivating the need for exascale-ready shallow-water solvers. Additionally, HPC technology is evermore available, not only via (inter)national research facilities but also through cloud-computing facilities. It is arguably timely to enable such an HPC-ready solver.

The HPC allows for not only large simulations but also large ensembles of simulations, allowing uncertainty issues to be addressed and enabling scenario analysis for engineering problems, parameter space exploration, and hypothesis testing. Furthermore, although the benefits of high resolution may be marginal for runoff hydrograph estimations, they allow the local dynamics to be better resolved in the domain. Flow paths, transit times, wetting–drying dynamics, and connectivity play important roles in transport and eco-hydrological processes. For these purposes, enabling very-high-resolution simulations will prove to be highly beneficial. We also envision that, provided with sufficient computational resources, SERGHEI-SWE could be used for operational flood forecasting and probabilistic flash-flood modelling. Altogether, this strongly paves the way for the uptake of shallow-water solvers by the broader ESM community and its coupling to Earth system models, as well as their many applications, from process and system understanding to hydrometeorological risk and impact assessment. We also envision that, for users not requiring HPC capabilities, the benefit of SERGHEI-SWE is access to a transparent, open-sourced, performance-portable software that allows workstation GPUs to be exploited efficiently.

As additional SERGHEI modules become operational, the HPC capabilities will further enable simulations that are unfeasible with the current generation of available solvers. For example, with a fully operational transport and morphology module, it will be possible to run decade-long morphological simulations relevant for river management applica-
Figure 19. Runtime (a) and speed up (b) for a strong-scaling experiment with SERGHEI-SWE using GPUs on Summit and JUWELS for the long rainfall event. See Table 4 for details on the systems.

...ations; to better capture sediment connectivity and sediment cascades across the landscape, a relevant topic for erosion and catchment management; or to perform catchment-scale hydro-biogeochemical simulations with unprecedented high spatial resolutions for better understanding of ecohydrological and biogeochemical processes.

Finally, SERGHEI is conceptualised and designed with extendibility and software interoperability in mind, with design choices made to facilitate foreseeable future developments on a wide range of topics, such as

1. numerics, e.g. the discontinuous Galerkin discretisation strategies (Caviedes-Voullième and Kesserwani, 2015; Shaw et al., 2021) and multiresolution adaptive meshing (Caviedes-Voullième et al., 2020b; Kesserwani and Sharifian, 2020, 2022; Özgen-Xian et al., 2020);

2. interfaces to mature geochemistry engines, e.g. CrunchFlow (Steefel, 2009) and PFLOTRAN (Lichtner et al., 2015);

3. vegetation models with varying degrees of complexity, for example, Ecosys (e.g. Grant et al., 2007; Grant and the Ecosys development team, 2022) and EcH2O (Maneta and Silverman, 2013).

Appendix A: Additional validation test cases

This appendix contains an extended set of relevant test cases that are commonly used as validation cases in the literature. It complements and extends the verification evidence in Sect. 4.

A1 C property: immersed bump

Using the same set-up as in Sect. 4.1.1, but with a higher water surface elevation, in Fig. A1 we demonstrate how SERGHEI-SWE also conserves the C property for an immersed bump.

Figure A1. Lake-at-rest solution for an immersed bump.

A2 Well-balancing

To further show that SERGHEI-SWE is well-balanced, we computed three steady flows over a bump. We include a transcritical flow with a shock wave, a fully subcritical flow, and a transcritical flow, as shown in Fig. A2. All of SERGHEI-SWE predictions show excellent agreement with the analytical solution. 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 Garcia-Navarro, 2010).

We also include two additional cases from MacDonald et al. (1995) for fully supercritical and subcritical flows in Fig. A3. These results and their $L$ norms in Table A1 further confirm well-balancing.

Additionally, MacDonald-type solutions can be constructed for frictionless flumes to study the bed slope source...
Figure A2. Analytical steady flows over a bump. SERGHEI-SWE captures moving equilibria solutions for transcritical flow with a shock (top left), fully subcritical flow (top right), and transcritical flow without a shock (bottom).

Figure A3. Analytical steady flows: flumes. SERGHEI-SWE captures moving equilibria solutions for subcritical (a, c) and supercritical (b, d) flow. Note that the solution is stable (no oscillations) and well-balanced (discharge remains constant along the flume).
term implementation in isolation. We present a frictionless test case with SERGHEI-SWE that is not part of the SWASHES benchmark compilation. We discretise the bed elevation of the flume as

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

(A1)

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

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

(A2)

Using \( C_0 = 1.0 \text{ m} \) and \( q_0 = 1.0 \text{ m}^2 \text{s}^{-1} \), we obtain the solution plotted in Fig. A4. SERGHEI-SWE’s prediction and the analytical solution show good agreement.

\[ L^1, L^2, L^\infty \text{ norms for errors in water depth are summarised in Table A1 for the sake of completeness.} \]

