

Highlights

TRITON: A Multi-GPU Open Source 2D Hydrodynamic Flood Model

M. Morales-Hernández, Md. B. Sharif, A. Kalyanapu, S. K. Ghafoor, T. T. Dullo, S. Gangrade, S.-C. Kao, M. R. Norman, K. J. Evans

- TRITON is released as a multi-GPU open source 2D hydrodynamic flood code
- It solves the 2D shallow water equations using a first order time-explicit scheme
- It runs efficiently for realistic configurations using heterogeneous architectures
- The runoff capability is demonstrated to be convenient for flood modelling
- Communication and I/O times may represent a bottleneck for operational purposes

TRITON: A Multi-GPU Open Source 2D Hydrodynamic Flood Model

M. Morales-Hernández^{a,b,*}, Md. B. Sharif^c, A. Kalyanapu^f, S. K. Ghafoor^c, T. T. Dullo^f, S. Gangrade^d, S.-C. Kao^d, M. R. Norman^e and K. J. Evans^{a,b}

^a Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

^b Climate Change Science Institute, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

^c Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA

^d Environmental Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

^e Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

^f Department of Civil and Environmental Engineering, Tennessee Technological University, Cookeville, TN 38505, USA

ARTICLE INFO

Keywords:

2D flood model
open source
multi-GPU
high-resolution
shallow water equations

ABSTRACT

A new open source multi-GPU 2D flood model called TRITON is presented in this work. The model solves the 2D shallow water equations with source terms using a time-explicit first order upwind scheme based on an Augmented Roe's solver that incorporates a careful estimation of bed strengths and a local implicit formulation of friction terms. The scheme is demonstrated to be first order accurate, robust and able to solve for flows under various conditions. TRITON is implemented such that the model effectively utilizes heterogeneous architectures, from single to multiple CPUs and GPUs. Different test cases are shown to illustrate the capabilities and performance of the model, showing promising runtimes for large spatial and temporal scales when leveraging the computer power of GPUs. Under this hardware configuration, communication and input/output subroutines may impact the scalability. The code is developed under an open source license and can be freely downloaded in <https://code.ornl.gov/hydro/triton>.

1. Software Availability

Name of the software: TRITON (Two-dimensional Runoff Inundation Toolkit for Operational Needs)

Contact address: Oak Ridge National Laboratory, 1 Bethel Valley Road, TN 37830, USA / Tennessee Technological University, 1 William L Jones Dr, Cookeville, TN 38505, USA

Email: moraleshern2@ornl.gov, akalyanapu@tntech.edu

Language: CUDA, C++

Hardware: Desktop/Laptop or clusters of CPUs/GPUs

Software: NVIDIA CUDA Toolkit, NetCDF (optional)

Availability: <https://code.ornl.gov/hydro/triton>


Year first available: 2020

2. Introduction

With the increasing frequency and intensity of extreme hydrologic events in a changing environment [36], faster and more accurate inundation models are particularly important tools for flood risk management. When translating the amount of rainfall (either from weather forecasting models or from rain gauge

*This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). 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 (<http://energy.gov/downloads/doe-public-access-plan>).

*Corresponding author

 moraleshern2@ornl.gov (M. Morales-Hernández)

ORCID(s): 0000-0001-6961-7250 (M. Morales-Hernández)

15 observations) all the way to surface inundation, coupled hydrologic-hydraulic models should be used, where
16 the flood inundation model represents “the last mile” that simulates flood wave propagation across high-
17 resolution terrain based on simulated runoff and streamflow from hydrologic models [65].

18 Flood inundation models can provide not just streamflow discharge but the temporal evolution of flood
19 location, depth and movement. However, this information is not always directly translated into practical
20 instructions to better support operators and emergency responders for more rapid decision making [52]. In
21 this regard, collaborative flood modeling is a great example for overcoming this challenge where end-users,
22 stakeholders and scientific knowledge meet and interact [19, 37]. Specifically, the use of high-resolution maps
23 built from flood inundation models together with non-technical terminology have proven to benefit decision-
24 making for end-users [50]. Beyond this methodology, integrated platforms using hydraulic modelling linked
25 with live geospatial information, remote sensing, flood visualization and community intercommunication
26 [66, 59], and other strategies such as probabilistic flood maps based on flood models and the Value of
27 Information (VOI) [1] have been also demonstrated to reduce the uncertainty and assess the consequences
28 of actions taken by decision-makers.

29 Despite the integrated flood risk management tools, large-scale operational hydrodynamics models are
30 still rarely used mainly because of their large computational cost. Simulation models must be particularly
31 fast (on the order of minutes) to meet decision-making needs resulting in the reliance on simplified and
32 analytical models. One example is the integrated version of Height Above Nearest Drainage (HAND) within
33 the US National Water Model (NWM) [34, 35]. HAND is based on Manning’s equation, some geometric
34 extrapolations, and precomputed rasters that are used to generate national flood maps at 10 m resolution.
35 Its European equivalent within the EFAS (European Flood Awareness System) is LISFLOOD-FP [4], a
36 spatially distributed rainfall-runoff-routing model. Its operational version [42, 63] uses the runoff generated
37 at a 5km grid spatial resolution as an input for a simplified shallow water model in which the convective term
38 is neglected. Flooding maps are derived from a catalog of flood hazard maps made of precalculated runs of
39 the LISFLOOD-FP model. The main limitation of these models is the range of applicability for all types of
40 flows and situations, which are restricted by the simplification hypothesis assumed, and consequently, the
41 validity and accuracy of the results.

42 One of the most complete inundation frameworks is the 2D full shallow water system [24], which solves
43 mass and momentum equations and provides water depth distribution and an accurate surface velocity
44 field. The resolution of the 2D shallow water equations is nevertheless computationally demanding at fine
45 resolutions and there is no obvious path for them to scale up to continental or global scales for operational
46 purposes. Two main efforts are highlighted in the last decades to overcome this constraint: efficient numerical
47 techniques and the adaptation to High Performance Computing (HPC). First, numerical improvements are
48 of crucial importance to reduce the computational burden. In the context of Augmented Riemann solvers,
49 a careful numerical estimation of topography and friction terms has been demonstrated to improve the
50 performance and accuracy of the solution [39, 40, 41]. Local time step methods [47, 12] are considered as a
51 worthwhile solution to improve the computational efficiency at the extra cost of having to deal with different
52 stages for flux and source computations, depending on the time level. As for the spatial discretization
53 – leaving aside adaptive mesh refinement techniques–, Discontinuous Galerkin (DG) schemes, as the one
54 proposed in [29], seem to deliver high quality solutions with desired scalability properties. On the same
55 page, a well-balanced no-neighbor method has recently been proposed for the 1D Saint Venant equations
56 [25]. Although the extension to the 2D framework is not clear, this family of schemes could open the door
57 to new efficient algorithms.

58 On the other side, parallel implementations, and the use of HPC on new architectures of modern super-
59 computers have become a fundamental requirement to study increasingly complex problems at large spatial
60 scale and high temporal resolution on water resources hydrodynamics [38]. Neal et al.[43] explored different
61 parallel strategies of LISFLOOD-FP across various types of architectures, stressing the efficiency, difficulty
62 and estimated development time of each implementation. The great majority of models are nevertheless
63 implemented on a single hardware type. One of the best exponents is PRIMo [48], a raster-based subgrid
64 flood model able to run on clusters of central processing units (CPUs) using both shared and distributed
65 memory. However, the use of Graphical Processing Units (GPUs) has become a new trend in the recent
66 decade [46, 7, 28, 6]. In the GPU context, the spatial discretization and efficiency have been studied for
67 regular [57] and irregular grids [44, 24, 17], seeking efficiency with i) different solvers [2], ii) different mem-

68 ory access patterns and mesh ordering algorithms for unstructured grids [31], iii) block uniform Quadtree
69 and adaptive mesh refinement on nested rectangular meshes [56], and iv) a local time stepping scheme to
70 improve performance [12]. Additional capabilities have been also reported and accelerated with the use of
71 GPUs: rainfall/runoff applications [32, 3], sediment transport and erosion processes [27, 8, 10], landslides
72 [30], hydraulic structures modelling [18, 11] and transport of contaminants and water quality models [58, 22].

73 Multi-GPU models could be the solution to improve the tradeoff among accuracy, speed and large scale
74 domains. The initial work by Sætra and Brodtkorb [46] studied both weak and strong scaling and the
75 effect of synchronizations of 2D shallow water equations using a 4-GPU machine. Xia et al. [65] used a
76 framework called HiPIMS to model a storm in a 2500 km^2 catchment using 8 GPUs and 100 million grid
77 cells around 2.5 times faster than real time. In [55, 54] the equations are discretized in a BUQ (quadtree)
78 grid following the proposed scheme of [56]. Although this type of spatial discretization could be convenient
79 to improve performance with respect to regular Cartesian grids and to address high resolution problems, the
80 domain decomposition increases complexity and could add an overhead in large scale simulations. In [51],
81 the authors compare two versions of solvers for the shallow water equations (finite-difference versus finite
82 volume). Weak and strong scaling up to 272 million grid cells are analyzed together with the CUDA-Aware
83 MPI feature, designed to optimize communications between the different sub-domains.

84 TRITON is presented here as the first (to date) multi-architecture (multiple CPUs and GPUs) open-
85 source 2D hydrodynamic flood model based on the resolution of full shallow water equations with source
86 terms. Different free or open source 2D models can be found in the literature: besides the LISFLOOD model
87 [4, 42] and the well-known HEC-RAS model (2D version released in 2016), other CPU-based (or multi-
88 CPU) models such as FullSWOF [13] and BreZo [49] solve the 2D shallow water equations in structured
89 and unstructured triangular grids, respectively. Delft3D [15] also permits 2D flow computations for different
90 applications ranging from hydrodynamics to sediment transport and water quality. Few GPU models are
91 freely available: GeoClaw [5] for certain applications [45], IBER for 2D flood modeling [23] and ANUGA,
92 enabling the GPU-offloading using PyCUDA [61]. TRITON has been nevertheless designed for a multi-
93 architecture paradigm and is able to run on several configurations including: single or multiple CPUs and
94 single or multiple GPUs, using a combination of Open Multi-Processing (OpenMP), Compute Unified Device
95 Architecture (CUDA), and Message Passing Interface (MPI). In addition to this architectural flexibility, all
96 computing subroutines are programmed only once, regardless of the hardware type, minimizing error sources
97 and bolstering the software portability. Finally, a simple input/output configuration is implemented that
98 would avoid significant geographic information system (GIS) pre- and post- processing. As an example, a
99 digital elevation model (DEM) is directly used as the computational mesh, circumventing the necessity of
100 site-specific mesh building as with most existing models. These features would eventually enable the use of
101 2D hydrodynamic models for operational purposes and other applications that were not feasible before.

102 In addition to introducing this TRITON as a fast and flexible open source suite to simulate both pluvial
103 and fluvial flood events –together with some freely available pre- and post-processing tools –, this work aims
104 to answer the following research questions:

- 105 1. Is it possible to simulate large temporal and spatial scales in the order of minutes using a hydrodynamic
106 model based on the solution to the full 2D shallow water equations? Which are the most appropriate
107 architectural and parallelization strategies required to achieve this?
- 108 2. Does spatially distributed rainfall/runoff have an impact on the results of predictive hydrodynamic
109 models?
- 110 3. Could communication and I/O times represent a bottleneck for large scales?

111 The paper is organized as follows: after describing TRITON, including the equations, numerical scheme
112 and the HPC implementation, the software features are presented. Three test cases are included to demon-
113 strate the capabilities of the model, showing the accuracy and performance of TRITON on different config-
114 urations and architectures, including the simulation of the multi-day event of Hurricane Harvey over Harris
115 County in Texas, on 384 GPUs using Summit supercomputer at Oak Ridge National Laboratory. The
116 trade-offs among parallel computation, communication and input/output are also analyzed, revealing the
117 importance of the latter two for large temporal and spatial scales.

118 3. The TRITON Model

119 TRITON, the Two-dimensional Runoff Inundation Toolkit for Operational Needs, is a physically-based
 120 hydrodynamic model that solves the 2D shallow water equations on a structured (Cartesian) grid. Based on
 121 the initial GPU model developed by Kalyanapu et al. [28], a new conservative numerical scheme has been
 122 implemented and integrated in an updated framework able run on multiple architectures.

