Journal of
Hydroinformatics

© 2023 The Authors

Journal of Hydroinformatics Vol 25 No 5, 1690 doi: 10.2166/hydro.2023.012

Investigating optimal 2D hydrodynamic modeling of a recent flash flood in a steep Norwegian river using high-performance computing

Adina Moraru[a,*], Nils Rüther[a,b] and Oddbjørn Bruland[a]

[a] Department of Civil and Environmental Engineering, Norwegian University of Science and Technology (NTNU), Trondheim N-7491, Norway
[b] Chair of Hydraulic Engineering, Technical University of Munich (TUM), Munich DE-80333, Germany
*Corresponding author. E-mail: adina.moraru@ntnu.no

AM, 0000-0003-2701-5272; NR, 0000-0002-7667-5966; OB, 0000-0002-1131-8976

ABSTRACT

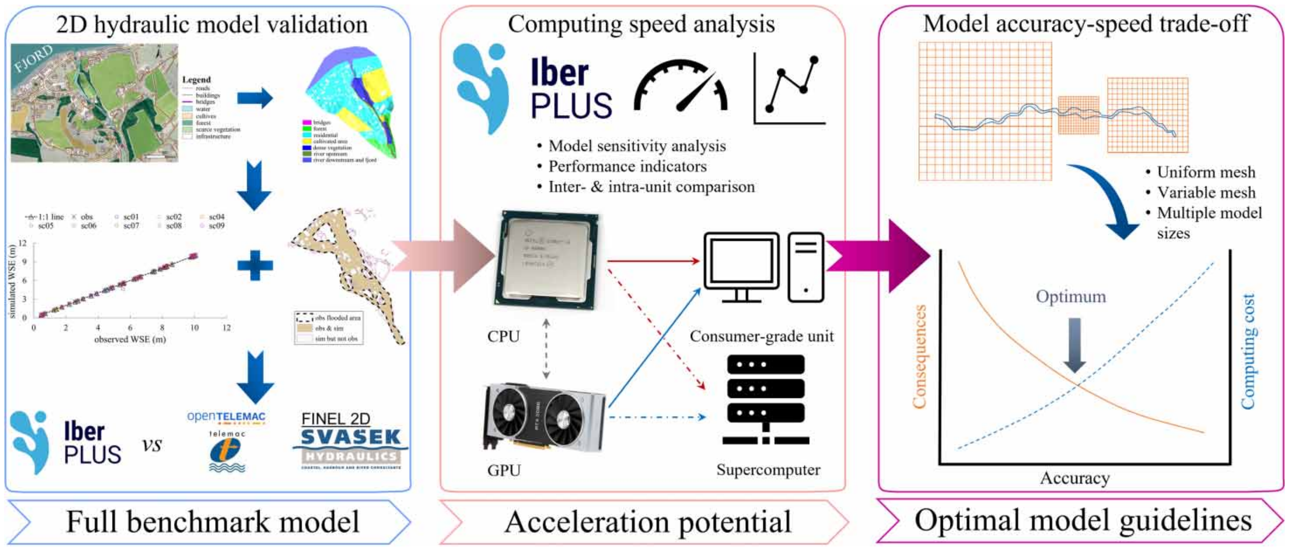
Efficient flood risk assessment and communication are essential for responding to increasingly recurrent flash floods. However, access to high-end data center computing is limited for stakeholders. This study evaluates the accuracy-speed trade-off of a hydraulic model by (i) assessing the potential acceleration of high-performance computing in PCs versus server-CPUs and GPUs, (ii) examining computing time evaluation and prediction indicators, and (iii) identifying variables controlling the computing time and their impact on the 2D hydrodynamic models' accuracy using an actual flash flood event as a benchmark. GPU-computing is found to be 130× and 55× faster than standard and parallelized CPU-computing, respectively, saving up to 99.5% of the computing time. The model's number of elements had the most significant impact, with $<$150,000 cells showing the best accuracy-speed trade-off. Using a PC equipped with a GPU enables almost real-time hydrodynamic information, democratizing flood data and facilitating interactive flood risk analysis.

Key words: 2D hydrodynamic modeling, flash flood modeling, high-performance GPU-computing, optimized hydraulic simulation, sensitivity analysis, steep rivers

HIGHLIGHTS

- Fast and reliable flood-predictive tools minimize flood damage in steep rivers.
- Parallelization methods in PCs can provide up to 130× faster results and save 99.5% of the computing time.
- Graphic cards in PCs can be as fast as data center processing units.
- The optimal precision-speed trade-off is achieved 5× faster for variable-sized meshes than for uniform-sized meshes.

GRAPHICAL ABSTRACT



1. INTRODUCTION

Europe is currently experiencing intensified flood seasonality (Blöschl *et al.* 2020). Moreover, civilians and stakeholders are unprepared for increasingly frequent, severe, and spatially extended floods. More than half of the 1,500 flood events reported in Europe over the last 150 years were flash floods, and only 10% of all the events had data available on the total flooded area (Paprotny *et al.* 2018). Citizen science and real-time social media coverage of flood events expand and speed up environmental data availability (Ansell & Dalla Valle 2022). The impact of a recent flash flood in a steep Norwegian river was witnessed and documented audio-visually (Bruland 2020; Moraru *et al.* 2021). The desire for improved decision-making and reducing flood risk has driven research on hydraulic modeling techniques capable of providing nearly real-time hydraulic information. These models could be integrated into state-of-the-art flood risk communication tools, including realistic visualization and interactive serious gaming in augmented and virtual reality (AR/VR). Such tools enhance understanding, awareness of natural hazards and enable the testing of emergency response plans and training without real exposure to hazards.

In Norway, the annual cost due to flooding in the last four decades is over 126 million NOK (or 12 million €) in the private sector alone (FinansNorge 2023). This cost estimate could be close to 1 billion NOK (or 100 million €) when accounting for road repairs, river restoration, and similar flood-related damage repairs. Flood-predictive and warning tools are very valuable when they provide reliable information with a lead time that allows for minimizing flood damage and, for instance, provides enough time for an evacuation (Fernández-Nóvoa *et al.* 2020). Compromises in model accuracy are oftentimes required for a gain in response time. Describing optimally the critical locations in a river and refining the model at these locations alone could ensure a suitable model accuracy and a significant gain in computing speed. More efficient hydraulic models will allow for more rivers to be analyzed and thus more cost-efficient measures to be implemented. Recent studies show that legacy models lack the necessary implementations to take advantage of the parallelism available on current hardware (e.g. Morales-Hernández *et al.* 2020), which hinders their use in applications such as real-time flood forecasting, where the scheduled model runs for periodic update, interactive visualization for a more user-friendly and effective flood risk communication or in large batch-processing tasks (e.g. Monte Carlo analysis). On the other hand, new parallel implementations of hydrologic and hydraulic modules have achieved speed-ups of up to one hundred times faster than their standard versions in mountain areas (Moraru *et al.* 2020) and up to a thousand times faster in urban areas (Buttinger-Kreuzhuber *et al.* 2022). Graphics processing units (GPUs) have up to several thousand computing cores allocated in one single processing unit (PU), which can be parallelized. For reference, it would require the best server multi-CPUs (central processing units) available in the market to reach a similar number of cores working together in one single task. GPUs are available for scientific computing and many GPU-based hydraulic models focused on solving the shallow water equations (SWEs) have been developed in the last decade. For instance, García-Feal *et al.* (2018) translated the code of the 2D numerical model Iber (Bladé *et al.* 2014) into Iber + , a code that allows parallelization both on multi-core CPU (hereafter 'parallelized version') and GPU. A comprehensive list of pioneer 2D GPU-based models oriented to flood applications is available in Kalyanapu *et al.* (2011) and Moraru *et al.* (2020). Several existing 2D numerical models that can compute simultaneously on multiple threads/cores, either on CPU or GPU, are mathematically based on Roe's upwind approximation (Roe 1981, 1986) – a finite volume method (FVM) Riemann solver – e.g. Lacasta *et al.* (2014), García-Feal *et al.* (2018), Echeverribar *et al.* (2019), Caviedes-Voullième *et al.* (2023), among others. González-Cao *et al.* (2019) developed a coupled hydrologic–hydraulic warning system based on HEC-HMS and Iber+ that simulated both models for 3 days in less than 1 h altogether. For reference, a 3× smaller hydraulic model (i.e. 74k elements instead of the 200k elements of the coupled model; Table S1 in Supplementary Material), simulated for 1 day using MIKE21 FM took 7 h to compute on the parallelized CPU configuration (Ejigu 2020; D. K. Ejigu personal communication, 16 February 2021). That is, a 3× smaller model, run for 1/3 of the time, took 7× longer to compute in MIKE21 than in HEC-HMS and Iber + , and it excluded the hydrologic model. The early warning system in González-Cao *et al.* (2019) was based on Iber + , which achieved a speed-up of up to 94× in benchmarked cases simulated with a mid-range GPU (García-Feal *et al.* 2018).

The numerical modeling presented in this study was carried out in Iber and Iber + , which are hydraulic models that solve 2D SWEs using an unstructured FVM. The detailed 2D depth-averaged SWEs that are solved in Iber can be found in García-Feal *et al.* (2018). This numerical model was preferred over other promising 2D models able to use HPC (high-performance computing) because, unlike many models used in research, it was adapted to be readable, usable and it is continuously maintained. Iber is very user-friendly and freeware, hence, very accessible for independent researchers and small stakeholders. Moreover, Iber's performance has been benchmarked and shown to be fast and precise when numerically modeling physical scenarios of different dimensions, as well as the 2D numerical modeling of a large river catchment (García-Feal *et al.* 2018; Cea *et al.* 2020).

Far too little is known about the use of GPU-based HPC in personal computers (PCs) for fast and affordable flood modeling that could increase and improve early data availability. The primary objective of the current study is to provide model specifications to facilitate the transition from a scientific to an operational tool in flood risk studies. This aim was investigated by analyzing the trade-off of the accuracy and the computing speed of the hydrodynamic model for the flood event herewith presented as a case study for illustration and benchmarking. The hydrodynamic model was calibrated and validated (see chapter 2, Figure 1(a)). The model resolution and mesh type used, together with the numerical approach and field observations are expected to provide a full 2D hydrodynamic model with enough accuracy for flood risk management, i.e. flood extent is adequately replicated (Liu *et al.* 2019). Although the flash flood event affecting Storelva river (Utvik, western Norway) in 2017 was highly driven by erosion and deposition processes, it is out of the scope of this study to complement former studies with complex morphodynamic information that would increase the computational demand of the numerical model (Beven 2012). Once the model accuracy was adequate, the current study aimed to (i) evaluate the potential acceleration achieved by HPC in PCs and their limitations when using numerical modeling for flood risk management, (ii) examine evaluation and prediction indicators for computing time in 2D numerical modeling of floods, (iii) identify variables controlling the computing time of 2D numerical models and propose an accuracy-speed trade-off. To achieve the first aim, the 2017 flood was studied with the Iber + GPU-based 2D hydrodynamic model, and its performance and stability were compared to several standard and parallelized CPU-based numerical models (see section 3.1, Figure 1(b) and 1(c)). The PUs used in this study represent a vast array of low- to high-range privately accessible PUs. Due to the difficult access to high-end data center (or server) PUs for municipalities and small stakeholders, these were excluded from the current comparison. The second aim was tackled by carrying out the statistical analysis of the data obtained during the study of aim (i) and evaluating the observations using performance indicators (see section 3.2, Figure 1(d)). To achieve the third aim, multiple model set-ups were tested and contrasted with the performance of models available in the literature. The sensitivity analysis, which estimates the uncertainty of all input and output variables in the hydrodynamic model (Beven *et al.* 2015; Dimitriadis *et al.* 2016), enabled the characterization of an optimal accuracy-speed curve (see section 3.3, Figure 1(e)–1(g)).

2. CASE STUDY

The hydrogeomorphic background of Storelva in Utvik and limitations identified in other modeling studies are described in section 2.1 (Figure 1(a)). The hydrodynamic model calibration and set-up are described in section 2.2, and its validation and hydraulic results are shown in section 2.3. The numerical model stability of Iber(+) is tested in section 2.4. The resulting model was then implemented in HPC testing and analysis.