### Table A1. 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 are omitted here.

<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>Fig. A1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Fig. A2 (top left)</td>
<td>0.371</td>
<td>0.07285</td>
<td>0.06984</td>
</tr>
<tr>
<td>Fig. A2 (top right)</td>
<td>0.293</td>
<td>0.02618</td>
<td>0.00332</td>
</tr>
<tr>
<td>Fig. A2 (bottom)</td>
<td>0.693</td>
<td>0.0306</td>
<td>0.00356</td>
</tr>
<tr>
<td>Fig. A3 (left)</td>
<td>5.21459</td>
<td>0.12162</td>
<td>0.00435</td>
</tr>
<tr>
<td>Fig. A3 (right)</td>
<td>1.0389</td>
<td>0.03805</td>
<td>0.00191</td>
</tr>
<tr>
<td>Fig. A4</td>
<td>0.74571</td>
<td>0.02743</td>
<td>0.00178</td>
</tr>
</tbody>
</table>

\[ h_1 = 0.005 \text{ m} \text{ and } h_R = 0.001 \text{ m}. \text{ The domain is } 10 \text{ m long, and the discontinuity is located at } x_0 = 5 \text{ m}. \text{ The total run time is } 6 \text{ s}. \text{ Figure A5 shows the model results obtained on successively refined grids compared against the analytical solution by Stoker (1957). Errors for this test case are reported in Table A2. We also report the observed convergence rate } q_R, \text{ calculated on the basis of the } L^1 \text{ norm. As the grid is refined, the model result converges to the analytical solution. Due to the discontinuities in the solution, the observed convergence rate is below the theoretical convergence rate of } R = 1. \]

### A4 Radially symmetrical paraboloid

Using the same computational domain and bed topography as the case in Sect. 4.3, results for the radially symmetrical oscillation in a frictionless paraboloid (Thacker, 1981) are presented here. The details about the initial condition and the analytical solution for the water depth and velocities can be found in Delestre et al. (2013). In particular, the analytical solution at \( t = 0 \text{ s} \) is set as the initial condition, and three pe-
Table A2. Analytical dam break: $L$ norms and empirical convergence rates ($R$) for water depth ($h$) and velocity ($u$).

<table>
<thead>
<tr>
<th>$n$</th>
<th>$L_1(h)$ (m)</th>
<th>$L_2(h)$ (m)</th>
<th>$R(h)$ (m)</th>
<th>$L_1(u)$ ((ms$^{-1}$))</th>
<th>$L_2(u)$ ((ms$^{-1}$))</th>
<th>$R(u)$ ((ms$^{-1}$))</th>
</tr>
</thead>
<tbody>
<tr>
<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>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>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>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>
</tbody>
</table>

Figure A6. Radially symmetrical paraboloid: snapshots of water depth by the model compared to the analytical solution (contour lines). Period $T = 2.242851$ s.

...periods are simulated using $\delta x = 0.01$ m as the grid resolution. Figure A6 shows the numerical and analytical solution at four different times. Although the analytical solution is periodic without dumping, the numerical results show a diffusive behaviour attributed to the numerical diffusion introduced by the first-order scheme. Other than that, model results show good agreement with the analytical solution.

A5 Experimental laboratory-scale tsunami

A 1 : 400-scale experiment of a tsunami run-up over the Monai valley (Japan) was reported by Matsuyama and Tanaka (2001); The third international workshop on long-wave runup 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 $2.05 \times 3.4$ 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, 2012; Kesserwani and Sharifian, 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 was discretised with a resolution of 1.4 cm, producing 95 892 elements. Simulated water surface elevations
Figure A7. Simulated and experimental results for the laboratory-scale tsunami case at gauges G1 (a), G2 (b), and G3 (c).

Figure A8. Simulated (black lines) and experimental (red points) transient water depths at seven gauge points \((x = 17.5, x = 19.5, x = 23.5, x = 25.5, x = 26.5, x = 28.5, x = 35.5)\) from a to b) for the dam break over a triangular sill.

are shown together with the experimental measurements in Fig. A7 at three gauge locations. The results agree well with experimental measurements, both in the water surface elevations and the arrival times of the waves.

### A6 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 Soulaimani, 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 \(\delta x = 0.1\) m resolution. Figure A8 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 to overestimate the waves downstream of the sill (see plot for gauge at \(x = 35.5\) m). Both behaviours are well documented in the literature.
Figure A9. Simulated (lines) and experimental (points) water depth profiles at $y = 0.2$ m, at four times (4, 5, 6 and 10 s, a to d) for the idealised urban dam break case.

Figure A10. Simulated hydrographs compared to experimental data from Cea et al. (2010b) for two rainfall pulses on the L180 building arrangement. (a) Rainfall intensity 180 mm h$^{-1}$, duration 60 s. (b) Rainfall intensity 300 mm h$^{-1}$, duration 20 s.