123 3.1. Governing equations

124 The 2D shallow water equations express the depth-averaged conservation of mass and momentum in x
 125 and y directions of the space. They can be written in a compact differential conservative form as presented
 126 in eq. (1):

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S}_r + \mathbf{S}_b + \mathbf{S}_f$$

$$\mathbf{U} = \begin{pmatrix} h \\ q_x \\ q_y \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} q_x \\ \frac{q_x^2}{h} + \frac{1}{2}gh^2 \\ \frac{q_x q_y}{h} \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} q_y \\ \frac{q_x q_y}{h} \\ \frac{q_y^2}{h} + \frac{1}{2}gh^2 \end{pmatrix} \quad (1)$$

$$\mathbf{S}_r = \begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{S}_b = \begin{pmatrix} 0 \\ -gh \frac{\partial z}{\partial x} \\ -gh \frac{\partial z}{\partial y} \end{pmatrix} \quad \mathbf{S}_f = \begin{pmatrix} 0 \\ -\frac{gn^2}{h^{7/3}} q_x \sqrt{q_x^2 + q_y^2} \\ -\frac{gn^2}{h^{7/3}} q_y \sqrt{q_x^2 + q_y^2} \end{pmatrix}$$

127 The vector \mathbf{U} represents the conserved variables (i.e. the unknowns of the system) and includes the water
 128 depth, h [L], and the x and y unit discharges, called q_x [$L^2 T^{-1}$] and q_y [$L^2 T^{-1}$] respectively. Eq. (1) also
 129 contains the fluxes of these conserved variables (\mathbf{F} and \mathbf{G}) –being g [LT^{-2}] the gravity acceleration– and the
 130 source terms. The latter encompass runoff terms, \mathbf{S}_r , expressed according to the runoff rate r [LT^{-1}]; bed
 131 slope terms, \mathbf{S}_b , accounting for the gradient of the elevation z [L]; and friction terms, \mathbf{S}_f , modeled by means
 132 of Gauckler-Manning's law in terms of the Manning's roughness coefficient n [$TL^{-1/3}$]. Here runoff refers to
 133 the effective rainfall (i.e., total rainfall minus losses due to infiltration, abstraction, and evapotranspiration)
 134 plus baseflow that are typical outputs from hydrologic or land surface models.

135 3.2. Numerical scheme

136 A finite volume upwind explicit scheme is used to solve Eq. (1) in a squared (Cartesian) mesh of grid
 137 spacing Δx . An Augmented Roe (ARoe) solver is implemented, based on [39, 17] for the fluxes and bed
 138 slope source terms nevertheless a different estimation of bed slope source terms at each edge are proposed
 139 in this work. This treatment also ensures the positivity of the solution without reducing the time step size.
 140 For the sake of clarity, the derivation of this part of the scheme can be found in Appendix A. Friction terms
 141 are discretized using a local implicit formulation [64] that does not alter the explicitness of the scheme.
 142 Accordingly, a two-step algorithm is proposed for the update of a cell i from time t^n to time $t^{n+1} = t^n + \Delta t$:

$$\mathbf{U}_i^* = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x} \sum_{k=1}^4 \sum_{m=1}^3 \frac{\tilde{\lambda}_m^-}{\tilde{\lambda}} \left[(\tilde{\lambda} \tilde{\alpha} - \tilde{\beta}_b) \tilde{\mathbf{e}} \right]_{m,k}^n \quad \mathbf{U}_i^{n+1} = \mathcal{F}(\mathbf{U}_i^n, \mathbf{U}_i^*) + r_i^n \Delta t \quad (2)$$

143 where, at each interface k , $\tilde{\alpha}$ and $\tilde{\beta}_b$ are the fluxes and bed slope source term linearizations, "minus"
 144 superscript accounts for the upwind discretization and $\tilde{\lambda}$ and $\tilde{\mathbf{e}}$ are the eigenvalues and eigenvectors of the

145 system of equations respectively (see Appendix A). The localized runoff rate is denoted by r_i while function
 146 \mathcal{F} stands for the friction discretization, written as:

$$\mathcal{F}^1 = h^* \quad \mathcal{F}^2 = -(q_x^*) \left(\frac{1 - \sqrt{1 + 4S_f}}{2S_f} \right) \quad \mathcal{F}^3 = -(q_y^*) \left(\frac{1 - \sqrt{1 + 4S_f}}{2S_f} \right) \quad (3)$$

147 where

$$S_f = \frac{\Delta t g n^2 \sqrt{(q_x^*)^2 + (q_y^*)^2}}{(h^n)^{7/3}} \quad (4)$$

148 The explicit character of the scheme restricts the time step size according to the Courant-Friedrich-Lewy
 149 (CFL) condition:

$$\Delta t = \text{CFL} \min_i \left\{ \frac{\Delta x}{\left| \frac{q_x}{h} \right|_i + \sqrt{gh_i}, \left| \frac{q_y}{h} \right|_i + \sqrt{gh_i}} \right\} \quad \text{CFL} \leq 0.5 \quad (5)$$

150 where index i loops over the number of grid cells. Note that, although formally the maximum wave speed
 151 should be estimated at the interfaces, the cell values are used instead in this work. Notwithstanding, this
 152 approach does not compromise the stability of the scheme—in fact, a value equal to or less than the size of
 153 the time step is selected using this formula—but it simplifies and ultimately accelerates the computations,
 154 providing simultaneously a new way to estimate some corrections for the source terms (see Appendix A).

155 3.3. HPC implementation

156 Increased problem complexity motivates heterogeneous HPC for hydrodynamics codes in the new era
 157 of parallel computing [38]. Computation time can be reduced effectively with the use of clusters of CPUs
 158 and GPUs. On-demand cloud workstations are also becoming more popular and affordable, allowing the
 159 simulation of larger spatial and temporal domains at finer scales. Varied programming paradigms have arisen
 160 as a consequence of this heterogeneity: Open Multi-Processing (OpenMP) for multicore CPUs, Message
 161 Passing Interface (MPI) for clusters, Compute Unified Device Architecture (CUDA) or Open Computing
 162 Language (OpenCL) for GPUs. For that reason, TRITON has been designed as a multi-architecture single
 163 code base, able to run on the following platforms through specific compilation instructions:

- 164 1. Multi-core shared memory platform using OpenMP
- 165 2. Multi-node cluster using MPI or MPI + OpenMP
- 166 3. Single node GPU machine using CUDA
- 167 4. Multi-node GPU cluster using MPI + CUDA

168 The simplified flowchart for the current implementation is depicted in Figure 1(a). After reading and
 169 parsing the input data (see Section 4.1), the domain is decomposed into different subdomains according to
 170 the desired number of MPI sub-tasks. Then, the simulation starts and runs until the time reaches the final
 171 simulation time, writing the output information (described in Section 4.2) each output interval. At each
 172 time state, the time step size is first computed according to Eq. (5); then the computing kernels (GPU)
 173 or subroutines (CPU) are executed, merely accounting for the numerical scheme in Eq. (2); finally the
 174 information is exchanged between the corresponding subdomains. Details of the later three processes along
 175 with the MPI decomposition are explained below.

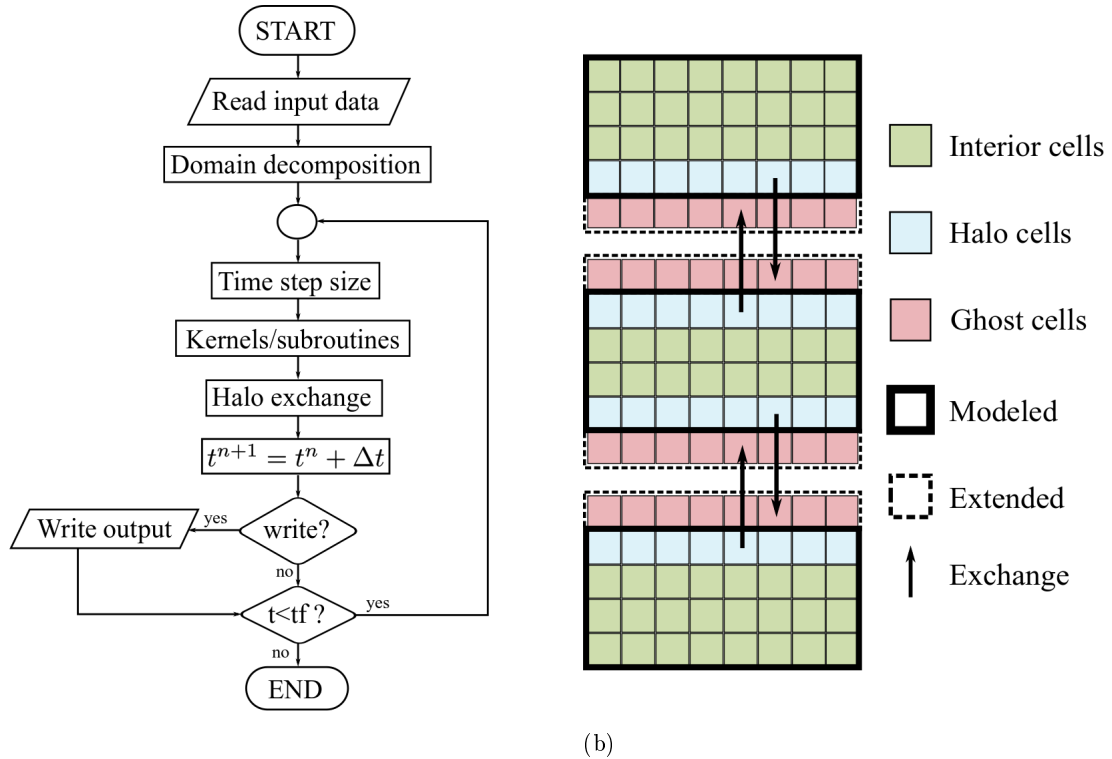


Figure 1: TRITON flowchart (a) and row-wise domain decomposition (b)

3.3.1. Domain decomposition and halo exchange

176 Domain decomposition is an important factor for large-scale parallelization using MPI libraries (or sim-
 177 ilar) in design of portable and scalable communication between subdomains. Although there are different
 178 ways to partition the information, a 1D row-wise decomposition is applied here for simplicity [38]. As the
 179 stencil for the numerical scheme Eq. (2) involves neighbouring data (see Figure 9(a), Appendix A), the
 180 information has to be exchanged each time step and overlap computations are performed. Figure 1(b)
 181 shows a sketch of the row-wise approach, where north-south communication is required and the halo size
 182 is the number of columns of the whole domain. A two-step non-blocking algorithm is implemented using
 183 *MPI_Isend* and *MPI_Irecv*, with the aid of *MPI_Wait* to wait for an MPI request. Let N be the number
 184 of ranks (e.g., the number of partitions of the domain). First, all subdomains $n < N$ will send the halo data
 185 to the subdomain $n + 1$ and receive from subdomain $n - 1$. Then, vice versa, each subdomain $n > 1$ will
 186 receive information from subdomain $n + 1$ and send to subdomain $n - 1$.

188 The subdomain exchange imposes a slightly greater level of complexity when dealing with a GPU imple-
 189 mentation due to memory allocation. As computations are performed by a GPU, data are fully allocated in
 190 the device (GPU) memory. However, the regular MPI calls require pointers to host (CPU) memory, which
 191 requires an additional data copy between host and device, in addition to an extra host memory alloca-
 192 tion. This might decrease the performance of the model. The CUDA-Aware MPI is used for halo exchange
 193 to overcome this challenge of possible performance degradation. CUDA-Aware MPI allows GPU to GPU
 194 direct communication via network, bypassing the CPU, if underlying hardware supports this technology.
 195 Specifically, Summit supercomputer has GPU direct communication support and the use of CUDA-Aware
 196 MPI has resulted in improved TRITON's performance. The impact of using CUDA-Aware MPI versus the
 197 conventional approach has been already studied in [51] (see results FVS and FVG) for a primitive version of
 198 TRITON. For the sake of flexibility, TRITON supports both approaches, allowing users to choose a preferred
 199 implementation according to their system requirements.

3.3.2. Time step size computation

In order to guarantee a stable and reliable solution at every time stage, Δt is limited according to Eq. (5), which requires the computation of a global minimum time step size that will be imposed to evolve the solution in time. To do that, each subdomain computes first its own local minimum time step size for all its cells. Again, two different implementations can be distinguished here depending on the architecture. The OpenMP approach is rather simple since it only consists of the computation of a *reduction* operation. The equivalent reduction operation in GPU is more complicated. Although some CUDA standard libraries (CuBlas, Thrust) or third party (Cub) libraries can be used, a reduction *ad-hoc* function has been implemented in CUDA. In contrast to the existing libraries, in which a global array of size the total number of cells is required and then perform the reduction operation, we make use of shared memory and thread synchronization at each kernel –where we calculate the time step size–, to launch multiple instances using a reduced global array of size "thread block" times smaller than the original size. Then, the reduction operation is done over the global reduced array, diminishing the global memory use, improving memory coalescing and providing a better performance than the existing libraries for large scale problems. Once the local minimum time step size is computed, the global minimum is found using *MPI_AllReduce* across all the processes (sub-tasks).