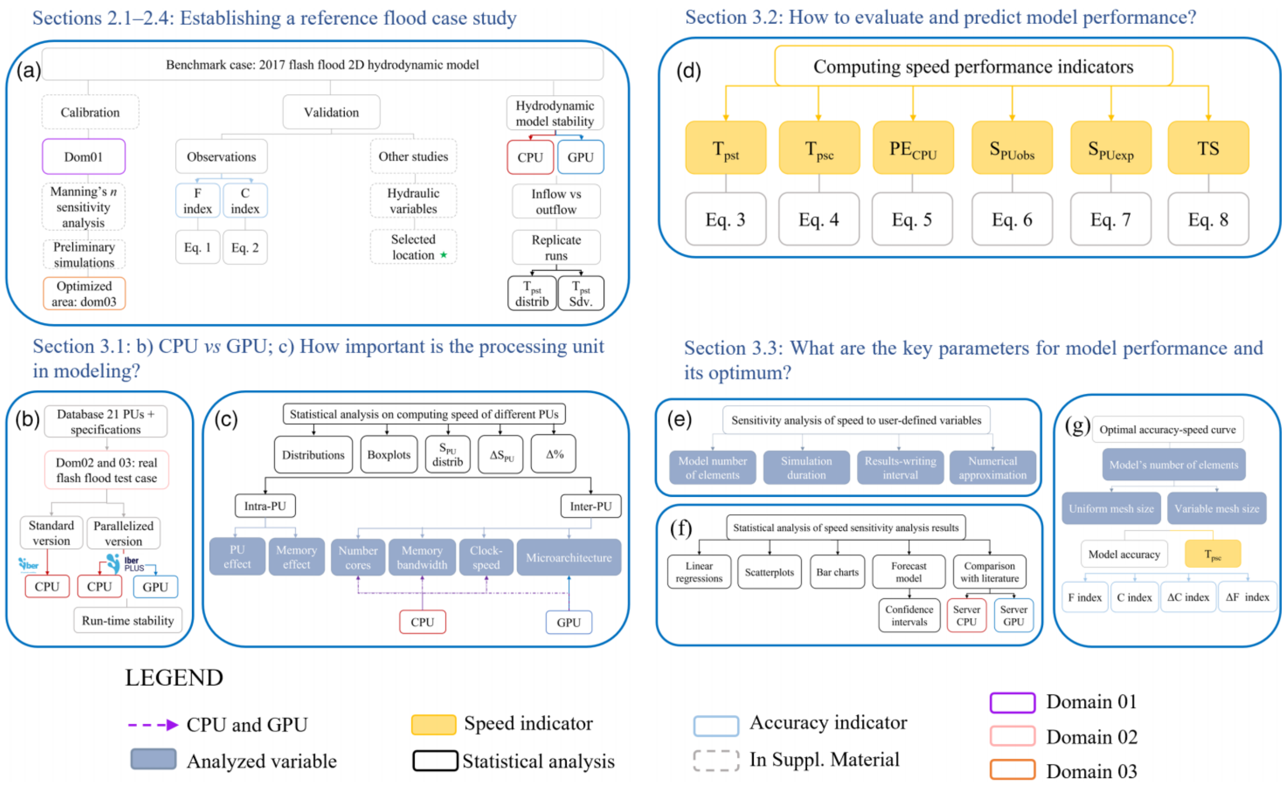


Figure 1 | Summary of the workflow followed in this study and the sections where each partial workflow is described. Symbology is indicated in the legend. PU, processing unit, CPU, central processing unit, GPU, graphics processing unit, S_{PU}, speed-up of PU, T_{pst}, computing time per step and thread, T_{psc}, computing time per step and cell, PE_{CPU}, parallelization efficiency of CPU, S_{PUobs}, observed S_{PU} in this study, S_{PUexp}, expected S_{PU}, TS, time-saving ratio, Distrib., distribution, Sdv, standard deviation. Equations are available in the text.

2.1. Flash flood in Storelva river (West Norway) in 2017

Storelva river, located in Utvik, West of Norway (Figure 2) was flooded on 24 July 2017, when ca. 4 h of heavy rainfall turned the 1.6 m^3/s average discharge in a peak discharge of ca. 200 m^3/s, exceeding the 200 years return period (Bruland 2020). The river is in a steep valley, with an average longitudinal channel slope of 10.52% in the 775 m-long study reach. The main land uses include cultivated land, scarce vegetation, forest, and a lot of Quaternary fine material (Figure 2). A detailed analysis of the dynamics during the flood is available in Moraru *et al.* (2021), where the extension of the flooded area, the preferential flow zone (PFZ; Figure 3(b)) and the main flow paths were mapped.

The hydrodynamics and morphodynamics of the 2017 flash flood were previously modeled with FINEL2D (TU Delft & Svašek Hydraulics) by Dam (2018) and with Telemac-Mascaret 2D (Électricité de France) by Pavlíček & Bruland (2019). Both studies aimed to reproduce as accurately as possible the observed preferential flow paths and newly created channel, including an analysis of erosion and deposition processes. Son (2020) created the training dataset for a machine learning (ML) model of the 2017 flash flood in Iber. In this latter study, the ML model surpassed the computing speed of the hydrodynamic models, yet it only marginally matched the non-calibrated Iber models it was validated against. The offset between ML and 2D hydraulic models is expected, as ML is not based on the complex physics that 2D models are. ML is still a good indicator of model performance (Rozos *et al.* 2022). An overview of the models and discharges considered by other studies for the 2017 flood is shown in Table S2. This study uses the same flood hydrograph and river reach modeled by Pavlíček & Bruland (2019) and Son (2020). These models have not been calibrated. The lack of on-site measurements of mobilized sediment masses during the flood makes the refinement and validation of the morphodynamic models challenging. So far, most of the flow paths and locations where the water left the channel were identified with similar success in both hydrodynamic and morphodynamic models (Pavlíček & Bruland 2019), thus the present study was carried out using only a hydrodynamic model (Figure 1(a)).

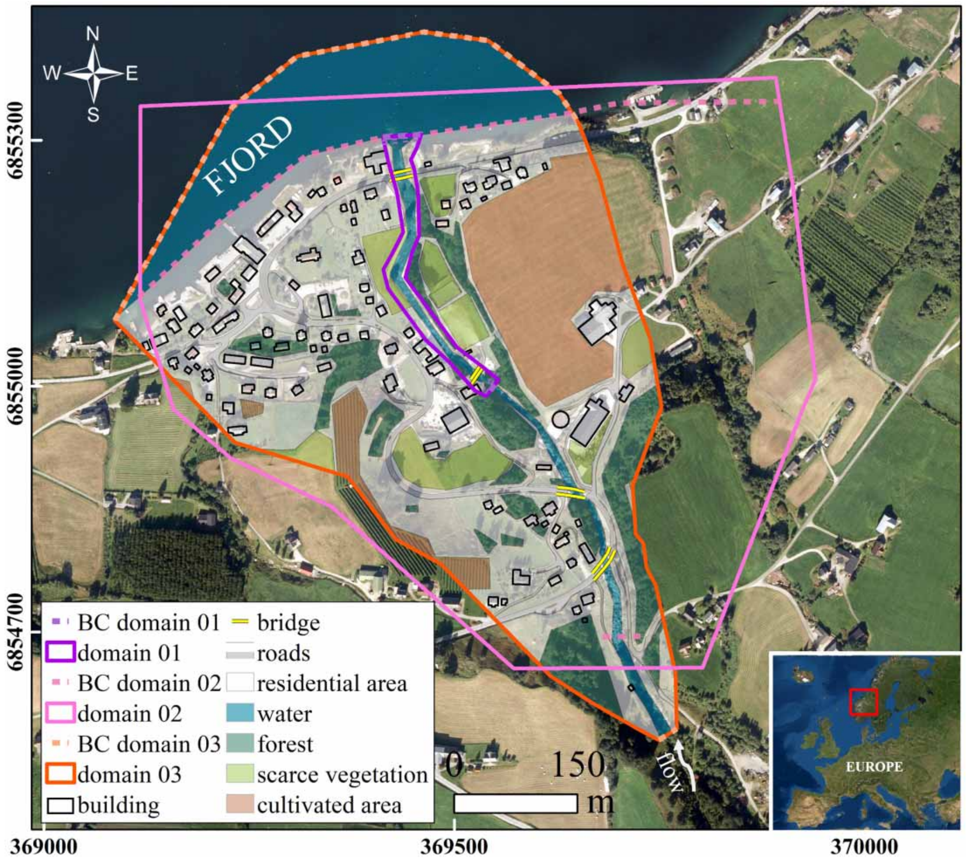


Figure 2 | Land use map and computing domain extensions for the flood models in Storelva river in Utvik. Domain01 was used for calibration and sensitivity analysis, whereas domains02 and 03 were used for model performance comparison. The context map is in the lower right corner. The orthophoto is in ETRS89, UTM33 (Norwegian Mapping Authority 2015).

The information on Storelva river can be enriched by calibrating a hydrodynamic model with recent field observations obtained post-flood and validating the model using the mapped observed flooded extent from Moraru *et al.* (2021). There is no roughness sensitivity analysis for the downstream reach of Storelva river, while the upstream reach was calibrated by Bruland (2020). All the aforementioned aspects are tackled hereafter, together with (i) an evaluation of the sensitivity to roughness variation of the Iber model and (ii) the analysis of the hydrodynamic performance of Iber + when simulating extreme events in steep slopes – where hydrodynamics are complex and flow is mostly supercritical.

2.2. 2D hydrodynamic model set-up

The DEM surveyed by the Norwegian Mapping Authority (2013) was used to build the computational domain on the 775 m-long reach most downstream and nearest to the local population affected by the flood. The hydrological modeling of the unsteady inflow hydrograph for the 2D hydrodynamic model of the 2017 flood is described by Bruland (2020). Although the 191 m^3/s peak discharge is reached within 4 h, a 25h-long simulation was run to ensure model stability. The event was simulated using the first order of Roe's approximation and a wet-dry limit of 0.01 m for two different domains: one with ca. 165,000 cells (domain02 in Figure 2) and another with ca. 120,000 cells (domain03 in Figure 2). The results were exported every 1 min (3,880,438 steps) and 5 min (2,140,820 steps), respectively.

The extension of the computational domain (domain02 in Figures 2 and 1(a)) was refined and reduced to ca. 0.34 km^2 (domain03 in Figure 2) based on preliminary simulations and inundation area analysis. The flood model had a mesh size of 2 m in the river channel and inundation platforms near the banks and of 4 m in areas in the domain furthest from the river channel and in the fjord, which follows the more refined mesh guidelines of previous modeling studies (Table S2). The model included four bridges (yellow lines in Figure 2) treated as single lines facing upstream, while the buildings

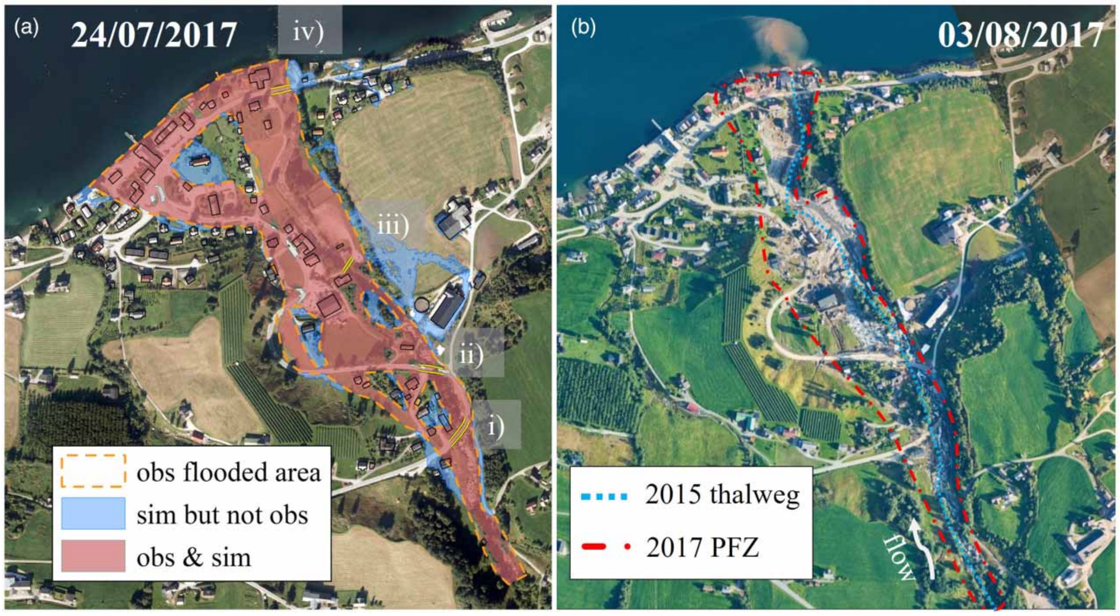


Figure 3 | (a) Map of simulated flood extent (red and blue) contrasted with the observed flooded area during the 2017 flash flood (dashed orange polygon; Moraru *et al.* 2021); (b) preferential flow zone (PFZ) observed 10 days after the flood (dashed red line) and original river thalweg (dotted blue line). The scale is the same as in Figure 2. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.012.

were excluded from the computed domain (black polygons in Figure 2; Bellos & Tsakiris 2015). The outlet boundary condition (BC) was considered supercritical/critical and assigned by the fjord (i.e. 0 masl, BC domain03 in Figure 2). Figure 2 shows the location of the BCs and the land use map used to assign the variable roughness coefficients for the 2D hydrodynamic model. The Courant–Friedrichs–Lewy (CFL; Courant *et al.* 1967) condition was set to 0.45. After carrying out a sensitivity analysis for Manning roughness coefficients in the downstream reach and calibrating the model with field observations of water levels and georeferenced position of water edge lines on both banks (see section 1.1 in Supplementary Material; Table S3), the final roughness values used in the 2017 flash flood model were bridges: 0.020, forest: 0.120, residential: 0.150, cultivated area: 0.080, dense vegetation: 0.180, river: 0.065 upstream, and 0.075 downstream and in the fjord (Fig. S1). The simulated spatial extent of the flood was processed in GIS and compared to flood documentation, which enabled validating the hydrodynamic model (see section 2.3).