A7 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 separated by 10 cm. It is widely used in the shallow-water community (Abderrezzak et al., 2008; Caviedes-Voullième et al., 2020b; Ginting, 2019; Hartanto et al., 2011; Jeong et al., 2012; Özgen et al., 2015a; 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 resolutions, both to capture the geometry and to capture the complex flow phenomena which are 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 Fig. A9, together with experimental data. The results agree well with the experimental observations to a similar degree as to what has been reported in the literature.
A8 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 set-up of Cea et al. (2010a) (Sect. 5.3), including 180 buildings (case L180) in contrast to the 12 buildings in Sect. 5.3, which potentially requires a higher resolution to resolve the building (6.2 cm sides) and street width (∼2 cm) and the flow in the streets. We keep a 1 cm resolution. Rainfall is a single pulse of constant intensity. Two set-ups were used with intensities 180 and 300 mm h⁻¹ and durations of 60 and 20 s, respectively. As Fig. A10 shows, the hydrographs are well captured by the simulation, albeit with a delay. Analogously to Sect. 5.3, this can be attributed to surface tension in the early wetting phase.

Appendix B: Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFL</td>
<td>Courant–Friedrichs–Lewy</td>
</tr>
<tr>
<td>Cori</td>
<td>Cori supercomputer at the National Energy Research Scientific Computing Center (USA)</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture, programming interface for Nvidia GPUs</td>
</tr>
<tr>
<td>El Capitan</td>
<td>El Capitan supercomputer at the Lawrence Livermore National Laboratory (USA)</td>
</tr>
<tr>
<td>ESM</td>
<td>Earth system modelling</td>
</tr>
<tr>
<td>Frontier</td>
<td>Frontier supercomputer at the Oak Ridge Leadership Computing Facility (USA)</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics processing unit</td>
</tr>
<tr>
<td>HIP</td>
<td>Heterogeneous Interface for Portability, programming interface for AMD GPUs</td>
</tr>
<tr>
<td>HPC</td>
<td>High-performance computing</td>
</tr>
<tr>
<td>JURECA-DC</td>
<td>Data Centric module of the Jülich Research on Exascale Cluster Architectures supercomputer at the Jülich Supercomputing Centre (Germany)</td>
</tr>
<tr>
<td>JUWELS</td>
<td>Jülich Wizard for European Leadership Science, supercomputer at the Jülich Supercomputing Centre (Germany)</td>
</tr>
<tr>
<td>JUWELS-booster</td>
<td>Booster module of the JUWELS supercomputer (Germany)</td>
</tr>
<tr>
<td>Kokkos</td>
<td>Kokkos, a C++ performance portability layer</td>
</tr>
<tr>
<td>LUMI</td>
<td>LUMI supercomputer at CSC (Finland)</td>
</tr>
<tr>
<td>OpenMP</td>
<td>Open MultiProcessing, shared-memory programming interface for parallel computing</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface for parallel computing</td>
</tr>
<tr>
<td>SERGHEI</td>
<td>Simulation EnviRonment for Geomorphology, Hydrodynamics, and Ecohydrology in Integrated form</td>
</tr>
<tr>
<td>SERGHEI-SWE</td>
<td>SERGHEI’s shallow-water equations solver</td>
</tr>
<tr>
<td>Summit</td>
<td>Summit supercomputer at the Oak Ridge Leadership Computing Facility (USA)</td>
</tr>
<tr>
<td>SWE</td>
<td>Shallow-water equations</td>
</tr>
<tr>
<td>SYCL</td>
<td>A programming model for hardware accelerators</td>
</tr>
<tr>
<td>UVM</td>
<td>Unified Virtual Memory</td>
</tr>
<tr>
<td>WSE</td>
<td>Water surface elevation</td>
</tr>
</tbody>
</table>
Code and data availability. SERGHEI is available through GitLab, at https://gitlab.com/sergei-model/serghei (last access: 6 February 2023), 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: https://doi.org/10.5281/zenodo.7041423 (Caviedes Voullième et al., 2022a).

A repository containing test cases is available at https://gitlab.com/sergei-model/serghei_tests. 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 dataset is archived in Zenodo, with DOI: https://doi.org/10.5281/zenodo.7041392 (Caviedes Voullième et al., 2022b).

Additional convenient pre- and post-processing tools are also available at https://gitlab.com/sergei-model/serghei (last access: 6 February 2023).

Author contributions. DCV contributed to conceptualisation, investigation, software development, model validation, visualisation, and writing. MMH contributed to conceptualisation, methodology design, software development, formal analysis, model validation, and writing. MRN contributed to software development. IÖX contributed to conceptualisation, methodology design, software development, model validation, visualisation, and writing.

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

Disclaimer. Publisher’s note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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 no. 22686. This work used resources of the National Energy Research Scientific Computing Center (NERS), a US Department of Energy, Office of Science, user facility operated under contract no. DE-AC02-05CH11231. This research was also supported by the US Air Force Numerical Weather Modelling programme 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.

Financial support. The article processing charges for this open-access publication were covered by the Forschungszentrum Jülich.

Review statement. This paper was edited by Charles Onuatha and reviewed by Reinhard Hinkelmann and Kenichiro Kobayashi.

References


Jeong, W., Yoon, J.-S., and Cho, Y.-S.: Numerical study on effects of building groups on dam-break flow


Singh, J., Altinakar, M. S., and Ding, Y.: Numerical Modeling of Rainfall-Generated Overland Flow Using Nonlin-