TRITON also offers the possibility of a constant time step size. This implementation does not require any reduction operation –making it easier– nevertheless the accuracy and robustness of the results are not ensured since the maximum allowed time step size, governed by Eq. (5) might be violated.

3.3.3. Kernel/subroutine execution

TRITON is written using C++ and CUDA. Each computing module has been implemented as a CPU subroutine as well as a CUDA kernel. During compilation, based on the computing platform, appropriate computing modules are compiled. A set of arrays with the same structure is allocated either in the CPU or in the GPU so that the difference between CPU and GPU execution only consists of the memory where the data is defined and the kernel/subroutine calls. Each kernel is then only programmed once, avoiding duplicated information. This fact improves the readability of the code at the same time as reproducibility and trustfulness between different architectures, minimizing or almost eliminating eventual human mistakes usually made when porting the code from one to another architecture. Figure 2 depicts the source code for the "wet/dry" kernel/subroutine, showing both the function call and declaration. As shown, the number of arguments required by the CPU and GPU versions are exactly the same, although arrays reside either in the device or the host memory according to the chosen architecture. Some *#ifdef*, *#else* and *#endif* directives allow to switch between CPU and GPU in compilation time.

4. Software Features

TRITON is a UNIX-based model targeting laptops, desktops, and optimized for supercomputers, leveraging the current power of workstations. The code, pre-/post-processing tools and some samples can be found in

<https://code.ornl.gov/hydro/triton>. The main features and tools are explained below.

4.1. Input data

The mandatory/optional TRITON input files include:

- **Configuration** (mandatory). A text file containing the path of all input files, output interval and format, and all parameters and constants needed for simulation (e.g., number of streamflow sources, external boundary conditions, initial and final time, CFL number, switches to enable/disable observation point and checkpointing, etc.). It can be configured either manually or with the aid of a Config File Tool Generator.
- **Topography** (mandatory). TRITON is a DEM-based code. Consequently, the mesh used for computation is a Cartesian (square) grid obtained directly from the DEM file (i.e., avoiding the ad-hoc and site-specific task of computational mesh building). DEMs follow the ESRI raster file format – both ASCII and binary formats are allowed, although binary format is recommended for large spatial


```

1
2 //variable definitions
3 ACTIVE_GPU: flag to enable GPU compilation           ACTIVE_OMP: flag to enable CPU compilation
4 nrows: number of rows                               ncols: number of columns
5 THREAD_BLOCK: group of threads executed in parallel  global_dt: time step size
6 device_vec: array containing all the GPU vectors     host_vec: array containing all the CPU vectors
7 H,HU,HV,DEM: integers pointing to the beginning of   id: temporary integer ranging all the cells
8 the water depth, x-unit discharge, y-unit-discharge
9 and elevation vectors respectively
10
11 //kernel/subroutine call
12 #ifndef ACTIVE_GPU
13     Kernels::wet_dry<< <(nrows*ncols+THREAD_BLOCK-1)/THREAD_BLOCK,THREAD_BLOCK,0,streams>> >(nrows*ncols,
14     nrows, ncols, global_dt, device_vec[H], device_vec[HU], device_vec[HV], device_vec[DEM]);
15 #else
16     Kernels::wet_dry(nrows*ncols, nrows, ncols, global_dt, host_vec[H], host_vec[HU], host_vec[HV],
17     host_vec[DEM]);
18 #endif
19
20 //kernel/subroutine declaration
21 template<typename T>
22 #ifndef ACTIVE_GPU
23     __global__
24 #endif
25 void wet_dry(int size, int nrows, int ncols, T dt, T *h_arr, T *hu_arr, T *hv_arr, T *dem)
26 {
27     #ifndef ACTIVE_GPU
28         int id = blockIdx.x * blockDim.x + threadIdx.x;
29         if (id >= size)
30             return;
31     #else
32         #pragma omp parallel for
33         for (int id = 0; id < size; id++)
34         {
35             //Kernel/subroutine implementation. Note that this is common for both architectures
36             .....
37 #ifndef ACTIVE_OMP
38         }
39 #endif
40     }

```

Figure 2: Source code for the call and declaration of "wet/dry" kernel/subroutine. GPU code is shown in green while CPU code is displayed in dark blue.

domains— that contains a header section with the number of columns, rows, origin coordinates and cell size, as well as a matrix of elevation values. NODATA values are not allowed in the current version of TRITON.

- **Streamflow hydrograph** (optional). Streamflow hydrograph is one of two possible hydrologic inputs to TRITON. Streamflow here refers to the point discharge typically from upstream incoming river channels (sources). When selecting streamflow hydrograph, two files are required: an (x, y) coordinate list with the location of all inflow sources and a streamflow hydrograph table including the time (in hours) and the timeseries of discharge (in cubic meters per second) at each source j , $Q_j(t)[L^3T^{-1}]$. The discharge is introduced in TRITON as a single mass release.
- **Runoff hydrograph** (optional). Runoff hydrograph is another possible hydrologic input to TRITON. It is a common output from various hydrologic and land surface models (immediately before such information is used for streamflow routing). By involving runoff in TRITON, the model can simulate local (pluvial) floods and hence increases its functionality. When selecting runoff hydrograph, two files are also required. The runoff regions (corresponding to the coarser hydrologic model grids) are defined in the form of a matrix raster map with the same format with the DEM file (but without the header). Each distinct area is labeled with a non-negative integer number that serves as a unique region identifier (links to the runoff hydrograph table). As in the streamflow hydrograph table, the runoff hydrograph table contains the time (in hours) and timeseries of runoff rate r (in mm per hour)

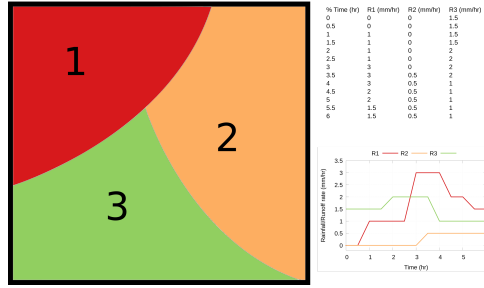


Figure 3: Example of runoff map and table

from all declared runoff regions. A sketch of the rainfall/runoff input files is depicted in Figure 3.

- **Roughness** (optional). The surface friction is represented by Manning’s roughness coefficient n . It can be provided either as a constant number, specified in the configuration file and applied globally to the whole domain, or in the form of a matrix raster map (without header) matching the number of DEM grid cells. The latter choice allows the user to define spatially varied roughness coefficients to represent the site-specific land use and land cover conditions.
- **External boundary conditions** (optional). The boundaries of the domain (north, east, south and west) can be provided with five type of external boundary conditions. By default, all boundaries are closed (i.e., water flux is prevented from exiting the domain). Four alternative flow conditions can be then be imposed:

 - Type 0: zero gradient. The original setup in [28].
 - Type 1: level versus time. An additional file containing a table with the time and the timeseries of water level is mandatory.
 - Type 2: normal slope. The desired slope is required.
 - Type 3: Froude number. The Froude number defined as $Fr = \frac{|\mathbf{q}|}{h\sqrt{gh}}$ is needed to be imposed across the external boundary, being $\mathbf{q} = (q_x, q_y)$.

More than one external boundary condition can be defined at each boundary edge. The user only has to provide the initial/ending (x, y) coordinates of the boundary and its corresponding boundary condition parameters according to the boundary type. The external boundary conditions are imposed at the ghost cells surrounding the whole domain. This implementation ensures the scalability of the solution for domain decomposition since no additional information has to be exchanged between the partitioned subdomains.

- **Initial conditions** (optional). A dry domain is the default initial condition. However, the user can also specify an initial condition for each conserved variable $(h, q_x$ or $q_y)$ in the configuration file, in the form of a matrix type file (without header). Furthermore, checkpointing/hotstart is also allowed. Backup files are written during the computation to retrieve the simulation from the last state in case it is necessary.
- **Point output** (optional). In addition to the default matrix output at user-specified time intervals, TRITON also supports output of timeseries at user-specified locations (to avoid data processing for known points of interest, such as locations with gauge observations). An (x, y) coordinate list file containing the location of the desired points is therefore required.

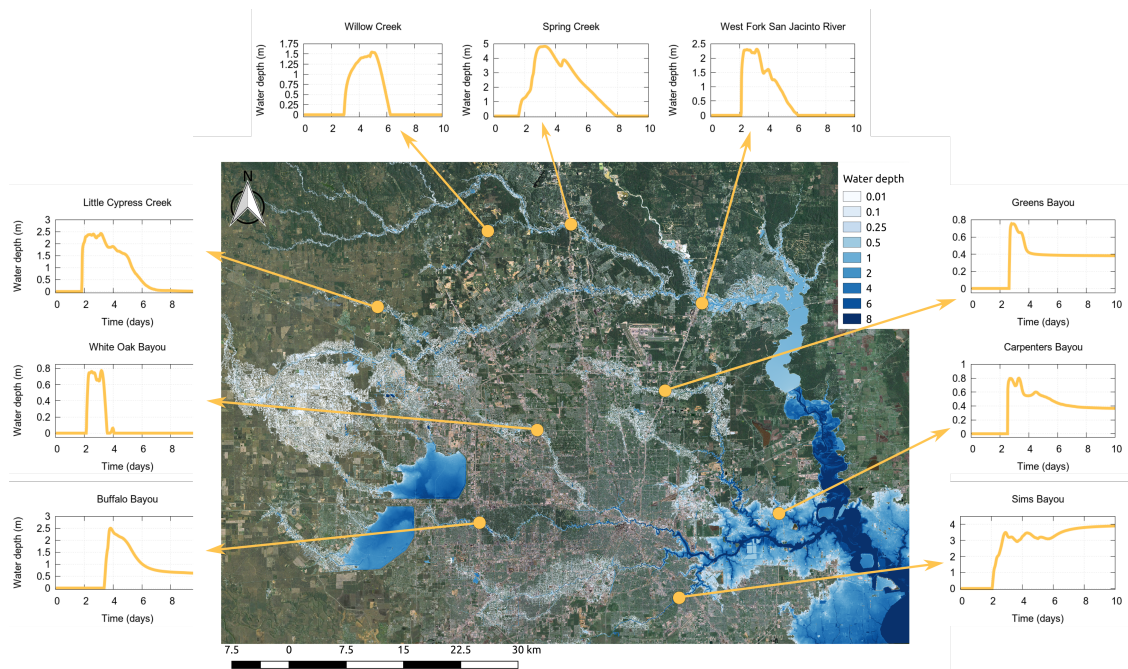


Figure 4: Example of output information obtained with TRITON

4.2. Output data

The output from TRITON is intended to be easy to post-process by standard graphing and GIS tools. Two types of output data (spatial and temporal) are generated in a separate folder at each user-specified interval.

- Matrix data** (spatial). A snapshot of water depth and unit discharge is written in the form of a matrix type file (without header) either in ASCII or in binary format (the latter is recommended for large scale domains [38]). Additionally, if the execution is run using domain decomposition (either multi-CPU or multi-GPU), a switch in the configuration file allows the user to choose the mode in which the data is written: 1) sequential that gathers all subdomain information in a single file during the computation, or 2) parallel that directly outputs subdomain data as separate files. For the latter, a separate script can then be used to combine all subdomain information into a single file during post-processing.
- Water depth and unit discharge profiles** (temporal). The timeseries of water depth and unit discharge at the specified point locations (defined in the input files) can be outputted in a single file for each variable.

Figure 4 displays a style of output information that can be obtained. It corresponds to the Hurricane Harvey test case studied in Section 5.3 and condenses in a single panel the stage hydrograph for each predefined observation point as well as a 2D view of the whole spatial domain, showing the topography and the maximum flooded area. Note that to be consistent with Eq. (1), TRITON outputs unit discharge (rather than velocity). The user should conduct proper conversion to estimate velocity based on unit discharge and water depth.

4.3. Other tools

A suite of tools is provided with TRITON in order to simplify some pre- and post-processing tasks. First, some bash scripts are included for the conversion between binary/ASCII formats and sequential/parallel mode. They can be used for both input and output data. Additionally, two programs containing a GUI

322 targeting Windows and Linux are included in the repository. The first one is a NetCDF converter, that takes
 323 both binary and ASCII files outputted from TRITON as input data, and converts them into NetCDF format.
 324 The second GUI software called Config File Tool Generator allows the user to generate the configuration
 325 file (mandatory input file for TRITON).