2.3. Validation of the 2017 flash flood model

The river roughness calibration, validation and roughness sensitivity analysis carried out for this case study is summarized in section 1.1 in the Supplementary Material. Moraru *et al.* (2021) mapped the maximum flood extent (Figure 3(a)), PFZ (Figure 3(b)), and main flow paths observed during the flood. The maximum estimated water depth was 5.88 m (near bridge ii, Figure 3(a)), when the flow reached a maximum velocity of 10.72 m/s. The 2D hydrodynamic model shows flooding near all four bridges in the domain, as well as in the lower areas of both sides of the floodplain, which fits the flood documentation.

The simulated flood extent (Figure 3(a)) matched the observed area by 72% according to the *F* index (Equation (1), which shows the over- and underestimations of the model for both dry and flooded areas) and by 87% according to the *C* index (Equation (2), which shows the total flooded area successfully modeled without an emphasis on the geographical overlap, cf. Casas-Mulet *et al.* 2015; Liu *et al.* 2019). The simulated flooded area is matching similarly well the traces of the PFZ observed days after the flood (Figure 3(b)).

$$F(\%) = 100 \cdot \left(\frac{A_{\mathrm{om}}}{A_{\mathrm{o}} + A_{\mathrm{m}} - A_{\mathrm{om}}} \right) \tag{1}$$

$$C(\%) = 100 \cdot \left(\frac{A_{\mathrm{om}}}{A_{\mathrm{o}}} \right) \tag{2}$$

in which A_o is the observed flooded area, A_m is the model-simulated flooded area, and A_{om} is the flooded area that is both observed and simulated.

2.4. Hydrodynamic model stability

The stability of the numerical model was studied by estimating the evolution of the output discharge throughout the simulation time in the CPU- (red dashed line) and GPU-based (blue dotted line) hydrodynamic models (Figure 4 and 1(a)). The model is very stable throughout the simulation time in both computing modalities, with few exceptions when the discharge raises drastically (Figure 4, arrows).

The stability of both CPU and GPU-computing time was analyzed for 25 replicate runs (cf. Liu *et al.* 2018), obtaining the distribution of T_{pst} (section 3.2) and its standard deviation for both CPU and GPU (Figure 1(b) and 1(c)). The results obtained from the analytical procedures presented in chapters 2 and 3 will be described in the next chapter.

3. METHODS

3.1. Benchmarking parallel CPU- and high-performance GPU-computing

To address the usability of HPC in PCs, i.e. research aim (i), a database was created to examine the impact of PU-related variables on the computing time of 2D hydrodynamic models (Figure 1(b)). The database included 21 different PUs (Table 1), their technical specifications, total computing time, and performance indicators derived from it (see section 3.2) for two different computing scenarios for the same benchmark case study (see section 2.2). The technical specifications considered were the number of physical and logical threads, microarchitecture, base frequency, memory, memory bandwidth, and type of device (Figure 1(c)). The total computing time includes the load case time, writing results time, and simulation computing time for the parallel version, as well as the initial and final computing time for the standard version. The effect of each of the PU's technical specifications was calculated. The dataset was then reclassified and represented in boxplots together with the relative difference between classes (in terms of speed-up or %).

Aiming to analyze the impact of the selected PU as well as the run-time stability of such PU, Iber, and Iber + were used to assess the computing time that different low- to high-end CPUs and GPUs could achieve on both notebooks and desktop computers (Figure 1(b)) for the 2D hydrodynamic model described in chapter 2. The model was computed both on the standard single-thread CPU version (i.e. 1 core, 'standard' hereafter) and on the parallelized multi-thread CPU version (i.e. half of the threads available, such as 4 for the i7-family, 3 for the i5-family or 8 for the i9-family, as read on the simulation log in Iber+).

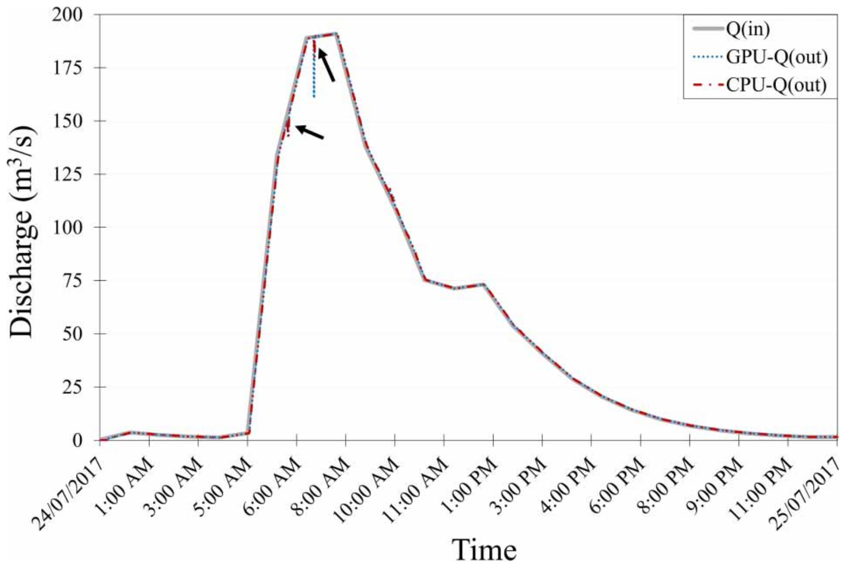


Figure 4 | Inflow BC (Q_{in}; Bruland 2020) and outflowing discharge (Q_{out}) for CPU-computing (i.e. i7-7700-b multi-thread; red dashed line) and GPU-computing (i.e. NVIDIA Quadro P620; blue dotted line) for the 2017 flash flood. Arrows point to discrepancies between the outputs of the CPU and GPU models. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.012.

Table 1 | CPUs and GPUs used to compute hydrodynamic simulations for the case study

Processing unit	Launch year	Microarchitecture	N° cores (threads)	Base frequency	Memory	Memory bandwidth (GB/s)	Device
i7-4710HQ	2014	NA	4 (8)	2.50 GHz	8 GB DDR3	25.6	Notebook
i7-8650U -a	2017	NA	4 (8)	1.90 GHz	16 GB DDR4	34.1	Notebook
i7-8650U -b	2017	NA	4 (8)	1.90 GHz	16 GB DDR4	34.1	Notebook
i7-8650U -c	2017	NA	4 (8)	1.90 GHz	16 GB DDR4	34.1	Notebook
i7-7700 -a	2017	NA	4 (8)	3.60 GHz	32 GB DDR4	35.8	Desktop
i7-7700 -b	2017	NA	4 (8)	3.60 GHz	64 GB DDR4	35.8	Desktop
i7-7700 -c	2017	NA	4 (8)	3.60 GHz	32 GB DDR4	35.8	Desktop
i5-9600K	2018	NA	6 (6)	3.70 GHz	32 GB DDR4	41.6	Desktop
i9-9880H	2019	NA	8 (16)	2.30 GHz	16 GB DDR4	41.8	Notebook
i7-10750H	2020	NA	6 (12)	2.60 GHz	32 GB DDR4	45.8	Notebook
NVIDIA GeForce GTX 860M	2014	Maxwell	640	1,029 MHz	2 GB GDDR5	80.2	Notebook
NVIDIA GeForce GTX 750	2015	Maxwell	512	1,020 MHz	1 GB GDDR5	80.2	Desktop
NVIDIA GeForce GTX 940M	2015	Maxwell	512	1,020 MHz	2 GB GDDR3	14.4	Notebook
NVIDIA GeForce MX 130	2017	Maxwell	384	1,122 MHz	2 GB GDDR5	40.1	Notebook
NVIDIA GeForce GTX 1060	2016	Pascal	1,152	1,506 MHz	3 GB GDDR5	192.2	Desktop
NVIDIA GeForce GTX 1050	2018	Pascal	768	1,392 MHz	3 GB GDDR5	84.1	Desktop
NVIDIA GeForce GTX 1050 MaxQ	2018	Pascal	640	1,189 MHz	4 GB GDDR5	112.1	Notebook
NVIDIA Quadro P620	2018	Pascal	512	1,252 MHz	2 GB GDDR5	80.1	Desktop
NVIDIA GeForce RTX 2080Ti	2018	Turing	4,352	1,350 MHz	11 GB GDDR6	616.0	Desktop
NVIDIA GeForce GTX 1650Ti	2020	Turing	896	1,410 MHz	4 GB GDDR6	192.0	Notebook
NVIDIA GeForce RTX 3080	2020	Ampere	8,704	1,440 MHz	10 GB GDDR6X	760.3	Desktop

Different units are denoted with letters '-a', '-b', and '-c'. Base frequency represents base clock speed. NA, not applicable.

To assess the inter- and intra- deviation of the CPU results, the model was run on different units of the exact same type (i.e. i7-8650U-a/b/c). This test was replicated to discern the effect of the CPU memory (i.e. i7-7700-a/b/c).

An identical numerical model was run on 11 different NVIDIA GPUs, covering the most recent microarchitectures available at consumer level (i.e. Kepler and Fermi were excluded due to their obsolescence), as well as a wide range of numbers of cores and clock speeds (i.e. base frequencies; Table 1). More recent GPUs not only have a higher number of cores available for parallel computing but also a higher base frequency for their cores and a larger and faster memory for the intermediate results to be written on. The cores of a GPU cannot be separated; hence, the case study was always run on all the GPU cores available.

3.2. Computing performance indicators

The database served to examine the computing time through statistical model performance indicators (Figure 1(c); research aim (ii). The performance of the PUs was evaluated in terms of:

(i) Average time per step and thread (T_{pst}, Equation (3), modified after García-Feal *et al.* 2018),

$$T_{\mathrm{pst}} = \frac{T}{n_{\mathrm{threads}} \times n_{\mathrm{steps}}} \tag{3}$$

where T is the computing time, n_{threads} is the number of threads used, and n_{steps} is the total number of time steps. The units of the T_{pst} are milliseconds per time step per thread (ms/ts $\times$ thread, 'MIST' hereafter). The number of steps used to calculate the T_{pst} depended on the model size (see section 2.2).

(ii) Average time per step and cell (T_{psc}, Equation (4)),

$$T_{\mathrm{psc}} = \frac{T}{n_{\mathrm{threads}} \times n_{\mathrm{cells}}} \tag{4}$$

where n_{cells} is the number of cells of a model. The units of T_{psc} are milliseconds per time step per cell (ms/ts × cell).

(iii) Parallelization efficiency of the CPU (PE, Equation (5)) reflects how shared the computational task is between the different threads, where PE values of 100% mean that all the threads are engaged in the computation and PE values of $1/n_{\mathrm{threads}}$ mean that only one thread carries out the task,

$$\mathrm{PE}_{\mathrm{cpu}}\ (\%) = \frac{T_{\mathrm{CPUst}}}{n_{\mathrm{threads}} \times T_{\mathrm{CPUmt}}} \tag{5}$$

where T_{CPUst} and T_{CPUmt} are the computing time of single- and multi-thread, respectively.

(iv) Observed speed-up ratio (S_{Puobs}, Equation (6)),

$$S_{\mathrm{PUobs}} = \frac{T_{\mathrm{CPUref}}}{T_{\mathrm{PU}}} \tag{6}$$

where T_{PU} and T_{CPUref} are the computing time (s) for the evaluated PU and the reference CPU, respectively. S_{PUobs} (or S_{PU} hereafter) is non-dimensional and was calculated based on both the single- and multi-thread CPU models.