326 5. Test Cases

327 A set of test cases are proposed here to demonstrate certain desired characteristics such as consistency,
 328 stability, convergence and robustness, and model capabilities. As TRITON is intended to be a reference
 329 software for the computation of large scale flood problems, reliability on the output results is mandatory.
 330 The first test case is therefore included as a verification and to show how the model behaves on a classic
 331 literature test case that involves all kind of flows (subcritical, supercritical, and sonic transitions), and to test
 332 the accuracy and grid convergence of the model. Once the model is able to provide a robust and trustworthy
 333 solution, the second test is designed to test the capability of a model to accept runoff hydrograph as a
 334 hydrologic input and illustrate its importance. This feature also allows users to simulate the effects of local
 335 (pluvial) floods that occur due to highly intense local precipitation in a non-floodplain region. The final
 336 test case is oriented to evaluate the performance of the model on different architectures (multiple CPUs and
 337 GPUs), analyzing the possible bottlenecks in large temporal and spatial scales. All test cases introduced
 338 below can be found in the repository. Additional test cases will be included as they are developed to provide
 339 users with further examples.

340 5.1. Test case 1: paraboloid bed topography

341 This test case consists of a square domain $[0, 4] \times [0, 4]$ with a frictionless paraboloid topography. The
 342 initial water depth condition is a planar surface with velocity in the y-direction. The domain boundaries are
 343 closed and after one period ($t=1T$), the numerical solution should recover the initial condition due to the
 344 rotating velocity field. This configuration is considered to be a challenge test where some numerical methods
 345 fail when trying to reproduce it, mainly due to an incorrect treatment of wet/dry interfaces or a non-balanced
 346 source terms/fluxes implementation. More details about the analytical periodic solution are described in [14].
 347 The package provided there is used to generate four different resolution grids: $\Delta x = 0.04, 0.02, 0.01, 0.005m$.
 348 The tolerance h_{tol} (see Appendix A) is set to 10^{-4} in this particular case without friction and $CFL=0.45$.

349 Three periods ($t=3T$) are simulated and the numerical results obtained by each grid resolution are
 350 compared to the exact solution, understood as the projection of the analytical solution on each computational
 351 grid. Figure 5 depicts those comparisons at $t=1T$, $t=2T$ and $t=3T$.

352 Accuracy decreases as time advances, especially with lower resolution. This is a consequence of the first
 353 order only numerical diffusion, which is resolution dependent. That said, the use of higher order schemes
 354 is not recommended since both pressure and dissipative terms usually dominate over the convective terms
 355 with the roughness of realistic applications. Also, the presence of wet/dry discontinuities would downgrade
 356 the accuracy to first order so that the cost of the implementation is not justified. Nonetheless, a correct
 357 estimation of the source strengths is mandatory to avoid dramatic reductions in the time step size and to
 358 ensure a correct well-balanced implementation [39, 41, 17].

359 To provide a quantitative measure of the error made for each resolution and to check the convergence
 360 rate, L_1 , L_2 and L_∞ error norms are computed with respect to the analytical solution at $t=3T$. Table 1
 361 contains those error norms computed as:

$$L_1(y_{\Delta x}) = \frac{1}{N} \sum_{i=1}^N |(y_n - y_e)_i| \quad L_2(y_{\Delta x}) = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_n - y_e)_i^2} \quad L_\infty(y_{\Delta x}) = \max_i |(y_n - y_e)_i| \quad (6)$$

362 where N is the number of grid cells and y_n and y_e are the numerical and exact solutions respectively.

363 These errors confirm the consistency of the scheme for a first order method. Even though this test case
 364 involves water depth values on the order of centimeters, millimeters, or below, together with strong velocities,
 365 the scheme is demonstrated to be robust and accurate enough, providing a solution free of oscillations, which
 366 is particularly of interest when dealing with wet/dry boundaries – a key factor in 2D flood models–.

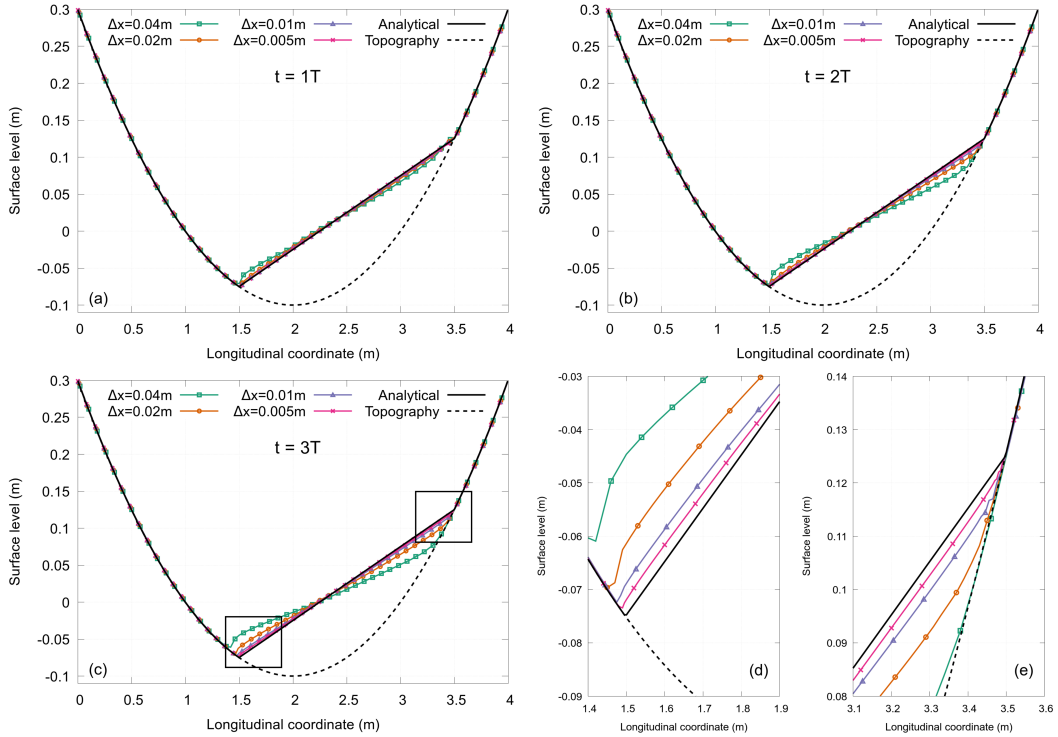


Figure 5: Exact and numerical solutions for different grid resolutions at (a) $t=1T$, (b) $t=2T$ and (c) $t=3T$. Panels (d) and (e) show a zoom of the left and right wet/dry interfaces respectively, at $t=3T$.

Δx	L_1 norm	L_2 norm	L_∞ norm
0.04	3.943e-03	9.612e-03	4.630e-02
0.02	2.065e-03	5.137e-03	2.405e-02
0.01	9.724e-04	2.457e-03	1.090e-02
0.005	4.433e-04	1.117e-03	5.311e-03

Table 1

L_2 , L_1 and L_∞ error norms for each resolution for the paraboloid test problem

5.2. Test case 2: runoff capability

TRITON admits runoff hydrographs to provide spatiotemporally distributed runoff as an input. Typically, a hydraulic model is driven by providing streamflow hydrographs at user-specified locations to simulate riverine (fluvial) floods. These locations can be where the observations of streamflow were made or can match a river routing model. However, the lack of sufficient streamflow source locations, due to hydrologic model limitations or other factors, may lead to the underestimation of flood extents, especially at smaller tributaries. To overcome this issue, users can utilize the “runoff function” within TRITON.

As an example to illustrate this capability, we select an area located upstream of Allatoona Reservoir in the northwest Georgia, US. The region was previously modelled by [21] [20] to study probable maximum flood using the Distributed Hydrologic Soil Vegetation Model (DHSVM) and Flood2D-GPU [28]. The 100-year peak streamflow was estimated at the outlet of the computational domain following the guidelines of Bulletin 17B [60], and then used to prepare 100-year return level streamflow hydrographs. For a detailed description, readers are referred to Section 3.1 of [20]. In TRITON, the 100-year flood is simulated with two different model configurations: Scenario A (no runoff) - using 13 streamflow source locations along the river network (Figure 6(a), and Scenario B (with runoff) - using 2 upstream streamflow source locations and runoff simulated at various catchments (Figure 6(b)). The computational domain spans 358 sq. km and consists of 400,000 grid cells at a 30m DEM resolution. In both scenarios, a 5-day simulation is conducted

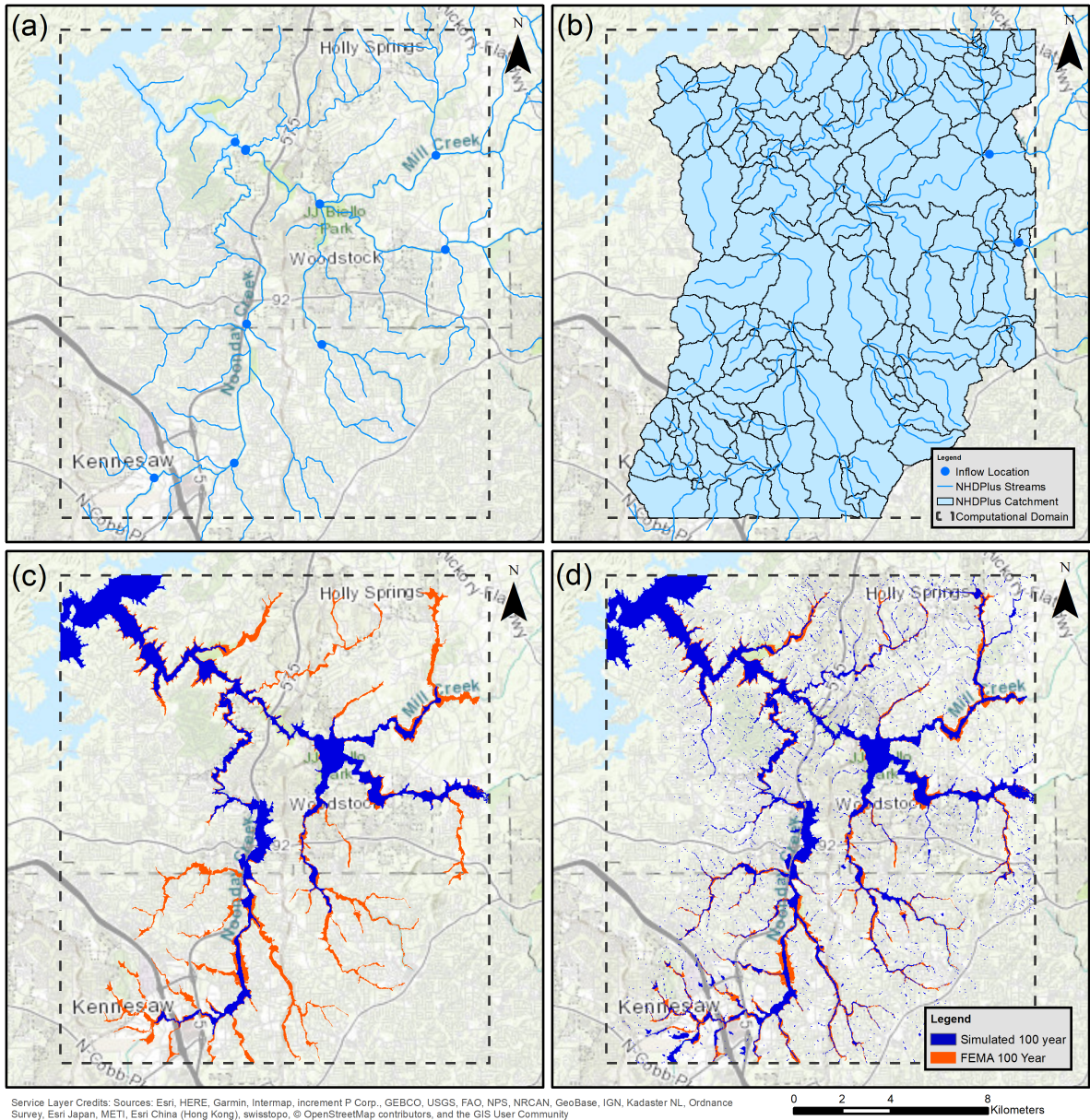


Figure 6: Comparison of runoff functionality. Panel (a) shows streamflow source locations for Scenario A (no runoff), and panel (b) shows streamflow source locations and runoff regions for Scenario B (with runoff). Panels (c) and (d) are the TRITON simulated 100-year flood plain against FEMA 100-year flood plain for Scenarios A and B, respectively.

384 (with peak streamflow occurring at day-3). The water depth is written in raster maps every 30 minutes.
 385 The maximum flood inundation extents for both scenarios, compared against a benchmark dataset from
 386 FEMA (100-year flood zones), are presented in Figure 6(c) and Figure 6(d), respectively. A qualitative compar-
 387 ison reveals that just using streamflow inputs (Scenario A, Figure 6(c)) may lead to the underestimation
 388 of flood extents, especially in tributaries where no upstream streamflow inputs are specified. In Scenario
 389 B, since the model is driven by runoff hydrograph across the entire domain, it can better capture the flood
 390 extents especially in the tributary areas. To evaluate the two scenarios quantitatively, we use a metric called
 391 “hit rate” which is a measure of model tendency to accurately predict the benchmark flood extents [20]
 392 [62]. Using FEMA 100-year flood plain as a benchmark, we obtain a hit rate of 0.50 for Scenario A and an

Summit nodes	multi-CPU			multi-GPU		
	# hardware cores	#MPI tasks	Runtime (h)	# devices	#MPI tasks	Runtime (h)
1	168	1	107.99	6	6	2.40
2	336	2	56.73	12	12	1.44
4	672	4	28.78	24	24	0.90
8	1344	8	14.39	48	48	0.63
16	2688	16	7.57	96	96	0.49
32	5376	32	3.68	192	192	0.43
64	10752	64	2.73	384	384	0.42

Table 2

Hardware configuration and runtimes for the Harvey test problem

393 improved rate of 0.68 for scenario B. Not only demonstrating a better floodplain simulation, local pluvial
 394 floods can also be simulated in Figure 6(d). The results showcase the advantage of using runoff for better
 395 and easier flood regime simulation.

396 5.3. Test case 3: Hurricane Harvey

397 The last test case is intended to evaluate the performance and applicability of TRITON using different
 398 architectures (multiple CPUs and GPUs) in a realistic configuration. In this scenario, we simulate the
 399 massive flood that Hurricane Harvey caused in the summer of 2017 along the US Gulf Coast. The spatial
 400 domain encompasses around 6800 km² of Harris County, Texas, US. The simulation covers ten days, with
 401 the heaviest rainfall occurring during day 7-to day-9. The runoff data are generated using the Variable
 402 Infiltration Capacity (VIC) hydrologic model [33] driven by hourly 4km radar-based Stage IV Quantitative
 403 Precipitation Estimate. This runoff is then routed using the Routing Application for Parallel computatIon
 404 of Discharge (RAPID) model [9] through the river network to generate streamflow input at 69 inflow source
 405 locations. Information from the US Geological Survey National Elevation Dataset is used to build a corrected
 406 DEM in which in we included Addicks and Barker’s reservoir capacities, incorporated the bathymetry of the
 407 main river segments and eliminated some noise present in the original data. The DEM resolution is 10m,
 408 with a total of around 68 million grid cells, and the Manning’s roughness coefficient is set to a constant
 409 value of $n = 0.035s/m^{1/3}$. The output is configured in sequential binary format, with an output interval of
 410 1800s. More information about this test case can be found in [16].

411 The purpose of this test case is to evaluate the efficiency and scalability of the model on different hardware
 412 configurations. Simulations are carried out on Summit supercomputer at Oak Ridge National Laboratory.
 413 Summit is comprised of 4,608 compute nodes, each of them containing six NVIDIA Volta V100 GPUs and 42
 414 physical cores (168 hardware cores using Simultaneous Multithreading). In order to make a fair comparison
 415 between multiple architectures, a Summit compute node is chosen and seven numerical experiments are
 416 performed using up to 64 nodes. Two TRITON hardware configurations are used: multiple CPUs using
 417 MPI + OpenMP and multiple GPUs using MPI+CUDA, each of them with the aim of minimizing the
 418 number of MPI tasks per configuration. Consequently, the multi-CPU version uses one MPI task per node,
 419 while the multi-GPU version uses one MPI tasks per GPU. CUDA-Aware MPI is used for the multi-GPU
 420 simulations. Table 2 condenses both hardware configurations as well as displays the runtimes.
 421 Following [51] we also define the following metrics: Billion Lattice Updates per Second (BLUPS) and speed-
 422 up:

$$423 \text{BLUPS} = \frac{N_c \times N_{ts}}{T_n \times 10^9} \quad \text{speed-up} = \frac{T_1}{T_n} \quad (7)$$

423 where N_c is the number of grid cells, N_{ts} is the number of time steps done and T_n is the runtime achieved
 424 using n computing nodes. The speed-up in this case measures how fast the simulation is compared to the
 425 runtime using 1 node. Figure 7 (left) depicts the runtime in log scale (y -axis) and the number of BLUPS
 426 (x -axis) achieved by each hardware configuration (plotted in circles of different sizes).

427 The main result is that a very large problem for a serial code can be computed more reasonably with
 428 a multi-GPU system, with runtimes of less than 30 minutes. Even with the lowest hardware configuration

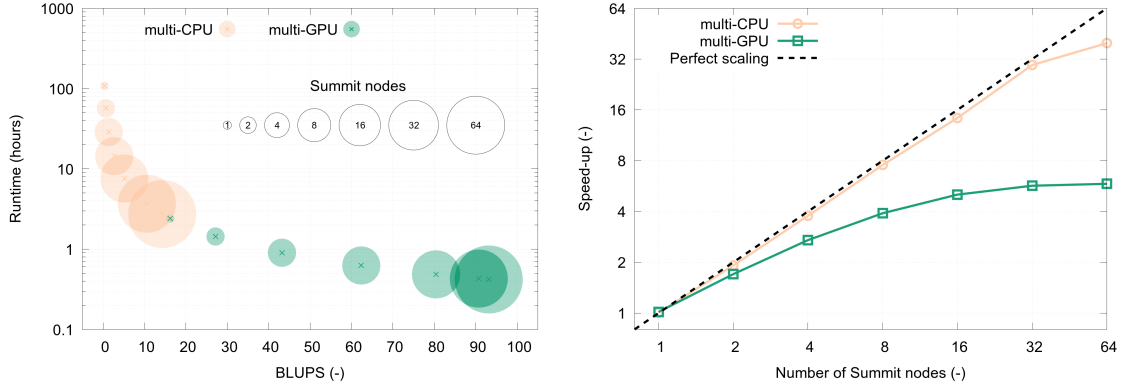


Figure 7: Runtime and BLUPS (left) and speed-up (right) achieved by each configuration for the Harvey test problem

(1 node, 6 GPUs), the multi-GPU version is able to achieve a lower runtime than the most demanding multi-CPU hardware configuration (64 nodes, 10752 OpenMP threads). Accordingly, the maximum BLUPS achieved by the multi-CPU system is still lower than the 1-node multi-GPU configuration. Additionally, given a fixed number of Summit nodes, the average BLUPS ratio between the multi-GPU and multi-CPU versions is 25x, with a maximum of 45x for 1 node, revealing the convenience of using this architecture. The circle series turns horizontal when the strong scaling limit is reached, and the asymptote is representative of the absolute fastest runtime expected for each configuration in this problem. With this, the multi-CPU system has not reach its maximum using 64 nodes while the multi-GPU version tails off significantly beyond 8 or 16 nodes.

Figure 7 (right) shows in log-log scale the speed-up relative to the simulation using 1 node (y -axis) as a function of node count (x -axis), compared with the perfect scaling. As shown, the multi-CPU version is able to achieve a satisfactory scaling factor up to 32 nodes (32 MPI tasks) although a performance hit is observed for 64 nodes. This might be due to the row-wise (1D) MPI partitioning. On the other hand, the multi-GPU speed-up does not scale acceptably. Although 6 MPI tasks are used per Summit node (1 per GPU), a lower speed-up is observed even for 32 MPI tasks (extrapolation between 4 and 8 nodes), where adequate speed-ups were achieved for the multi-CPU version. Many factors are responsible for this: first, this test case does not have runoff as an inflow and only around 20% (as an average) of the domain is wet. A loss of efficiency is therefore caused by thread divergence due to an “if statement” implemented in the code to avoid the computations on dry cells. This is an issue reported for CUDA and it is not the case with OpenMP since each CPU core runs like a single-threaded subroutine, executing its own independent set of instructions. Additionally, a static MPI subdomain partitioning aggravates this fact since some subdomains could not have a significant computational burden, resulting in an imbalance among MPI processes [38]. To investigate other sources of scaling limitations in the multi-GPU system, the computation time (GPU), the MPI communication time, the I/O time and the rest are plotted in Figure 8. Two additional simulations are carried out using 1 and 3 GPUs. The left panel shows (in log-log scale) the absolute number (in minutes) while the right plot displays the percentage of the time consumed by each process against the GPU count.

Note that memory copies between the host and the device are counted within the GPU time and the “Other” time stands for subroutine calls and basic operations run on the CPU. As observed on the left plot, the GPU time is reduced at almost the same rate for each configuration, meaning that it actually scales according to the number of GPUs. A slightly lower rate is detected for the last two values (192 and 384 GPUs) due to the low number of grid cells per GPU –each GPU does not have enough work to leverage its computing power. I/O and other (CPU) time remain almost constant for all configurations as expected. Communication time using a blocking algorithm and CUDA-Aware MPI does not increase noticeably as the number of GPUs grows, indicating that communicating every time step carries significantly more weight than the number of GPUs.

The right panel demonstrates that MPI and I/O times govern this problem for large GPU count. In particular, from 40% to 66% of the time in this test case is consumed by these processes at 96 GPUs and

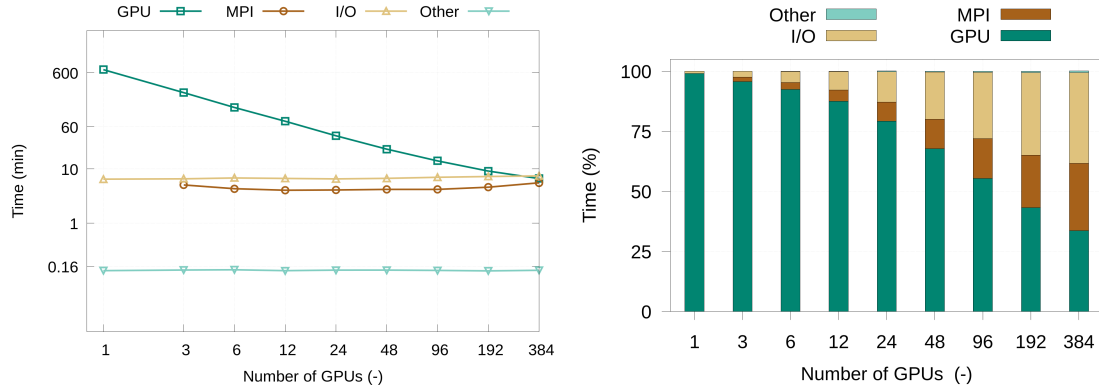


Figure 8: GPU, MPI, I/O and "other" time (left) and percentage (right) achieved by each configuration for the Harvey test problem

466 beyond. This fact suggests that the operational scale bottleneck is not with computation but on communication
 467 and I/O tasks. Parallel I/O should improve these results, and a more efficient communication strategy
 468 should be designed. These results motivate even larger test cases to leverage the massive parallelization in
 469 a substantial number of GPUs, paving the way to even larger temporal and spatial scales.

470 6. Conclusions and Perspectives

471 A new open-source 2D flood model, TRITON, is available to run on heterogeneous architectures using
 472 single and multiple CPUs and GPUs to enable hydraulic computations at large temporal and spatial scales.
 473 It consists of a simple input file structure (meshless, standard GIS formats and a configuration text file) and
 474 provides spatiotemporal information of water depths and velocities as output information.

475 Three test cases have been provided to demonstrate the TRITON capabilities. The accuracy, consistency
 476 and robustness of the scheme has been proved by means of the paraboloid test case. The importance of runoff
 477 capability in TRITON has been highlighted using a flood test case in northwest Georgia, US. Results with
 478 and without runoff have been compared against the 100-year flood extension provided by FEMA, revealing
 479 the benefit of considering the runoff capability for this sort of studies. Finally, the Hurricane Harvey test
 480 case has been used to show the performance on the scheme on different architectures: multi-CPU using
 481 OpenMP + MPI and multi-GPU with CUDA + MPI. The numerical experiments highlight the convenience
 482 of using the multi-GPU version against the multi-CPU, achieving a low runtime (less than 30 mins) for real-
 483 world configurations – a large spatial domain at 10m resolution and a 10-day hydrograph – and unlocking
 484 operational purposes at even larger spatial and temporal scales.

485 Although the runtimes are promising, the scalability of the multi-GPU version is nevertheless unsatis-
 486 factory for large number of GPUs. Many aspects are responsible for this: thread divergence for dry cells,
 487 poor load balancing between MPI ranks with a static decomposition, and principally the execution time
 488 consumed by I/O and MPI communication for a large scale problem, with respect to the computation time.
 489 Although these times (in the order of minutes each) are almost constant and does not depend on the number
 490 of GPUs, they represent a great percentage of the total runtime when trying to achieve operational purposes.
 491 Future perspectives are therefore aimed at designing optimized I/O parallel algorithms and exploring new
 492 communication techniques such as a 2D MPI decomposition or overlapping strategies. Particularly, the
 493 latter would considerably improve the performance as communication between subdomains would not be
 494 required to be every time step.