(v) Expected speed-up ratio (S_{PUexp}, Equation (7)) relative to a defined CPU. S_{PUexp} has been previously described as a potential proxy for S_{PUobs} in Tomczak *et al.* (2013),

$$S_{\mathrm{PUexp}} = \frac{\mathrm{MB}_{\mathrm{PU}}}{\mathrm{MB}_{\mathrm{CPUref}}} \tag{7}$$

where S_{Puexp} is the expected speed-up ratio, $\mathrm{MB}_{\mathrm{PU}}$ and $\mathrm{MB}_{\mathrm{CPUref}}$ are memory bandwidth (GB/s) for the evaluated PU and the reference CPU, respectively. S_{Puexp} is non-dimensional.

(vi) Time-saving ratio (TS, Equation (8)) of the GPU (Liu *et al.* 2018). The GPU-based model was used as a reference for the calculation of the TS.

$$\mathrm{TS}\ (\%) = \frac{(T_{\mathrm{CPU}} - T_{\mathrm{GPU}})}{T_{\mathrm{CPU}}} \tag{8}$$

3.3. Sensitivity analysis of computing speed to key variables and optimal accuracy–speed curve

To investigate the third aim of this study, i.e. identifying what variables control the computing time of 2D hydrodynamic models, a comprehensive literature search was conducted. Table S1 in the Supplementary Material shows an overview of model characteristics, computing times, and speed-ups achieved in recent studies using unstructured-mesh models. It was noted that the computing time (or run-time) of a numerical model is dependent on multiple variables, such as the order of accuracy of the numerical approximation used (Bermúdez *et al.* 1998) and the model's total number of elements (i.e. model size; Lacasta *et al.* 2014; Dimitriadis *et al.* 2016).

A series of sensitivity analyses were carried out to evaluate the effect of user-specified model parameters identified as relevant in literature, such as the total number of elements in the model, total simulation time, numerical approximation, and the interval at which results were written, on the computing time on CPU and GPU (Figure 1(e), Table 2). In this study, the model size refers to the number of cells, where small: $N < 100{,}000$ cells, and large: $N > 500{,}000$ cells (Lacasta *et al.* 2014; Echeverribar *et al.* 2018). The effect of the model size on the computing speed was evaluated both as absolute and normalized values, i.e. T_{psc} (Equation (4)). The simulation times tested were short, i.e. max. 24 h, consistent with the nature of a flash flood and supporting the aim of finding a fast and precise model. The sensitivity analysis focused on domain01 (Figure 2) and a few selected PUs. The statistics used were limited to simple linear regressions, scatter plots, and their trendlines, as well as bar charts (Figure 1(f)).

Table 2 | Configurations for sensitivity analyses and effect on computing time

Model size effect on computing time					*($t = 3$ h, exported every 5 min)*					
Mesh size (m)	0.25	0.5	1	2	4					
N° Elements	653,302	162,734	40,672	10,081	2,495					
Simulation time effect on computing time					*(653k, exported every 1 min)*					
Total simulation time (h)	3	12	24							
Exporting interval effect on computing time					*($t = 3$ h, 162k)*					
Interval duration (s)	3	30	300		3,600					
Numerical scheme effect on computing time					*($t = 3$ h, exported every 5 min, 162k)*					
Approximation	First order	Second order								
Model size effect on precision and computing time				*($t = 25$ h, exported every 5 min, first order approximation)*						
Scenario ID	Sc01	Sc02	Sc03	Sc04	Sc05	Sc06	Sc07	Sc08	Sc09	Sc10
N° Elements	44,679	57,246	71,856	132,088	163,398	178,594	594,611	719,606	2,890,549	11,601,898

All the models were run using the second-order solution unless indicated otherwise.

All tests for the computing speed sensitivity analyses were carried out on two GPUs (i.e. desktop RTX3080, notebook GTX1650Ti) and one CPU (desktop i5-9600K), except for the exporting interval, which was carried out on RTX3080 and i5-9600K (Table 2). For instance, to analyze the effect of the model number of elements, the steady flow model was run for 3 h (n.b. the steady condition is reached after 15 min), and results were exported every 5 min, with homogeneous mesh resolutions that resulted in different model sizes up to 653k elements. Then, the computing speed was forecasted for models up to 1.3 million elements with a confidence interval of 80 and 95% (Figure 1(f)). The availability of field observations for steady flow enabled running simulations of variable duration. This is relevant as the total number of time steps, which was used to estimate the T_{pst} (Equation (3)) that every PU needs to compute, is dependent on the simulation time for a given model number of elements.

The effect of the simulation time was analyzed on a model of 653k elements, and exporting results every 1 min. Another interesting parameter in the model set-up that was tested for its effect was the exporting (or writing) results interval, which was compared by writing results at constant intervals for a model of 162k elements simulated for 3 h. To assess the effect of the numerical approximation used in the computing time, a model of 162k elements was simulated for 3 h, and the results were exported every 5 min (Table 2).

Furthermore, the dataset obtained from this sensitivity analysis and CPU *versus* GPU benchmarking was further analyzed in the context of other modeling studies (Table S1) for this or other case studies where sufficient information allowed comparing the potential performance of PCs- and server-CPUs and GPUs (Figure 1(f)).

The effect of the mesh size on the accuracy of the hydraulic model was evaluated to provide the optimal model configuration to achieve both accuracy and speed in flood analysis. The F and C indices (Equations (1) and (2)) and the incremental accuracy and computing speed (in terms of T_{psc}) between scenarios were estimated for fully calibrated models ranging between 44,000 and 11.6 million cells (Figure 1(g)). The mesh size was homogeneous in tests sc01, sc06, and sc08–sc10, and heterogeneous in the rest of the tests. The tests were run on the desktop GPU RTX3080.

The first-order, faster solving, solution was used for the computing time comparison in different PUs. However, the calibration of the hydrodynamic model and sensitivity analyses were carried out using the second order, more accurate, approximation. An optimal model accuracy was ensured before investigating model speed optimization for each of the steps described above. All models presented in this study are based on an unstructured mesh unless indicated otherwise.

4. RESULTS

4.1. CPU-computing

This section comprehends the outcome of CPU-computing with a constant model set-up, using the total computing time and T_{pst} as indicators. The analysis included: (i) the variability of parallelized CPU-computing time classified by type of device, (ii) the performance of standard- and parallelized CPU configurations, (iii) intra-PU analysis, (iv) the stability test for replicate CPU runtimes, (v) the effect of the CPU memory, and (vi) the effect of the PU's base frequency on the total computing time.

The overview of testing the domains of 165k and 120k elements on different PUs is shown in Figure 5. The parallelized CPU version was on average 3.3× faster than the standard CPU version for all the tested PUs (Figure 5(a), values above boxplots). The computing time was shorter and had a lower variability for the smaller model for both computing modalities (Table S4). The fastest CPU tested for the standard version (i.e. Intel Core i7-10750H) was 1.7–2× faster, i.e. in terms of T_{pst}, than the slowest CPU tested for the same version (i.e. Intel Core i7-4710HQ). Likewise, the fastest CPU tested for the parallelized version (i.e. Intel Core i9-9880H) tended to be 4.5–6× faster than the slowest CPU tested (i.e. Intel Core i7-8650U). Moreover, the deviation of the total population's CPU-based computing time was large, i.e. 18–23% to 37–41% for standard- and parallelized computing, respectively, for both models (Figure 5(a)).

Furthermore, a preliminary intra-unit analysis for both the standard and parallelized CPU showed that, although the parallelized version was on average 2.85× faster, it had a larger deviation from the average total computing time, i.e. 9.5–10.5% to 19–21% for single- and multi-thread computing, respectively (Figure 5(b)). The CPU-computing times for replicate runs were less stable for the smaller and faster model (Table S4, Figure 6, red lines). When replicating the intra-unit CPU analysis to discern the effect of the memory of the CPU, no significant differences were observed (not shown).

The intra-unit variation could not be tested on GPU; however, desktop CPUs were 70–80% faster and more stable for both computing domains (Figure 5(c)). The base frequency of the PU (Table 1) alone did not explain the variability in the total computing speed or T_{pst} observed for the PUs analyzed for either of the computational domains (not shown).

4.2. GPU-computing

This section comprehends the outcome of GPU-computing, using computing time and T_{pst} as indicators. The analysis included: (i) computing speed for different GPUs, (ii) dispersion of GPU-computing time depending on the type of device, (iii) the effect of the GPU's microarchitecture on the T_{pst}, and (iv) a stability test for replicate GPU runtimes.

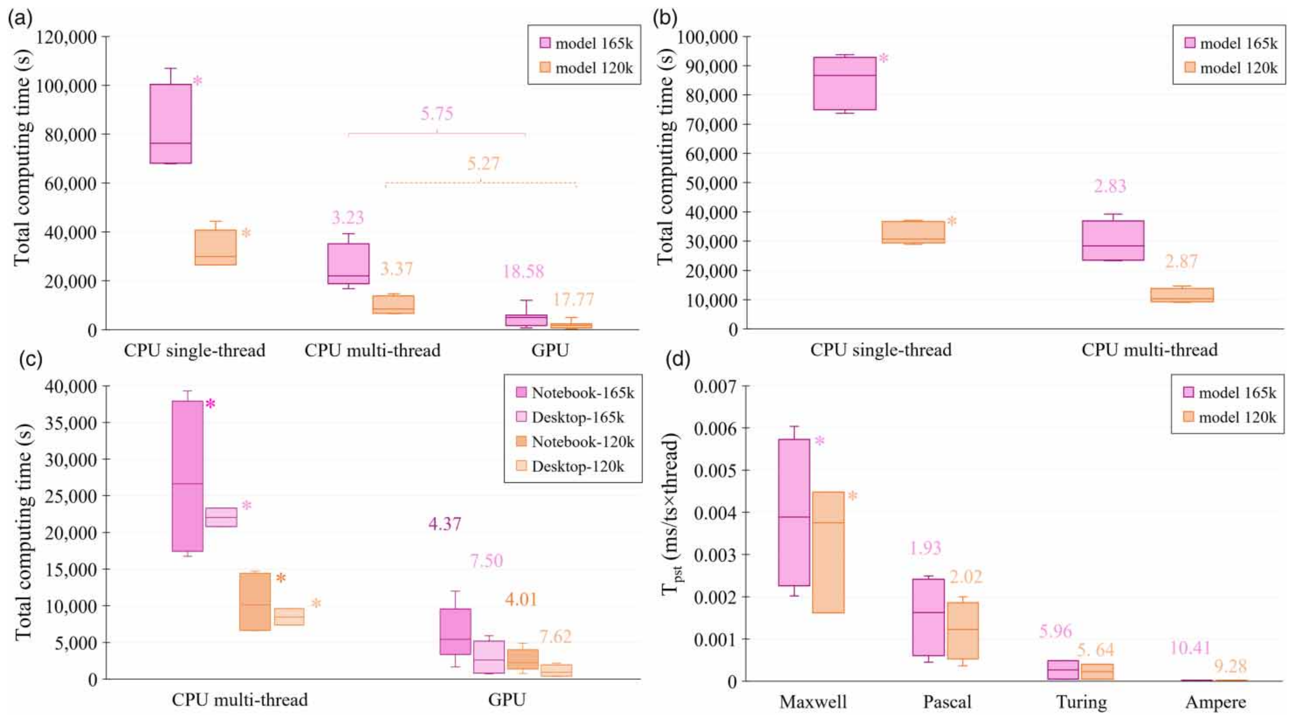


Figure 5 | For two domains (Figure 2), (a–c) boxplots representing total computing time (s): (a) on all the PUs tested ($N = 62$), (b) on different units of the same CPU ($N = 12$), (c) for parallelized computing on the tested device types ($N = 42$); (d) average time per step and thread (T_{pst}, in MIST) for the different GPU microarchitectures (Table 1, $N = 22$). In boxplots, the inner line and whiskers represent median, min, and max, respectively; numerical values on top of boxplots indicate the S_{PU} (–) of means, where * marks the reference for those S_{PU} (–) of means.