495 TRITON is under continuous development. Future planned improvement includes, besides new efficient
 496 I/O and communication algorithms, support for other GPU libraries such as OpenACC. All of these efforts
 497 will be regularly documented in the repository. We provide this baseline to enable the use of a fully 2D
 498 hydraulic model for new science questions such as uncertainty quantification or climate change problems
 499 beyond the studied temporal and spatial scales heretofore. The extension to very high resolution conti-

500 mental/global scale flood modeling will require multi-GPU computations so new algorithms are obviously
 501 planned to extend these capabilities to operational scales.

502 7. Acknowledgment

503 This research was supported by the US Air Force Numerical Weather Modeling Program. This research
 504 used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory,
 505 which is a US Department of Energy (DOE) Office of Science User Facility. Some of the co-authors are
 506 employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US DOE. Accordingly, the
 507 US government retains and the publisher, by accepting the article for publication, acknowledges that the
 508 US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the
 509 published form of this manuscript or allow others to do so, for US Government purposes. DOE will provide
 510 public access to these results of federally sponsored research in accordance with the DOE Public Access Plan
 511 (<http://energy.gov/downloads/doe-public-access-plan>).

512 A. Numerical Scheme

513 The derivation of the numerical scheme from (1) to (2) is detailed here. First, integrating the hyperbolic
 514 system of equations (1) in $\Omega_i \times [t^n, t^*]$, where Ω_i is referred to the i -th computational cell of the domain
 515 and $t^* = t^n + \Delta t$, and applying the Gauss theorem:

$$\int_{t^n}^{t^*} \int_{\Omega_i} \frac{\partial \mathbf{U}}{\partial t} dSdt + \int_{t^n}^{t^*} \int_{\Omega_i} (\nabla \cdot \mathbf{E}) dSdt = \int_{t^n}^{t^*} \int_{\Omega_i} \mathbf{S}_b dSdt \quad (8)$$

where $\mathbf{E} = (\mathbf{F}, \mathbf{G})$. Applying the Gauss theorem and replacing the contour integral by the sum across its
 four edges (Cartesian square grid of size Δx):

$$\int_{t^n}^{t^*} \int_{\Omega_i} \frac{\partial \mathbf{U}}{\partial t} dSdt + \int_{t^n}^{t^*} \sum_{k=1}^4 \mathbf{E} \mathbf{n} \Delta x dt = \int_{t^n}^{t^*} \int_{\Omega_i} \mathbf{S}_b dSdt \quad (9)$$

516 with $\mathbf{n} = (n_x, n_y)$ the outward normal direction. Assuming a piecewise discretization and a Roe's solver
 517 with an upwind discretization of fluxes and source terms, the updating of the conserved variables can be
 518 written in flux-difference splitting form as [26]:

$$\mathbf{U}_i^* = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x} \sum_{k=1}^4 (\delta \mathbf{E} \mathbf{n} - \mathbf{H}_b \mathbf{n})_k^{n,-} \quad (\delta \mathbf{E} \mathbf{n})_k^- = (\tilde{\mathbf{P}} \tilde{\Lambda}^- \tilde{\mathbf{A}})_k \quad (\mathbf{H}_b \mathbf{n})_k^- = (\tilde{\mathbf{P}} \frac{\tilde{\Lambda}^-}{\tilde{\Lambda}} \tilde{\mathbf{B}}_b)_k \quad (10)$$

519 The meaning of this expression is simple: the conserved variables $\mathbf{U} = (h, q_x, q_y)$ at each grid cell i will
 520 be updated according to the in-going contributions that come from its four edges k (east, north, west and
 521 south) shared by the four neighbouring cells j . A sketch of the scheme is shown in Figure 9(a).

522 Equation (10) includes the matrices and vectors coming from Roe's linearization: $\tilde{\mathbf{P}}$ is the matrix containing
 523 the three right eigenvectors $\tilde{\mathbf{e}}_k$ of the Jacobian Matrix, $\tilde{\Lambda}$ represents the diagonal matrix of the three
 524 eigenvalues $\tilde{\lambda}_k$ – the minus superscript denotes the upwind discretization – and $\tilde{\mathbf{A}}_k$ and $\tilde{\mathbf{B}}_{b_k}$ account for the
 525 three wave and source strengths respectively. With this, it is easy to derive equation (2) from (10). The

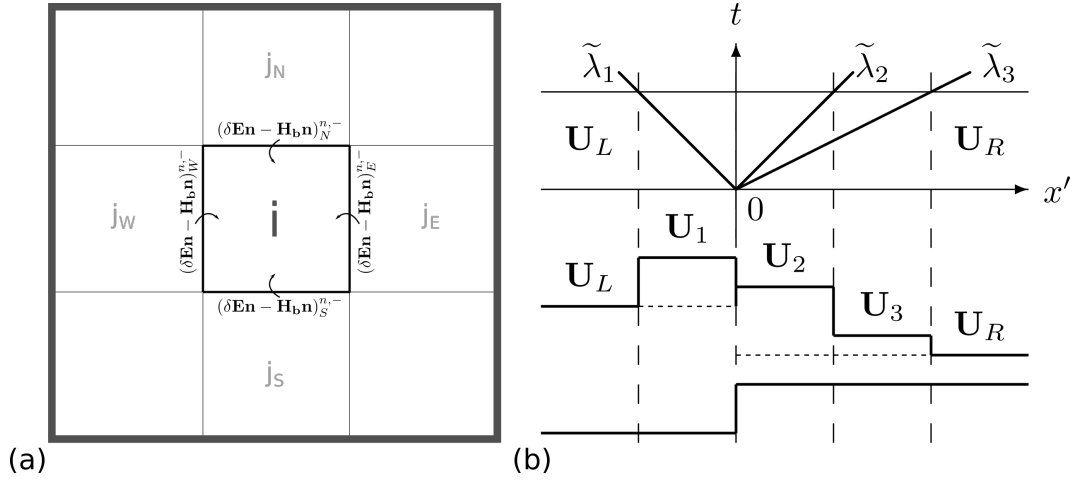


Figure 9: Stencil for the proposed numerical scheme (a) and approximate solution for $\tilde{\lambda}_1 < 0, \tilde{\lambda}_2, \tilde{\lambda}_3 > 0$ (b)

526 expressions for those matrices and vectors are detailed here:

$$\tilde{\mathbf{P}}_k = (\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, \tilde{\mathbf{e}}_3)_k = \begin{pmatrix} 1 & 0 & 1 \\ \tilde{u} - \tilde{c} n_x & -\tilde{c} n_y & \tilde{u} + \tilde{c} n_x \\ \tilde{v} - \tilde{c} n_y & \tilde{c} n_x & \tilde{v} + \tilde{c} n_y \end{pmatrix}_k$$

$$\tilde{\Lambda}_k = \begin{pmatrix} \tilde{\lambda}_1 & 0 & 0 \\ 0 & \tilde{\lambda}_2 & 0 \\ 0 & 0 & \tilde{\lambda}_3 \end{pmatrix}_k = \begin{pmatrix} \tilde{\mathbf{u}}\mathbf{n} - \tilde{c} & 0 & 0 \\ 0 & \tilde{\mathbf{u}}\mathbf{n} & 0 \\ 0 & 0 & \tilde{\mathbf{u}}\mathbf{n} + \tilde{c} \end{pmatrix}_k \quad \tilde{\Lambda}_k^- = \frac{\tilde{\Lambda}_k - |\tilde{\Lambda}_k|}{2}$$

$$\tilde{\mathbf{A}}_k = (\tilde{\mathbf{P}}^{-1} \delta \mathbf{U})_k = \begin{pmatrix} \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \\ \tilde{\alpha}_3 \end{pmatrix}_k = \begin{pmatrix} \frac{\delta h}{2} - \frac{\delta \mathbf{q}\mathbf{n} - \tilde{\mathbf{u}}\delta h}{2\tilde{c}} \\ \frac{1}{\tilde{c}} [(\delta q_y - \tilde{v}\delta h)n_x - (\delta q_x - \tilde{u}\delta h)n_y] \\ \frac{\delta h}{2} + \frac{\delta \mathbf{q}\mathbf{n} - \tilde{\mathbf{u}}\delta h}{2\tilde{c}} \end{pmatrix}_k \quad (11)$$

$$\tilde{\mathbf{u}}_k = (\tilde{u}, \tilde{v})_k = \left(\frac{u_i \sqrt{h_i} + u_j \sqrt{h_j}}{\sqrt{h_i} + \sqrt{h_j}}, \frac{v_i \sqrt{h_i} + v_j \sqrt{h_j}}{\sqrt{h_i} + \sqrt{h_j}} \right)_k \quad \tilde{c} = \sqrt{g \frac{h_i + h_j}{2}}$$

$$\delta h_k = h_j - h_i \quad \delta \mathbf{q}_k = (\delta q_x, \delta q_y)_k = (q_{x_j} - q_{x_i}, q_{y_j} - q_{y_i})_k$$

527 where $\mathbf{u} = (u, v) = (q_x/h, q_y/h)$ is the flow velocity.

528 The term $\tilde{\mathbf{B}}_{\mathbf{b}k}$ accounts for the bed slope source estimation. In this work, the integral formulation
529 proposed in [39] is adopted, leading to the following definition of $\tilde{\mathbf{B}}_{\mathbf{b}k}$:

$$\tilde{\mathbf{B}}_{\mathbf{b}k} = \begin{pmatrix} \tilde{\beta}_{b_1} \\ \tilde{\beta}_{b_2} \\ \tilde{\beta}_{b_3} \end{pmatrix}_k = \begin{pmatrix} \frac{g}{2\tilde{c}} \left(h_p - \frac{|\delta z'|}{2} \right) \delta z' \\ 0 \\ -\frac{g}{2\tilde{c}} \left(h_p - \frac{|\delta z'|}{2} \right) \delta z' \end{pmatrix}_k \quad h_p = \begin{cases} h_i & \delta z \geq 0 \\ h_j & \delta z < 0 \end{cases} \quad \delta z' = \begin{cases} h_i & \delta z \geq 0 \text{ and } h_i + z_i < z_j \\ -h_j & \delta z < 0 \text{ and } h_j + z_j < z_i \\ z_j - z_i & \text{otherwise} \end{cases}$$

Although this estimation is more accurate than the conventional ‘finite-difference-like’ approach, it can still lead to non-physical solutions when dealing with complex flows. When these problems arise, the conventional solution consists in reducing CFL and consequently the time step size, impacting directly on the performance of the computation. Another approach is adopted here, based on the augmented approach, which allows us to reconstruct the cell-averaged approximate solution with the objective of avoiding negative water depth values at time t^* . This can be particularly useful for the treatment of wet/dry fronts, a key factor for reliable 2D hydraulic models. In this work, the wet/dry treatment detailed in [17, 39] is implemented. Following this approach, the intermediate states of the Riemann problem are analyzed and, depending on their sign, the information is sent to the left or the right side of the discontinuity. This has been proved to be mass conservative and robust for any kind of flows [17, 39].

Besides the wet/dry fronts, stiff source terms can lead to negative water depth values, even in the presence of wet/wet problems -i.e., thin layers of water with strong bed discontinuities and high friction terms. In order to consider these situations in our numerical scheme, a similar technique of that followed in [39] is adopted here. However, as the time step is computed before the source terms estimation, a new limitation based on the final integration –rather than zero the intermediate states– is proposed in this work. As detailed in [39], only wet/wet subcritical cases are analyzed. Let us denote L and R the left and right states of a discontinuity and assume $\tilde{\lambda}_2 > 0$ –the derivation does not change for $\tilde{\lambda}_2 < 0$ due to the properties of the intermediate states for the water depth in the augmented approach–. The numerical solution at time t^* should guarantee $h_L^* \geq 0$ and $h_R^* \geq 0$. Then, according to Figure 9(b), the following inequations must be fulfilled:

$$h_L \left(\frac{\Delta x}{2} - |\tilde{\lambda}_1| \Delta t \right) + |\tilde{\lambda}_1| \Delta t h_1 \geq 0 \quad \tilde{\lambda}_2 \Delta t h_2 + (\tilde{\lambda}_3 - \tilde{\lambda}_2) \Delta t h_3 + h_R \left(\frac{\Delta x}{2} - \tilde{\lambda}_3 \Delta t \right) \geq 0 \quad (13)$$

In agreement with the definitions of the intermediate states [53, 39]:

$$h_1 = h_L + \tilde{\alpha}_1 - \frac{\tilde{\beta}_{b_1}}{\tilde{\lambda}_1} \quad h_2 = h_3 = h_R - \tilde{\alpha}_3 - \frac{\tilde{\beta}_{b_1}}{\tilde{\lambda}_3} \quad (14)$$

and doing some simple algebraic manipulations, the following restrictions are derived for $\tilde{\beta}_{b_1}$ –and consequently for $\tilde{\beta}_{b_3}$ –:

$$\tilde{\beta}_{min} \leq \tilde{\beta}_{b_1} \leq \tilde{\beta}_{max} \quad \tilde{\beta}_{min} = -h_L \frac{\Delta x}{2\Delta t} + \tilde{\alpha}_1 \tilde{\lambda}_1 \quad \tilde{\beta}_{max} = \frac{\Delta x}{2\Delta t} h_R - \tilde{\lambda}_3 \tilde{\alpha}_3 \quad \tilde{\beta}_{b_3} = -\tilde{\beta}_{b_1} \quad (15)$$

These conditions are implemented in TRITON, minimizing the appearance of negative water depths. Additionally, this scheme needs an entropy correction for transonic rarefactions hence the Harten-Hyman entropy fix is used here. As this approach is based on decomposing the existing jump into two new jumps, a special emphasis should be put in the source term split [39].