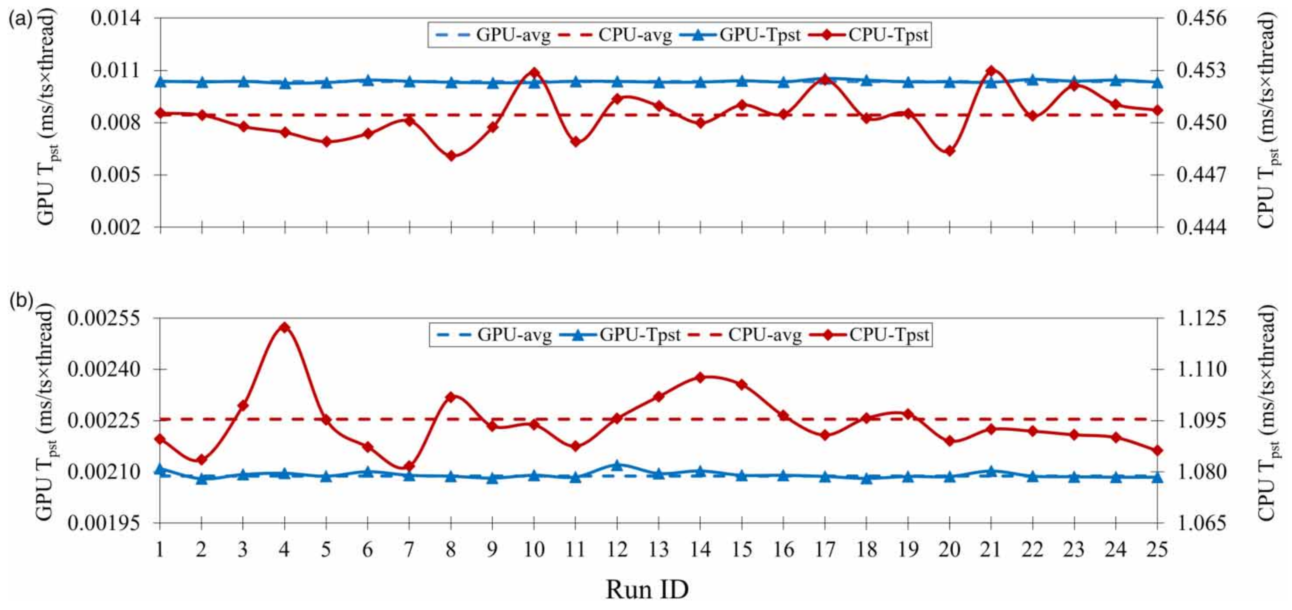


Figure 6 | Distribution of run-average time per step and thread (T_{pst}, Equation (3)) in relation to the population average (avg) estimated for replicate runs on CPU (i.e. i5-9600K, multi-thread) and GPU (i.e. NVIDIA Quadro P620; after Liu *et al.* 2018) for (a) a model of 165k elements, (b) a model of 120k elements.

Computing on the GPU was on average 17.5–20× and 5.5–6.5× faster than average CPU single- and multi-thread computing for the tested population, respectively (Figure 5(a), values above boxplots). As for CPU-computing, the computing time was shorter and had lower variability for the smaller model (Table S4). It is noteworthy that the variation within the tested GPUs was 91–96% of the observed T_{pst}, depending on the characteristics of the graphic card at hand (Table 1), which contrasts with the 18–41% variation observed for the CPU population. For instance, the fastest GPU tested (i.e. NVIDIA GeForce RTX3080) was 228–282× faster, i.e. in terms of T_{pst}, than the slowest GPU tested (i.e. NVIDIA GeForce GTX940M). When looking at the performance by the type of device, desktop GPUs were 7.5× faster than parallelized desktop CPUs (Figure 5(c), lighter boxplots). Notebook GPUs were 4× faster than parallelized notebook CPUs (Figure 5(c), darker boxplots). Moreover, desktop GPUs are faster than their notebook version, as shown in the case of NVIDIA GeForce GTX1050 and its mobile version GTX1050 MaxQ (Figure 5(c)).

Regarding the GPU architecture, it is usually an indicator of the number of threads available, and newer architectures have more threads/processors. Figure 5(d) shows the performance of each thread for each microarchitecture with the normalized average T_{pst}. Newer architectures, i.e. Ampere, outperform older architectures, i.e. Maxwell, by up to one order of magnitude regardless of the number of threads available. This observation cannot be extrapolated to all GPUs without considering additional specifications, which are further addressed in section 5.

Similar to the CPU-computing time stability test, when comparing the replicate runtimes for both computational domains, the GPU runtime was more stable in the case of the larger model. This was indicated by a lower standard deviation and by the rather constant average T_{pst} for the multiple runs in the GPU-computing time stability test (Figure 6, blue lines; Table S4). Although both computing modalities show similar tendencies, the GPU runtimes were significantly more stable than the CPU runtimes.

4.3. Model performance comparison

This section addresses the use of different indicators that highlight the performance of the two parallelization modalities on Iber + (i.e. multi-thread CPU and GPU) as opposed to the standard Iber version and each other. The selected indicators for performance are PE, S_{PU} and TS (Equations (5)–(8)).

Most of the CPUs reach a PE_{cpu} (Equation (5)) between 60 and 80%, with slightly higher PE for the smaller model, and desktop CPUs generally reach a higher PE than their notebook counterparts (Table 3). Given that there are no observations for single-thread GPU, the PE_{gpu} could not be estimated (long dash in Table 3). However, the PE_{cpu} observed insinuated that PE_{gpu} cannot be assumed to be 100%.

Table 3 | Calculated parallelization efficiency (PE, Equation (5)) for CPU-computing, speed-up ratio (S_{PU}, Equations (6) and (7) for single- and multi-thread CPU-computing (reference CPU in bold; S-t: single-thread modality, M-t: multi-thread modality), as well as the time-saving ratio (TS, Equation (8)) for GPU- relative to CPU-computing for computational domains of 165k and 120k elements

		165k model					120k model				
Processing unit	**Modality**	**PE (%)**	S^a_{PU-obs}	S^a_{PU-obs}	S^b_{PU-exp}	**TS**[c] **(%)**	**PE (%)**	$S^a_{PU\text{-}obs}$	$S^a_{PU\text{-}obs}$	$S^b_{PU\text{-}exp}$	**TS**[c] **(%)**
i7-4710HQ	S-t	79.28	0.88	0.47	0.75	99.32	82.08	0.84	0.33	0.75	99.17
	M-t		2.78	1.17		97.86		2.75	1.09		97.28
i7-8650 U -a	S-t	67.55	0.69	0.37	1.0	99.47	–	–	–	1.0	–
	M-t		1.86	0.78		98.56		–	–		–
i7-8650U -b	S-t	66.48	1.02	0.55	1.0	99.21	69.38	1.02	0.41	1.0	98.99
	M-t		2.71	1.14		97.91		2.84	1.13		97.2
i7-8650U -c[a,b]	S-t	59.59	**1.0**	0.54	**1.0**	99.23	63.12	**1.0**	0.4	**1.0**	99.01
	M-t		2.38	**1.0**		98.16		2.52	**1.0**		97.51
i7-7700 -a	S-t	81.8	1.23	0.66	1.05	99.05	77.97	1.24	0.49	1.05	98.77
	M-t		4.02	1.69		96.9		3.87	1.53		96.18
i7-7700 -b	S-t	76.22	1.08	0.58	1.05	99.17	74.35	1.21	0.48	1.05	98.81
	M-t		3.3	1.38		97.46		3.6	1.43		96.45
i7-7700 -c	S-t	77.76	1.27	0.68	1.05	99.02	79.72	1.29	0.51	1.05	98.73
	M-t		3.95	1.66		96.95		4.11	1.63		95.95
i5-9600K	S-t	109.33	1.37	0.74	1.22	98.94	120.25	1.4	0.55	1.22	98.62
	M-t		4.5	1.89		96.53		5.04	2.0		95.02
i9-9880H	S-t	–	–	–	1.23	–	–	–	–	1.23	–
	M-t		5.6	2.35		95.68		5.64	2.23		94.44
i7-10750H	S-t	57.8	1.38	0.74	1.34	98.74	65.29	1.4	0.56	1.34	98.62
	M-t		4.79	2.01		96.31		5.49	2.17		94.58
NVIDIA GeForce GTX 860M	GPU	–	18.68	7.84	2.35	85.59	–	16.78	6.64	2.35	83.44
NVIDIA GeForce GTX 750	GPU	–	15.8	6.63	2.35	87.82	–	–	–	2.35	–
NVIDIA GeForce GTX 940M	GPU	–	7.81	3.28	0.42	93.97	–	7.56	2.99	0.42	92.54
NVIDIA GeForce MX 130	GPU	–	13.11	5.5	1.18	89.89	–	12.04	4.77	1.18	88.11
NVIDIA GeForce GTX 1060	GPU	–	46.61	19.55	5.64	64.05	–	40.95	16.22	5.64	59.58
NVIDIA GeForce GTX 1050	GPU	–	29.5	12.38	2.47	77.25	–	22.46	8.9	2.47	77.83
NVIDIA GeForce GTX 1050 MaxQ	GPU	–	17.24	7.23	3.29	86.7	–	18.84	7.46	3.29	81.4
NVIDIA Quadro P620	GPU	–	17.87	7.49	2.35	86.22	–	16.3	6.46	2.35	83.91
NVIDIA GeForce RTX 2080Ti	GPU	–	110.37	46.3	18.06	14.89	–	84.07	33.3	18.06	17.02
NVIDIA GeForce GTX 1650Ti	GPU	–	56.0	23.49	5.63	56.82	–	48.59	19.24	5.63	52.04
NVIDIA GeForce RTX 3080[c]	GPU	–	129.67	54.4	22.3	**0.00%**	–	101.31	40.13	22.3	**0.00%**

Long dash: not calculated.

The reference processing unit for the estimation of the: [a]observed speed-up ratio, [b]expected speed-up ratio, [c]time-saving ratio.

When it comes to observed speed-ups, the notebook CPU Intel Core i7-8650U resulted in the lowest value of S_{PU} for the 165k model, whereas the notebook CPU Intel Core i7-4710HQ yielded the lowest S_{PU} for the 120k model. For comparison purposes, however, the observed S_{PU} was estimated for both models based on the i7-8650U single- and multi-thread, respectively. The observed S_{PU} for multi-thread CPU was between 1.86–2.52× and 5.60–5.64× when using the standard version as a benchmark for both models. Considering that many numerical modeling methods nowadays allow CPU-based parallelized computing, the model run in Iber + on the same CPU multi-thread is also used as a benchmark. In such a case, the maximum CPU S_{PU} goes down to less than half, i.e. 2.23–2.35×. Further statistical tests revealed that GPU-parallelization was always faster, with GPU S_{PU} of ca. 7.5× to ca. 100–130× when using the standard version as reference and GPU S_{PU} between ca. 3× and 40–55× when using the parallelized CPU-computing as reference (Table 3). Surprisingly, although the PE was higher for the 120k model, the observed S_{PU} was always higher for the 165k model. The expected S_{PU} (S_{PUexp} in Table 3) is independent of the number of threads of the PU or the model size, which makes it constant for both tested models. The S_{PUexp} values ranged between 0.75× and 22.30× , which was lower than the observed S_{PU} for both models when using the standard version as a reference. However, when the reference was the parallelized version, the S_{PUexp} was higher than the S_{PUobs} on the CPU single-thread modality and significantly lower than the S_{PUobs} on both parallelization modalities.

High-performance GPU-computing provided TS of minimum ca. 99 and 95.7%, for single- and multi-thread CPU-computing, respectively (Table 3). Moreover, using a high-end GPU seemed to save ca. 15–94% of the total computing time compared to using an old, low-range GPU. Simply said, using high-performance GPU-computing could save stakeholders up to 37.5 and 13.8 h as compared to using the non-parallelized version and the parallelized CPU version, respectively, on a high-range notebook CPU. If the user opted for a high-range desktop GPU instead of a low-range notebook GPU, it would save over 3 h of computing time for a 2D hydrodynamic model of the characteristics herewith described.

A model of similar size and characteristics run on the same GPU model (i.e. RTX 2080Ti), using also Iber + , took Fernández-Nóvoa *et al.* (2020) 3–5 min, whereas it took 7–14 min in this study. Also, running a hydrodynamic model twice as large and for twice as long on the same CPU model (i.e. i7-7700 @ 3.60 GHz, multi-thread) for the same case study, took Iber + ca. 8 h in this study and 1.25 h in Telemac-Mascaret using FEM (Pavlíček & Bruland 2019; M. Pavlíček, personal communication, 8 February 2021; Table S1). Seemingly, the computing time might not be determined only by the PU used but also by user-defined variables in the model.