Two additional assumptions are considered. First, a cell is considered dry if its water depth is below 10^{-12} . Second, velocities are set to zero for water depths below a tolerance. This value depends on the characteristics of the problem study (spatial scales and roughness mainly). As a physical explanation for this parameter, dissipation terms (in practical applications with realistic roughness values) dominate at this scale over convective and inertial terms. Therefore, it is not arbitrary to model this phenomenon as a tolerance under which velocities are considered null. A value of 10^{-3} is suggested for real-world scenarios as a general recommendation, although it can be also modified in TRITON for every configuration. The scheme has been proved to be well-balanced, robust and reliable for flows under various conditions, according to Section 5.

References

- 566
- 567 [1] Alfonso, L., Mukolwe, M.M., Di Baldassarre, G., 2016. Probabilistic flood maps to support decision-
568 making: Mapping the value of information. *Water Resources Research* 52, 1026–1043. URL:
569 <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015WR017378>, doi:10.1002/2015WR017378,
570 arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2015WR017378>.
- 571 [2] de la Asunción, M., Castro, M.J., Fernández-Nieto, E., Mantas, J.M., Acosta, S.O., González-Vida, J.M., 2013. Effi-
572 cient gpu implementation of a two waves tvd-waf method for the two-dimensional one layer shallow water system on
573 structured meshes. *Computers & Fluids* 80, 441 – 452. URL: <http://www.sciencedirect.com/science/article/pii/S0045793012000217>, doi:<https://doi.org/10.1016/j.compfluid.2012.01.012>. selected contributions of the 23rd Interna-
574 tional Conference on Parallel Fluid Dynamics ParCFD2011.
- 575 [3] Aureli, F., Prost, F., Vacondio, R., Dazzi, S., Ferrari, A., 2020. A gpu-accelerated shallow-water scheme for surface runoff
576 simulations. *Water* 12, 637.
- 577 [4] Bates, P.D., Horritt, M.S., Fewtrell, T.J., 2010. A simple inertial formulation of the shallow water equations for efficient
578 two-dimensional flood inundation modelling. *Journal of Hydrology* 387, 33 – 45. URL: <http://www.sciencedirect.com/science/article/pii/S0022169410001538>, doi:<https://doi.org/10.1016/j.jhydrol.2010.03.027>.
- 579 [5] Berger, M.J., George, D.L., LeVeque, R.J., Mandli, K.T., 2011. The geoclaw software for depth-averaged flows with
580 adaptive refinement. *Advances in Water Resources* 34, 1195 – 1206. URL: <http://www.sciencedirect.com/science/article/pii/S0309170811000480>, doi:<https://doi.org/10.1016/j.advwatres.2011.02.016>. new Computational Meth-
581 ods and Software Tools.
- 582 [6] Brodtkorb, A.R., Sætra, M.L., Altinakar, M., 2012. Efficient shallow water simulations on gpus: Implementation, visu-
583 alization, verification, and validation. *Computers & Fluids* 55, 1 – 12. URL: <http://www.sciencedirect.com/science/article/pii/S0045793011003185>, doi:<https://doi.org/10.1016/j.compfluid.2011.10.012>.
- 584 [7] Castro, M.J., Ortega, S., de la Asunción, M., Mantas, J.M., Gallardo, J.M., 2011. Gpu computing for shallow wa-
585 ter flow simulation based on finite volume schemes. *Comptes Rendus Mécanique* 339, 165 – 184. URL: <http://www.sciencedirect.com/science/article/pii/S1631072110002147>, doi:<https://doi.org/10.1016/j.crme.2010.12.004>.
586 high Performance Computing.
- 587 [8] Caviedes-Voullieme, D., Morales-Hernández, M., Juez, C., Lacasta, A., García-Navarro, P., 2017. Two-dimensional nu-
588 merical simulation of bed-load transport of a finite-depth sediment layer: Applications to channel flushing. *Journal of*
589 *Hydraulic Engineering* 143, 04017034. doi:10.1061/(ASCE)HY.1943-7900.0001337.
- 590 [9] David, C.H., Maidment, D.R., Niu, G.Y., Yang, Z.L., Habets, F., Eijkhout, V., 2011. River network routing on the
591 nhdplus dataset. *Journal of Hydrometeorology* 12, 913–934. URL: <https://doi.org/10.1175/2011JHM1345.1>, doi:10.
592 1175/2011JHM1345.1, arXiv:<https://doi.org/10.1175/2011JHM1345.1>.
- 593 [10] Dazzi, S., Vacondio, R., Mignosa, P., 2019. Integration of a levee breach erosion model in a
594 gpu-accelerated 2d shallow water equations code. *Water Resources Research* 55, 682–702. URL:
595 <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018WR023826>, doi:10.1029/2018WR023826,
596 arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018WR023826>.
- 597 [11] Dazzi, S., Vacondio, R., Mignosa, P., 2020. Internal boundary conditions for a gpu-accelerated 2d shallow water model:
598 Implementation and applications. *Advances in Water Resources* 137, 103525. URL: <http://www.sciencedirect.com/science/article/pii/S0309170819309157>, doi:<https://doi.org/10.1016/j.advwatres.2020.103525>.
- 599 [12] Dazzi, S., Vacondio, R., Palù, A.D., Mignosa, P., 2018. A local time stepping algorithm for gpu-accelerated 2d shallow
600 water models. *Advances in Water Resources* 111, 274 – 288. URL: <http://www.sciencedirect.com/science/article/pii/S0309170817304311>, doi:<https://doi.org/10.1016/j.advwatres.2017.11.023>.
- 601 [13] Delestre, O., Darboux, F., James, F., Lucas, C., Laguerre, C., Cordier, S., 2017. Fullswof: Full shallow-water equations
602 for overland flow. *Journal of Open Source Software* 2, 448. URL: <https://doi.org/10.21105/joss.00448>, doi:10.21105/
603 joss.00448.
- 604 [14] Delestre, O., Lucas, C., Ksinant, P.A., Darboux, F., Laguerre, C., Vo, T.N.T., James, F., Cordier, S., 2013. Swashes:
605 a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for*
606 *Numerical Methods in Fluids* 72, 269–300. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/flid.3741>, doi:10.
607 1002/flid.3741, arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/flid.3741>.
- 608 [15] Deltares, 2014. Delft3D-Flow, Simulation of multi-dimensional hydrodynamic flows and transport phenomena, including
609 sediments. User Manual, Version 3.15.34158.
- 610 [16] Dullo, T.T., Gangrade, S., Morales-Hernandez, M., Sharif, M.B., Kao, S.C., Kalyanapu, A., Ghafoor, S., Evans, K.J.,
611 2021. Simulation of hurricane harvey flood event through coupled hydrologic-hydraulic models: Challenges and next steps.
612 Accepted in *Journal of Flood Risk Management* .
- 613 [17] Echeverribar, I., Morales-Hernández, M., Brufau, P., García-Navarro, P., 2019a. 2d numerical simulation of unsteady flows
614 for large scale floods prediction in real time. *Advances in Water Resources* 134, 103444. URL: <http://www.sciencedirect.com/science/article/pii/S0309170819304786>, doi:<https://doi.org/10.1016/j.advwatres.2019.103444>.
- 615 [18] Echeverribar, I., Morales-Hernández, M., Brufau, P., García-Navarro, P., 2019b. Use of internal boundary conditions for
616 levees representation: application to river flood management. *Environmental Fluid Mechanics* , 1–19.
- 617 [19] Evers, M., Jonoski, A., Maksimović, v., Lange, L., Ochoa Rodriguez, S., Teklesadik, A., Cortes Arevalo, J., Almoradie,
618 A., Eduardo Simões, N., Wang, L., Makropoulos, C., 2012. Collaborative modelling for active involvement of stakeholders
619 in urban flood risk management. *Natural Hazards and Earth System Sciences* 12, 2821–2842. URL: [https://nhess.
620 copernicus.org/articles/12/2821/2012/](https://nhess.copernicus.org/articles/12/2821/2012/), doi:10.5194/nhess-12-2821-2012.

- 629 [20] Gangrade, S., Kao, S.C., Dullo, T.T., Kalyanapu, A.J., Preston, B.L., 2019. Ensemble-based flood vulnerability assessment
630 for probable maximum flood in a changing environment. *Journal of Hydrology* 576, 342–355.
- 631 [21] Gangrade, S., Kao, S.C., Naz, B.S., Rastogi, D., Ashfaq, M., Singh, N., Preston, B.L., 2018. Sensitivity of probable
632 maximum flood in a changing environment. *Water Resources Research* 54, 3913–3936.
- 633 [22] García-Feal, O., Cea, L., González-Cao, J., Domínguez, J., Gómez-Gesteira, M., 2020. Iberwq: A gpu accelerated tool for
634 2d water quality modeling in rivers and estuaries. *Water* 12, 413.
- 635 [23] García-Feal, O., González-Cao, J., Gómez-Gesteira, M., Cea, L., Domínguez, J., Formella, A., 2018. An accelerated tool
636 for flood modelling based on iber. *Water* 10, 1459.
- 637 [24] García-Navarro, P., Murillo, J., Fernández-Pato, J., Echeverribar, I., Morales-Hernández, M., 2019. The shallow water
638 equations and their application to realistic cases. *Environmental Fluid Mechanics* URL: <https://doi.org/10.1007/s10652-018-09657-7>, doi:10.1007/s10652-018-09657-7.
- 640 [25] Hodges, B.R., Liu, F., 2019. Timescale interpolation and no-neighbour discretization for a 1d finite-volume saint-venant
641 solver. *Journal of Hydraulic Research* 0, 1–17. URL: <https://doi.org/10.1080/00221686.2019.1671510>, doi:10.1080/
642 00221686.2019.1671510, arXiv:<https://doi.org/10.1080/00221686.2019.1671510>.
- 643 [26] Hubbard, M., García-Navarro, P., 2000. Flux difference splitting and the balancing of source terms and flux gradi-
644 ents. *Journal of Computational Physics* 165, 89 – 125. URL: <http://www.sciencedirect.com/science/article/pii/S0021999100966038>, doi:<https://doi.org/10.1006/jcph.2000.6603>.
- 645 [27] Juez, C., Lacasta, A., Murillo, J., García-Navarro, P., 2016. An efficient gpu implementation for a faster simulation of
646 unsteady bed-load transport. *Journal of Hydraulic Research* 54, 275–288. URL: <https://doi.org/10.1080/00221686.2016.1143042>,
648 2016.1143042, doi:10.1080/00221686.2016.1143042, arXiv:<https://doi.org/10.1080/00221686.2016.1143042>.
- 649 [28] Kalyanapu, A.J., Shankar, S., Pardyjak, E.R., Judi, D.R., Burian, S.J., 2011. Assessment of gpu computational enhance-
650 ment to a 2d flood model. *Environmental Modelling & Software* 26, 1009 – 1016. URL: <http://www.sciencedirect.com/science/article/pii/S1364815211000582>, doi:<https://doi.org/10.1016/j.envsoft.2011.02.014>.
- 651 [29] Kesserwani, G., Ayog, J.L., Bau, D., 2018. Discontinuous galerkin formulation for 2d hydrodynamic modelling: Trade-offs
652 between theoretical complexity and practical convenience. *Computer Methods in Applied Mechanics and Engineering* 342,
653 710 – 741. URL: <http://www.sciencedirect.com/science/article/pii/S004578251830389X>, doi:<https://doi.org/10.1016/j.cma.2018.08.003>.
- 654 [30] Lacasta, A., Juez, C., Murillo, J., García-Navarro, P., 2015a. An efficient solution for hazardous geophysical flows
655 simulation using gpus. *Computers & Geosciences* 78, 63 – 72. URL: <http://www.sciencedirect.com/science/article/pii/S0098300415000321>, doi:<https://doi.org/10.1016/j.cageo.2015.02.010>.
- 656 [31] Lacasta, A., Morales-Hernández, M., Murillo, J., García-Navarro, P., 2014. An optimized gpu implementation of a 2d
657 free surface simulation model on unstructured meshes. *Advances in Engineering Software* 78, 1 – 15. URL: <http://www.sciencedirect.com/science/article/pii/S0965997814001331>, doi:<https://doi.org/10.1016/j.advengsoft.2014.08.007>.
- 658 [32] Lacasta, A., Morales-Hernández, M., Murillo, J., García-Navarro, P., 2015b. Gpu implementation of the 2d shallow water
659 equations for the simulation of rainfall/runoff events. *Environmental Earth Sciences* 74, 7295–7305.
- 660 [33] Liang, X., Lettenmaier, D.P., Wood, E.F., Burges, S.J., 1994. A simple hydrologically based model of land surface
661 water and energy fluxes for general circulation models. *Journal of Geophysical Research: Atmospheres* 99,
662 14415–14428. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/94JD00483>, doi:10.1029/94JD00483,
663 arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/94JD00483>.
- 664 [34] Liu, Y., Maidment, D., Tarboton, D., Zheng, X., Yıldırım, A., Sazib, N., Wang, S., 2016. A cybergis approach to
665 generating high-resolution height above nearest drainage (hand) raster for national flood mapping. doi:10.13140/RG.2.2.
666 24234.41925/1.
- 667 [35] Liu, Y.Y., Maidment, D.R., Tarboton, D.G., Zheng, X., Wang, S., 2018. A cybergis integration and computation
668 framework for high-resolution continental-scale flood inundation mapping. *JAWRA Journal of the American Water
669 Resources Association* 54, 770–784. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1752-1688.12660>,
670 doi:10.1111/1752-1688.12660, arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/1752-1688.12660>.
- 671 [36] Madsen, H., Lawrence, D., Lang, M., Martinkova, M., Kjeldsen, T., 2014. Review of trend analysis and climate change
672 projections of extreme precipitation and floods in europe. *Journal of Hydrology* 519, 3634 – 3650. URL: <http://www.sciencedirect.com/science/article/pii/S0022169414008889>, doi:<https://doi.org/10.1016/j.jhydrol.2014.11.003>.
- 673 [37] Maskrey, S.A., Mount, N.J., Thorne, C.R., Dryden, I., 2016. Participatory modelling for stakeholder involvement
674 in the development of flood risk management intervention options. *Environmental Modelling & Software* 82, 275 –
675 294. URL: <http://www.sciencedirect.com/science/article/pii/S1364815216301220>, doi:<https://doi.org/10.1016/j.envsoft.2016.04.027>.
- 676 [38] Morales-Hernández, M., Sharif, M.B., Gangrade, S., Dullo, T.T., Kao, S.C., Kalyanapu, A., Ghafoor, S.K.,
677 Evans, K.J., Madadi-Kandjani, E., Hodges, B.R., 2020. High-performance computing in water resources hydrody-
678 namics. *Journal of Hydroinformatics* URL: <https://doi.org/10.2166/hydro.2020.163>, doi:10.2166/hydro.2020.163,
679 arXiv:<https://iwaponline.com/jh/article-pdf/doi/10.2166/hydro.2020.163/652305/jh2020163.pdf>, jh2020163.
- 680 [39] Murillo, J., García-Navarro, P., 2010. Weak solutions for partial differential equations with source terms: Application
681 to the shallow water equations. *Journal of Computational Physics* 229, 4327 – 4368. URL: <http://www.sciencedirect.com/science/article/pii/S0021999110000896>, doi:<https://doi.org/10.1016/j.jcp.2010.02.016>.
- 682 [40] Murillo, J., García-Navarro, P., 2012. Wave riemann description of friction terms in unsteady shallow flows: Application
683 to water and mud/debris floods. *Journal of Computational Physics* 231, 1963 – 2001. URL: <http://www.sciencedirect.com>.