4.4. Computing time: sensitivity analysis to model set-up and accuracy-speed trade-off

The effect of user-defined variables in the hydraulic model set-up on the computing time was tested on both parallelized CPU- and GPU-computing (see section 3.3; Figure 7). In summary, the effect of selecting the numerical approximation (i.e. S_{PU}

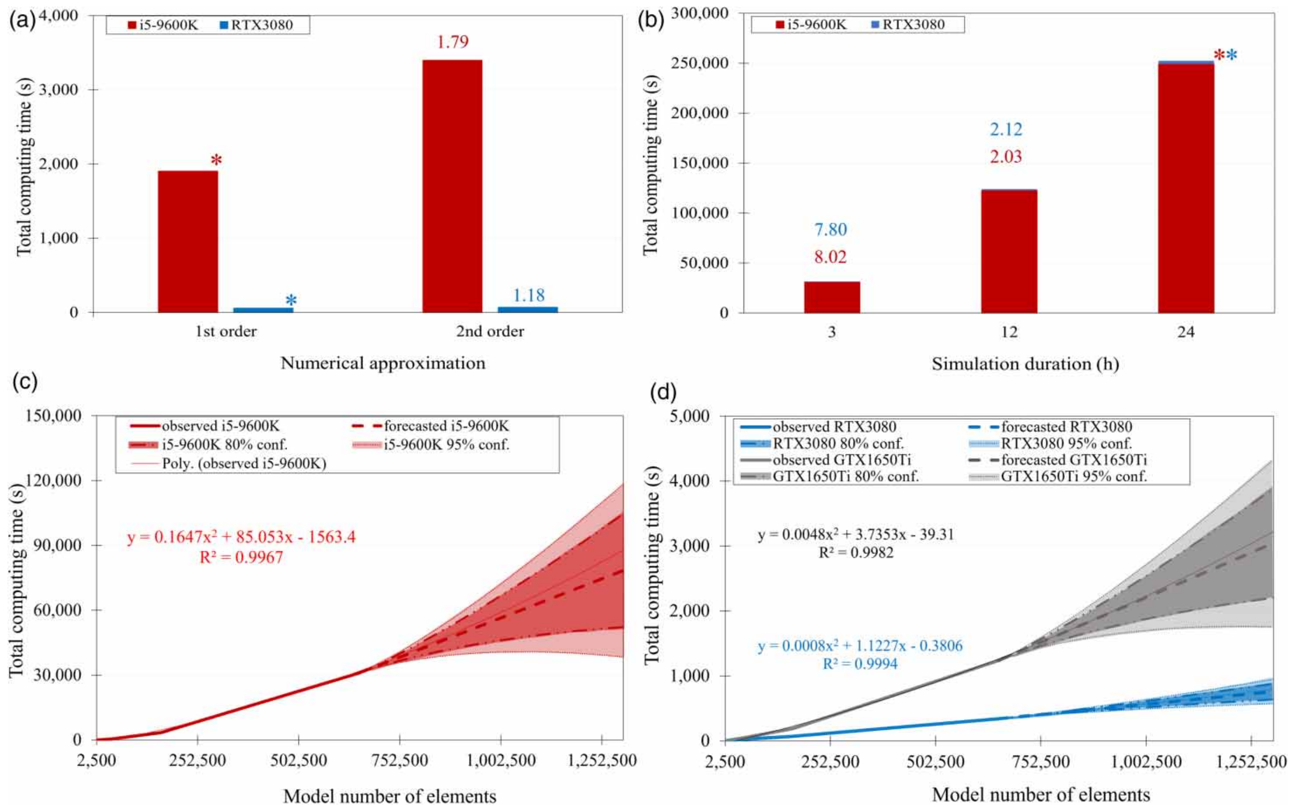


Figure 7 | Sensitivity analysis of the effect of user-defined variables in computing time (a–d) in selected PUs, i.e. desktop CPU Intel Core i59600K (red), desktop GPU NVIDIA GeForce RTX3080 (blue) and notebook GPU NVIDIA GeForce GTX1650Ti (gray; Table 1); *reference for S_{PU} (–); dashed lines in c–d) are forecasted, including the 80 and 95% confidence intervals. GPU desktop shows the best fit, followed by GPU notebook and CPU desktop, respectively. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.012.

<2×; Figure 7(a)) and varying the exporting interval (i.e. 1.5×) had a substantially lower impact on the PC-computing time and achieved S_{PU} than varying the duration of the simulation time, i.e. between 2× and 8× (Figure 7(b)), or the effect that selecting an adequate model number of elements had, i.e. between 5× and 3,000× (Figure 7(c) and 7(d)). The sensitivity analysis showed that GPU-computing was always faster. Expanding the analysis from this study's dataset to other modeling studies provided a context where high-end PC-GPUs performed similarly to server-CPUs, and PC-GPUs could compute up to 5× longer simulations than any CPU in <1 h (Figure 8(b)). Moreover, the relative difference between PC-CPU and PC-GPU performance is >25× larger than that observed when comparing PC- and server-GPUs. The difference between PC-CPU and PC-GPU performance is, on the other hand, 2.5× larger than that observed when comparing server-CPU and server-GPU (Figure 8(a)). This trend is also reported when comparing PC-CPUs and their server counterparts. Models with variable mesh sizes were more optimal in the accuracy–speed trade-off than the uniform mesh size models (Table 4). It was also reported that computing gets relatively faster (up to 36%) the larger the model is.

The total computing time was plotted for easier visualization of the results, and the more intuitive S_{PU} (Equation (6); values above bars in Figure 7(a) and 7(b)) was used to describe the main findings. The S_{PU}s achieved in the sensitivity analyses range from 1× –the reference PU for each test– to more than 8,700×.

The lowest S_{PU}s observed in the analyses were due to a less precise numerical approximation and less frequent writing of results. Using the first-order instead of second-order solution, and GPU- instead of CPU-computing, saves time: computing the first-order model on GPU was ca. 57× faster than computing the second-order model on parallelized CPU (Figure 7(a)). The

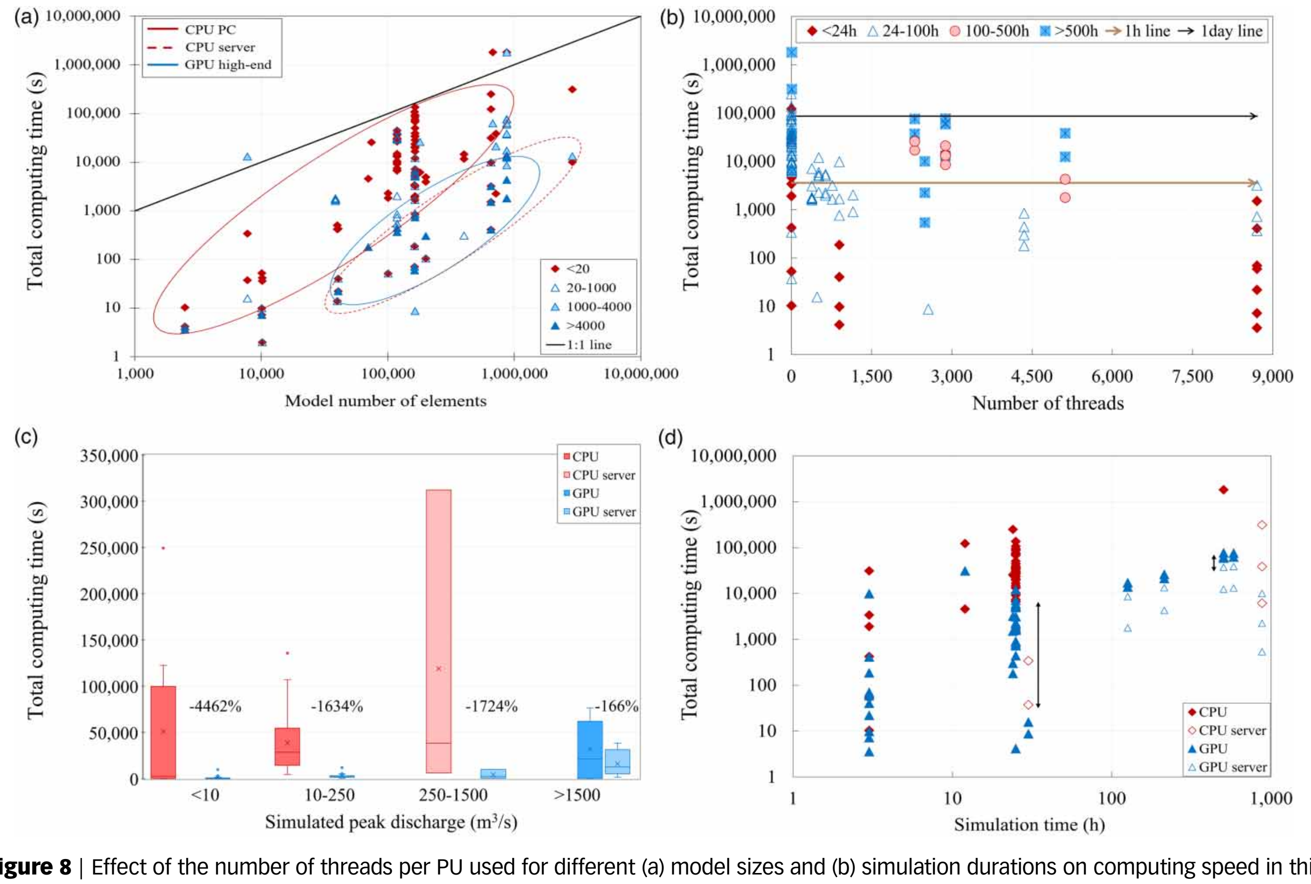


Figure 8 | Effect of the number of threads per PU used for different (a) model sizes and (b) simulation durations on computing speed in this study and literature (Table S1). Performance of PC- and data center PUs used for (c) different peak discharge and (d) model simulation duration classes in this study and literature. Ellipsoidal areas in (a) mark clusters for PC-CPUs (red line), server-CPUs (dashed red line), and high-end GPUs (blue line). Horizontal arrows in (b) mark the 1 h- (brown) and 1-day (black) reference lines; (c) relative performance difference (%) on top of boxplots; (d) performance gap between PC and data center PUs marked by arrows. In boxplots, 'x' is the mean marker, dots are outliers; inner line and whiskers represent the median, min, and max, respectively. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.012.

Table 4 | Computing time normalized per time step and cell (T_{psc}) and model accuracy (in terms of *F* and *C* indices) for different model sizes

Mesh size	Test ID	Number cells	T_{psc} (ms/ts $\times$ cell)	ΔT_{psc} (%)	*F* index (%)	ΔF index (%)	*C* index (%)	ΔC index (%)
uniform	sc01	44,679	4.34×10^{-6}	NA	60.22	NA	73.13	NA
variable	sc02	57,246	3.57×10^{-6}	−17.71	63.69	3.47	75.10	1.97
variable	sc03	71,856	2.99×10^{-6}	−16.38	64.77	1.08	74.37	−0.72
variable	sc04	132,088	1.90×10^{-6}	−36.37	63.45	−1.32	76.60	2.23
variable	sc05	163,398	1.67×10^{-6}	−11.96	65.03	1.58	75.40	−1.21
uniform	sc06	178,594	1.08×10^{-6}	−35.40	64.27	−0.75	77.06	1.66
variable	sc07	594,611	7.76×10^{-7}	−28.25	66.17	1.89	77.15	0.10
uniform	sc08	719,606	6.59×10^{-7}	−15.09	64.79	−1.38	76.97	−0.19
uniform	sc09	2,890,549	4.25×10^{-7}	−35.52	66.08	1.29	77.97	1.00
uniform	sc10	11,601,898	3.62×10^{-7}	−14.74	67.10	1.02	78.20	0.23

GPU-based model was ca. 50× faster for the 1 h-long exporting interval, whereas the S_{PU} reduced to 2.80× for the same device when the exporting interval was 3s long (not shown). Similar trends were observed for CPU-computing. The results exporting interval of 1 h was 78× faster on GPU than exporting results every 3 s on parallelized CPU.

The user-defined duration of the simulation and the model number of elements had the largest impact on the total computing time (Figure 7(b)–7(d)). Shortening the simulation time from 24 to 3 h was linearly faster regardless of the computing modality and running the model for 3 h on GPU was ca. 607× faster than running the model for 24 h on parallel CPU. For the same desktop GPU (i.e. RTX3080), the T_{pst} value, including the results exporting time per time step, was relatively the shortest for the 12 h-long simulation (i.e. 286 and 18% faster than the 24- and 3 h-long simulations, respectively). Such an effect of the results exporting time per time step on T_{pst} was not observed in CPU-computing.

The computing time needed for different simulation duration ranges in this study and literature was also analyzed (Figure 8(b)). For the analyzed dataset, it took <1 h to compute simulations of up to 100 h in GPU, and between <1 h and <1 day to compute >100 h simulations in GPU. As for the CPU, it took ca. 1 day to compute 24–100 h-long simulations, whereas high-end GPUs needed <1 h. Although most of the data available were for <100 h-long simulations, these observations suggest that there is no need to limit the duration of a simulation when using GPU-based computing.