- com/science/article/pii/S0021999111006693, doi:<https://doi.org/10.1016/j.jcp.2011.11.014>.
- [41] Murillo, J., Navas-Montilla, A., 2016. A comprehensive explanation and exercise of the source terms in hyperbolic systems using Roe type solutions. application to the 1d-2d shallow water equations. *Advances in Water Resources* 98, 70 – 96. URL: <http://www.sciencedirect.com/science/article/pii/S0309170816305917>, doi:<https://doi.org/10.1016/j.advwatres.2016.10.019>.
- [42] Neal, J., Schumann, G., Bates, P., 2012. A subgrid channel model for simulating river hydraulics and floodplain inundation over large and data sparse areas. *Water Resources Research* 48. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2012WR012514>, doi:10.1029/2012WR012514, arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2012WR012514>.
- [43] Neal, J.C., Fewtrell, T.J., Bates, P.D., Wright, N.G., 2010. A comparison of three parallelisation methods for 2d flood inundation models. *Environmental Modelling & Software* 25, 398 – 411. URL: <http://www.sciencedirect.com/science/article/pii/S1364815209002965>, doi:<https://doi.org/10.1016/j.envsoft.2009.11.007>.
- [44] Petaccia, G., Leporati, F., Torti, E., 2016. Openmp and cuda simulations of sella zerbino dam break on unstructured grids. *Computational Geosciences* 20, 1123–1132.
- [45] Qin, X., LeVeque, R.J., Motley, M.R., 2019. Accelerating an adaptive mesh refinement code for depth-averaged flows using gpus. *Journal of Advances in Modeling Earth Systems* 11, 2606–2628. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001635>, doi:10.1029/2019MS001635, arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019MS001635>.
- [46] Sætra, M.L., Brodtkorb, A.R., 2012. Shallow water simulations on multiple gpus, in: Jónasson, K. (Ed.), *Applied Parallel and Scientific Computing*, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 56–66.
- [47] Sanders, B.F., 2008. Integration of a shallow water model with a local time step. *Journal of Hydraulic Research* 46, 466–475. URL: <https://doi.org/10.3826/jhr.2008.3243>, doi:10.3826/jhr.2008.3243, arXiv:<https://doi.org/10.3826/jhr.2008.3243>.
- [48] Sanders, B.F., Schubert, J.E., 2019. Primo: Parallel raster inundation model. *Advances in Water Resources* 126, 79 – 95. URL: <http://www.sciencedirect.com/science/article/pii/S0309170818308698>, doi:<https://doi.org/10.1016/j.advwatres.2019.02.007>.
- [49] Sanders, B.F., Schubert, J.E., Detwiler, R.L., 2010. Parbrezo: A parallel, unstructured grid, godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. *Advances in Water Resources* 33, 1456 – 1467. URL: <http://www.sciencedirect.com/science/article/pii/S0309170810001429>, doi:<https://doi.org/10.1016/j.advwatres.2010.07.007>.
- [50] Sanders, B.F., Schubert, J.E., Goodrich, K.A., Houston, D., Feldman, D.L., Basolo, V., Luke, A., Boudreau, D., Karlin, B., Cheung, W., Contreras, S., Reyes, A., Eguiarte, A., Serrano, K., Allaire, M., Moftakhari, H., AghaKouchak, A., Matthew, R.A., 2020. Collaborative modeling with fine-resolution data enhances flood awareness, minimizes differences in flood perception, and produces actionable flood maps. *Earth's Future* 8, e2019EF001391. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019EF001391>, doi:10.1029/2019EF001391, arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019EF001391>. e2019EF001391 2019EF001391.
- [51] Sharif, M.B., Ghafoor, S.K., Hines, T., Morales-Hernández, M., Evans, K.J., Kao, S.C., Kalyanapu, A., Dullo, T.T., Gangrade, S., 2020. Performance evaluation of a two-dimensional flood model on heterogeneous high-performance computing architectures. *PASC '20: Proceedings of the Platform for Advanced Scientific Computing Conference*.
- [52] Spiekermann, R., Kienberger, S., Norton, J., Briones, F., Weichselgartner, J., 2015. The disaster-knowledge matrix – reframing and evaluating the knowledge challenges in disaster risk reduction. *International Journal of Disaster Risk Reduction* 13, 96 – 108. URL: <http://www.sciencedirect.com/science/article/pii/S2212420915300091>, doi:<https://doi.org/10.1016/j.ijdr.2015.05.002>.
- [53] Toro, E., 2013. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer Berlin Heidelberg.
- [54] Turchetto, M., Palù, A.D., Vacondio, R., 2020. A general design for a scalable mpi-gpu multi-resolution 2d numerical solver. *IEEE Transactions on Parallel and Distributed Systems* 31, 1036–1047. doi:10.1109/TPDS.2019.2961909.
- [55] Turchetto, M., Vacondio, R., Paù, A.D., 2018. Multi-gpu implementation of 2d shallow water equation code with block uniform quad-tree grids, in: Loggia, G.L., Freni, G., Puleo, V., Marchis, M.D. (Eds.), *HIC 2018. 13th International Conference on Hydroinformatics, EasyChair*. pp. 2105–2111. URL: <https://easychair.org/publications/paper/9Qpz>, doi:10.29007/xwc.
- [56] Vacondio, R., Palù, A.D., Ferrari, A., Mignosa, P., Aureli, F., Dazzi, S., 2017. A non-uniform efficient grid type for gpu-parallel shallow water equations models. *Environmental Modelling & Software* 88, 119 – 137. URL: <http://www.sciencedirect.com/science/article/pii/S1364815216309252>, doi:<https://doi.org/10.1016/j.envsoft.2016.11.012>.
- [57] Vacondio, R., Palù, A.D., Mignosa, P., 2014. Gpu-enhanced finite volume shallow water solver for fast flood simulations. *Environmental Modelling & Software* 57, 60 – 75. URL: <http://www.sciencedirect.com/science/article/pii/S136481521400053X>, doi:<https://doi.org/10.1016/j.envsoft.2014.02.003>.
- [58] Viñas, M., Lobeiras, J., Fraguera, B., Arenaz, M., Amor, M., García, J., Castro, M., Doallo, R., 2013. A multi-gpu shallow-water simulation with transport of contaminants. *Concurrency and Computation: Practice and Experience* 25, 1153–1169. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.2917>, doi:10.1002/cpe.2917, arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.2917>.
- [59] Wang, C., Hou, J., Miller, D., Brown, I., Jiang, Y., 2019. Flood risk management in sponge cities: The role of integrated simulation and 3d visualization. *International Journal of Disaster Risk Reduction* 39, 101139. URL: <http://www>.

- 755 [sciencedirect.com/science/article/pii/S2212420918313803](https://doi.org/10.1016/j.ijdr.2019.101139), doi:<https://doi.org/10.1016/j.ijdr.2019.101139>.
- 756 [60] on Water Data, I.A.C., 1982. Guidelines for determining flood flow frequency, hydrology subcommittee bulletin 17b. Water
757 Resources Research .
- 758 [61] Weng, Z., Strazdins, P.E., 2014. Acceleration of a python-based tsunami modelling application via cuda and openhmp, in:
759 2014 IEEE International Parallel Distributed Processing Symposium Workshops, pp. 1275–1284. doi:10.1109/IPDPSW.
760 2014.143.
- 761 [62] Wing, O.E., Bates, P.D., Sampson, C.C., Smith, A.M., Johnson, K.A., Erickson, T.A., 2017. Validation of a 30 m
762 resolution flood hazard model of the conterminous united states. Water Resources Research 53, 7968–7986.
- 763 [63] Wing, O.E.J., Bates, P.D., Smith, A.M., Sampson, C.C., Johnson, K.A., Fargione, J., Morefield, P., 2018. Estimates
764 of present and future flood risk in the conterminous united states. Environmental Research Letters 13, 034023. URL:
765 <https://doi.org/10.1088/1748-9326/aaac65>, doi:10.1088/1748-9326/aaac65.
- 766 [64] Xia, X., Liang, Q., 2018. A new efficient implicit scheme for discretising the stiff friction terms in the shallow water
767 equations. Advances in Water Resources 117, 87 – 97. URL: <http://www.sciencedirect.com/science/article/pii/S0309170818302124>, doi:<https://doi.org/10.1016/j.advwatres.2018.05.004>.
- 769 [65] Xia, X., Liang, Q., Ming, X., 2019. A full-scale fluvial flood modelling framework based on a high-performance integrated
770 hydrodynamic modelling system (hipims). Advances in Water Resources 132, 103392. URL: <http://www.sciencedirect.com/science/article/pii/S030917081930243X>, doi:<https://doi.org/10.1016/j.advwatres.2019.103392>.
- 771 [66] Xie, J., Chen, H., Liao, Z., Gu, X., Zhu, D., Zhang, J., 2017. An integrated assessment of urban flooding mitigation strate-
772 gies for robust decision making. Environmental Modelling & Software 95, 143 – 155. URL: <http://www.sciencedirect.com/science/article/pii/S1364815216303449>, doi:<https://doi.org/10.1016/j.envsoft.2017.06.027>.
- 773
- 774