The computing demand is quadratically increasing with the size of the model, regardless of the PU used to run the simulation (Figure 7(c) and 7(d)). Computing the finest model on a desktop GPU was 8,716× faster than running the coarsest model on a parallel desktop CPU. For instance, running a model of mesh size 0.25 m (i.e. 653k elements) was up to 3,007×, 298× and 96× slower, for parallelized CPU, notebook GPU and desktop GPU, respectively, than running a model of mesh size 4 m (i.e. 2.5k elements), yet the gain decreased steadily with the total number of elements. The normalized T_{psc} observed for models from 44k cells to 11.6 million cells indicated that models get relatively faster with an increasing number of elements, i.e. relative decrease in T_{psc} of 12–36% from model size *n* to model size *n-1* (Table 4). The model accuracy, although very slightly, generally improves with a more refined mesh, e.g. F index increases up to 7% and the C index increases up to 5%.

Figure 8(c) shows the relative performance difference (%, values above bars) between different PU types for Q_{peak} classes modeled in this study and literature (Table S1). The trend observed suggests that there is always a more pronounced difference between PC-CPU and PC-GPU than between server-CPU and server-GPU. The smallest difference is always between PC-GPU and server-GPU. The same analysis also indicates that larger models (generally those with higher Q_{peak}) are usually carried out using higher-end PUs (Figure 8(a), clusters indicated by dashed red and blue lines). When comparing the computing speed at different Q_{peak} for the same type of PU, however, the difference is largest for PC-GPUs, followed by PC-CPUs and server-GPU (not shown). Similarly, the performance of different PU types for different simulation durations was analyzed, highlighting a larger difference between PC- and server-performance for CPUs than GPUs, as marked by the black arrows in Figure 8(d). It is noteworthy that GPUs are more versatile, as indicated by the fact that CPUs were used for shorter simulation times and GPUs were used for longer simulation times both in this and other modeling studies.

5. DISCUSSION

5.1. Main factors affecting computing time for the 2017 flood model

The hydrodynamics observed in the model herein illustrated are briefly discussed in section 1.2 of the Supplementary Material. CPUs and GPUs have previously been used in all model sizes and applications, but there is a clear performance difference of at least one to two orders of magnitude, as shown by the gap between each class of PU and the 1:1 line in Figure 8(a). The number of elements affects the computing time, yet there seems to be a model number of elements threshold after which the acceleration is negligible given a certain set of conditions (Lacasta *et al.* 2014). Contrary to what was indicated by large-scale studies (e.g. Liu *et al.* 2018; González-Cao *et al.* 2019; Xia *et al.* 2019), Figure 7(c) and 7(d) show that the model number of elements increases the computing time following very strongly a quadratic polynomial trend both on CPU and GPU (i.e. $R^2 > 0.996$ in all cases). This observation indicates that medium to small models (e.g. $<$ 200,000 elements) can also benefit from significant speed-ups and support the use of any model number of elements for future investigation on computing acceleration–; i.e. the conclusions drawn from it can be as valid as those observed in a larger (e.g. $>$ 500,000 elements) and slower computing model. Normalizing the computing speed by the model size indicated that, although larger models always take longer to compute in absolute terms, larger models are relatively faster, in terms of T_{psc}, for a given PU (Figure 9). GPU-based computing was always faster in the small dataset used for the sensitivity analyses in this study (Figures 5 and 7).

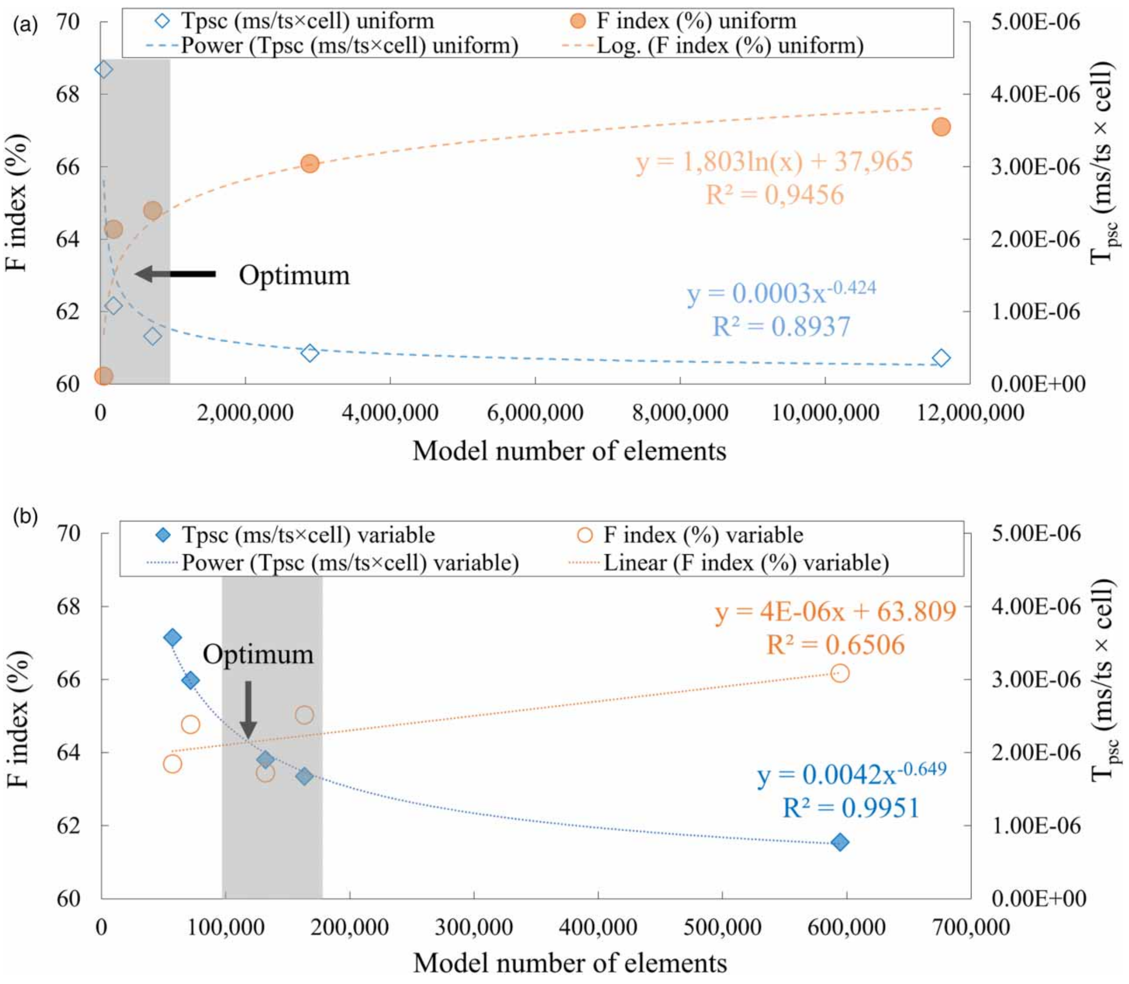


Figure 9 | Optimal curves of GPU-computing speed (in terms of T_{psc}, ms/ts $\times$ cell; blue diamonds) and model accuracy (in terms of *F* index, %; orange circles) evolution with increasing model number of elements for (a) uniform and (b) variable mesh size. Expected optimum between model accuracy and computational cost is marked in gray. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.012.

These observations indicate that running fewer and larger models on a GPU is most cost-efficient once the model configuration is optimized.

Now, although GPU-based computing should always be the preferred user choice when computing speed is key, it is important not to disregard the GPU specifications. Some older low-range GPUs might perform very similarly to newer high-range CPUs (when parallelized), and some newer mid-range GPUs can outperform some older high-end GPUs due to continuous technological advances (Figures 8(a) and 5(d), Table 3). Moreover, older yet superior versions of a PU, oftentimes designed for HPC, can outperform lower-end versions of newer PUs and even more modern architectures. An example of this is the notebook NVIDIA GeForce GTX 860M, released in 2014 with Maxwell microarchitecture, which outperformed newer generations of notebook graphic cards, such as GeForce GTX 940M, MX 130 (S. Son, personal communication, 11 June 2020), and GTX 1050 MaxQ, as well as the newer desktop NVIDIA GeForce GTX 750 and Quadro P620, all released between 2015 and 2018 with Maxwell or Pascal microarchitectures. This is probably because the GTX 860M belongs to a superior GPU version, i.e. the 'x60' version outperforms the 'x30', 'x40', and 'x50' versions. In this study, a high-end desktop GPU was found to fit the quadratic trend slightly better ($R^2 = 0.9994$, Figure 7(d)), and even be more stable for replicate runtimes (Figure 6(b)), than a notebook GPU or a desktop CPU of the same category. One reason could be that background processes occurring during the GPU-based simulation do not require the GPU, which ensures that most of the GPU processors available are used in the computing task. However, most of the background processes are CPU-demanding and exert a relatively larger impact on the CPU performance, especially when CPU-based computing is accelerated. Lastly, high-end GPUs (clustered by a blue line) can compute as fast as server-CPUs (clustered by a red dashed line) but are significantly more affordable (Figure 8(a)).

Reducing the simulation time from 1 day to 3 h yielded an 8× faster model for a given device, regardless of whether it is CPU or GPU. This observation is linear with the reduction of the simulation duration, i.e. an 8× shorter simulation computing 8× faster regardless of the type of PU (Figure 7(b)). The same reduction in simulation time resulted in a $>600\times$ faster computation on GPU than on parallelized CPU. The computation was up to two orders of magnitude faster for a server-CPU than a desktop CPU, whereas this difference was not observed between desktop GPUs and their server counterparts. This resulted in studies with simulation times of >48 h often preferring GPU-computing (Figure 8(d), Table S1). For a given simulation time, the GPU *versus* CPU speed-up was slightly higher when running a longer simulation than a shorter one, suggesting that short simulations are fast regardless of the PU used and that it is relatively more challenging to overcome the short computing times they take. This could be because background processes that require the use of CPU might have a more evident effect on shorter computing times. This hypothesis was supported by the CPU-computing time stability test (Figure 6). Considering interactive simulations that adjust the mesh size, simulation duration or results exporting frequency at critical locations as needed could increase the computing speed and provide the desired accuracy. Other modeling studies preferred using server-CPUs or GPUs when the peak discharge was >250 m^3/s (Figure 8(c)), which indicates that high peak discharges tend to lead to larger models that have a high computing demand only feasible in HPC PUs such as GPUs or server PUs. It is a common practice by flood modelers to subjectively adjust the simulation duration to the purpose of the study and the timeframe available. Choosing to model a too-short simulation time of a flash flood could be critical, nevertheless, especially if that means skipping the hours before the peak discharge is reached, as the soil saturation at Q_{peak} is highly dependent on previous hydrologic conditions. Consequently, it would not make sense to model only the 3 h near the peak precipitation, as it would not illustrate the same flood event. One way to save computing time when the aim is to model the floodplain water volume is to model the entire hydrograph and use a coarser mesh. If the aim is to detect erosion processes, the modeling focus could be on the timeframes in the hydrograph when a certain velocity threshold is exceeded, i.e. peak discharge, and use a smaller cell size. Depending on the shape of the hydrograph, however, limiting the simulation time to the 12 h around the flood peak, e.g. from 2 am to 4 pm for the 2017 flood, might be enough to capture the hydraulics of the flood and would also save significant time (i.e. 165× faster on GPU than parallelized CPU, and twice as fast when using the same PU).

The frequency at which the results are written determines flood data availability and could affect the computing time. However, there was very little difference in GPU-computing time when writing results every 1 h or every 30 s (not shown), with almost negligible absolute GPU writing times, i.e. ca. 0.01–0.06 s. The effect of decreasing the exporting interval seemed very steady unless the results were printed very frequently (i.e. 3 s). Looking at the same desktop GPU (i.e. RTX 3080), using the T_{pst} –including the results exporting time per time step–, instead of the S_{PU}, it was noticed that the computation was relatively the shortest for the 12 h-long simulation (i.e. 286 and 18% faster than the 24- and 3 h-long simulations, respectively). For the low T_{pst} values observed in GPU-computing, it can be more cost-efficient to run one longer 12 h simulation than splitting the model into several, shorter simulations that would add up to relatively large result exporting times. Such effect of the results exporting time per time step on T_{pst} has not been observed in CPU-computing. CPUs are, in fact, in charge of the writing of simulation results, yet the additional time this task takes is negligible when added to the already large computing times. The most obvious concern to emerge in a hydraulic study is that the most relevant results might not be captured by a too-long exporting interval. The results exporting interval should be defined primarily by the dynamics of the phenomenon studied. For instance, the peak discharge here modeled was 1 h long, hence, it would have not made sense to export results every 1 h, as the Q_{peak} might have been missed out. On the other hand, the peak of the hydrograph was practically flat and there were no substantial changes (Figure 4), thus, no need to monitor every second or even every 10 s. Finding the key combination between the level of detail and time invested is desired yet challenging without prior experience with the problem at hand. From the exporting intervals here analyzed, 30 s would have been as good as 1 h (in terms of computing speed), but the data processing and storage capacity requirements would have been very extensive. Tentatively, in this study, it was preferred to export every 1–5 min due to the steepness of the hydrograph, although there were no significant changes in the latest hours of the simulation and exporting then every 15 min would have sufficed.

Choosing the appropriate numerical approximation also has an impact on the speed of the model. More complex mathematics can overload the model and slow it down, as they require data transfer between the parallel threads (Morales-Hernández *et al.* 2020). When it is not possible to use a too-fine mesh, yet the accuracy required is high, it is recommended to use a second-order Roe's upwind approximation, where the accuracy can be increased faster when the exact solution is close. However, explicit upwind approximations compute ca. 20× slower than the implicit formulation when using an unstructured mesh (Bermúdez *et al.* 1998). Consistent with Bermúdez *et al.* (1998), modeling the 2017 flash flood using the first-order approximation was 20–80% quicker for a given device and ca. 60× faster on GPU than using the parallelized CPU version (Figure 7(a)), compared to the second-order approximation. Considering the adequate accuracy of the flood model using the first-order approximation (Figures 3 and 9), the compromise in accuracy when stepping down in order of accuracy is compensated by the gained speed-up, which could be very valuable in flood management. Although the impact of selecting the PU and user-defined variables on the Iber models herein presented has been addressed, the effect of choosing a different numerical model than Iber could be important. For example, running the 2D hydrodynamic model with twice as many elements and for twice as long on the same CPU model (i.e. i7-7700@3.60 GHz, parallelized on its 4 cores) for the same case study, took Iber + FVM first-order ca. 8 h (this study) and significantly shorter (i.e. 1.25 h, 6× faster) in Telemac 2D using FEM (Pavlíček & Bruland 2019; M. Pavlíček, personal communication, 8 February 2021, Table S1). Based on these data, it can be inferred that FEM solutions are generally faster than first-order FVM solutions, which was also hinted at by Pavlíček & Bruland (2019) when comparing FEM and FVM for the same Telemac 2D model. GPU-computing provides computing power that can overcome the additional cost of using FVM or other more complex numerical approaches.

Additionally, the type of grid used also affects the total computing time of a model; structured-mesh models solve faster than unstructured-mesh models (Lacasta *et al.* 2014; Vacondio *et al.* 2017), although they experience limitations when representing complex topographies accurately. The benchmarked case study presented herein was modeled using unstructured mesh due to the high complexity of the topography. It was attempted to counterbalance the lag expected due to a more computing-demanding type of mesh by a smaller model number of elements. The model with the variable mesh size optimally describes the most critical locations of the flash flood (i.e. F index 72%, Figure 3). The model accuracy will decline if a uniform mesh size is used (Table 4). If the aim of a modeling study is, for example, to obtain better information on the scour at a hydraulic structure, the mesh could be refined at a higher computational expense (Figure 9).

5.2. Potential acceleration by HPC in PCs and implications for flood studies

The results of this investigation show that GPU-computing of a 2D hydrodynamic model can be up to 130× faster than standard CPU-computing in a PC. Moreover, affordable consumer-grade GPUs can compute flash flood models up to 55× faster than the parallelized CPU version in a PC. CPU- and GPU-based simulations need different computational efforts based on the number of threads available in the PU, i.e. the CPUs tested in this study have from 1 to 8 threads, whereas the respective GPUs have from 384 to 8,704 threads (Tables 1 and 3, Figure 5(a)). As the simulation will experience a speed-up until all threads are fully used, the speed-up will be larger in GPU-based simulations than in CPU-based ones.

Another significant finding to emerge from this study is that using HPC in desktop GPUs could save up to 99.5 and 98.5% of the computing time needed for the standard- and parallelized CPU versions in a PC, respectively (Table 3). The time saved could be lower; however, if the modeling software used was originally designed to run on CPU and its code has not been rewritten to benefit as efficiently as possible from the GPU-parallelization, as has been the case for Iber (García-Feal *et al.* 2018). Overall, this study strengthens the idea of Tomczak *et al.* (2013) that single GPUs could outperform parallelized server-CPUs (Figure 8(a)). Single GPUs contain more threads than any HPC multi-CPU (depending on the microarchitecture; Figure 5(d)) and the communication between threads is simpler in GPUs than in CPUs. Interestingly, PE $>100\%$ was observed for the CPU i5-9600K (Table 3), which indicates that perhaps the parallelized CPU version of Iber + uses more than half of the logical threads available –note that i5-9600K has the same amount of physical and logical threads–.

Surrogate flood models (i.e. models based on Artificial Intelligence, AI) have addressed the issue of the high computational time often required by hydrodynamic models. Even in surrogate procedures, the current study could offer vital insights into the computational-time management in cases where a probabilistic flood mapping, which is nowadays the most modern approach in flood inundation modeling, is used. Most surrogate models are GPU-based. The speed-ups observed here hint at a great potential acceleration of surrogate flood modeling.

5.3. Prediction of computing time of hydraulic flood models and accuracy-speed trade-off

The ratio between the memory bandwidths for two given PUs (S_{PUexp} in this study, Equation (7)) was proposed as a proxy for the speed-up ratio in Tomczak *et al.* (2013), as the S_{PUobs} for a GPU *versus* a server-CPU was almost identical to the S_{PUexp}. The high similarity between server-CPU and GPU was observed in data from other modeling studies (Figure 8(a) and 8(d)), however, PCs' CPUs compute at least one order of magnitude slower. This results in more conservative S_{PUexp} than the S_{PUobs} for PCs, in some cases up to 8× lower (Table 3). Lastly, the S_{PUexp} does not account for differences between standard- and parallelized versions, nor differences in model size, yet it has been observed that larger models are faster than smaller models when normalized per time step and cell (Table 4, Figure 9). Considering the important differences in performance observed between computing versions (Figure 5(a)) and that the model number of elements was the most important user-defined variable controlling the computing time (Figure 7(c) and 7(d)), $\mathrm{S}_{\mathrm{PUexp}}$ is not a suitable proxy for the S_{PUobs} in flood studies carried out on a PC.

An alternative to S_{PUexp} is to use the computing time forecasted for different model sizes in the sensitivity analysis carried out in this study (Figure 7(c) and 7(d)). Additionally, the PE for CPUs and the TS for any PU could support selecting a suitable PU for other studies. Potentially, the optimal curve that shows the equilibrium between model size, accuracy, and computing time (Figure 9), can be used to confect a model of the most suitable characteristics. For the case study analyzed, it was easier to predict the model accuracy gain if the mesh was refined when using a uniform mesh size (i.e. $R^2 > 0.94$) than a variable mesh size (i.e. $R^2 = 0.65$). The optimal trade-off between accuracy gain and computing speed was reached at smaller models in the case of variable mesh configurations, i.e. optimum was at model size $<150{,}000$ cells for variable mesh *versus* $>700{,}000$ cells for the uniform mesh (Figure 9, gray area). Although the models get relatively faster the larger the model is, the gain in accuracy is limited to 5–7% at best when using a finer grid (Table 4).

A great part of the modeled discrepancies (i.e. $>43\%$ of the incorrectly simulated wet cells; Figure 3(a)) were due to the assumption that the owner of the land on the right floodplain did not block the water flow into the property, contrary to documented testimonies (Moraru *et al.* 2021; blue areas between bridges (ii) and (iii) on the right floodplain in Figure 3(a)), showing that real-time flood protections minimized the damage. The accuracy of the hydrodynamic model is limited by the lack of post-event validation data and the on-site flood protections implemented to protect private property during the flood event. The model validation was carried out against a fully water-covered area, where some local dry cells were considered wet (Moraru *et al.* 2021). This reduced the observed model accuracy regardless of the indicator used. Some adjustments to the model set-up might improve the model's accuracy without affecting the computing speed. The optimum accuracy-speed observed in this study is thus expected to be valid also for models with more extensive validation data. Simple measures based on early hydraulic modeling outputs in cases like Storelva in Utvik could prevent the river channel to change direction, which, in turn, could significantly reduce the consequences and total restoration and compensation costs associated with flooding. For instance, the 2017 flood damaged multiple properties and the reported repair costs exceeded 120 million NOK (Sunnmørsposten 2017). The economic loss of this single flood event is comparable to the average annual cost of flood repairs paid out by Norwegian insurance companies for 1,240 events per year for the period 1980–2023 (FinansNorge 2023). A conservative estimate of 10% of flood damage reduction by including hydraulic modeling information early in the river management process could have saved tens of millions of € in a single river alone.

6. CONCLUSIONS AND OUTLOOK

This study has shown that affordable consumer-grade GPU-modeling of flash floods in steep rivers can be up to 130× faster than the standard- and up to 55× faster than the parallelized CPU version in a PC. In other words, using consumer-grade high-performance GPU-computing could save stakeholders up to 1 day and a half (i.e. 99.5% of the computing time) for a small flash flood model, and even more for larger case studies. Moreover, a single PC-GPU can outperform parallelized data center CPUs, making it possible to model larger areas and even multiple case studies in a short time. The forecast computing times and optimal accuracy-speed trade-off curve provided could be used as a reference to predict the computing time needed for other studies. The sensitivity analyses carried out also support the selection of a suitable PU and model set-up.

The effect of user-defined variables such as the simulation time and model number of elements was most significant. The acceleration of the computing time strongly followed a quadratic correlation with the model number of elements ($R^2 > 0.996$), which suggests that future studies using a smaller model number of elements will experience very similar speed-ups. Normalized computing times by the number of time steps and model cells showed that larger models are relatively faster for a given PU. The optimal accuracy-speed trade-off for a given PU was observed in smaller models for variable grids (i.e. $<$ 150,000 cells) than for uniform grids (i.e. $>$ 700,000 cells). The research has also shown that first-order FVM approximations can provide a good compromise between accuracy and speed when run on a GPU. For instance, this study has satisfactorily modeled 65–80% of the observed flooded area in the most downstream 775 m of Storelva river during the 2017 flash flood at a very optimal computing speed (7–15 min). Describing optimally the critical locations in a river, and adjusting the mesh size in these locations, could ensure a model accuracy of >80% and a significant gain in computing speed. If a model of such characteristics is coupled with a monitoring system that gathers hydraulic data in real-time, flood data could be available on a minute- or a quarter of an hour-basis without additional computational cost.

In future work, it could be interesting to test what model set-up results in even better accuracy-speed trade-off both in the flash flood affecting Storelva river in Utvik and other test cases. Flood events with high peak discharges (e.g. 250–$>$ 1,500 m^3/s) are very computationally challenging to simulate, making these cases very suitable for HPC. More broadly, research is also needed to determine if the use of structured meshes would significantly affect the model's accuracy and suitability for flood management in steep rivers.

Furthermore, recent technological advances permit parallelization of GPUs – the so-called multi-GPUs, which would boost the speed-ups observed in the current study if the code in Iber + would be adapted to permit this computing modality. The present investigation lays the groundwork for future research into the use of GPU-based AI to predict the computing time of hydraulic models, as AI could be used complementarily to the performance indicators provided in this study to predict and shorten computing time. Further research could explore how to use AI models as an extension of a 2D hydraulic model, where hydraulic models could be used to train AI models for regions with little to no data, such as mountainous Norwegian rivers. Furthermore, the use of a PC equipped with a GPU could provide hydrodynamic information on flash flood events in nearly real-time, enabling improved and affordable flood risk preparedness and disaster management.

The findings of this study have several important implications for future flood management practices. For instance, it is possible to integrate GPU-based hydrodynamic models with early warning systems in small communities, which would spare valuable reaction time for the population even in case of extreme flash floods like the one herewith illustrated. The speed-ups achieved for small rivers are considered relevant for flood management, as the monetary cost of flood damage restorations is estimated to be of hundreds of millions, and the potential cost could be reduced by tens of millions of € if the knowledge from fast hydraulic simulations is implemented early on. Another important practical implication is that fast-computing flood models are compatible with modern visualization methods, such as AR/VR. This sustains the analysis and estimation of the consequences of flash floods with sufficient lead time by users with little numerical modeling background and facilitates their usability in communication and training applications due to their versatility and affordability.

ACKNOWLEDGEMENTS

The careful reading and constructive suggestions by Panayiotis Dimitriadis and an anonymous reviewer have strengthened the manuscript. The authors thank E. Bladé and M. Sanz-Ramos (Institut Flumen, UPC, Spain) for providing the macro for multiple results exportation and the insights on Iber. Many thanks to O. García-Feal (EPHYSLAB, UVigo, Spain) for clarification and guidance on GPU-based computing in Iber+. H. Pötting (RWTH Aachen, Germany) facilitated access to the wide range of consumer-grade processing units tested in this study. K. Alfredsen's (NTNU; Norway) formal advice made this manuscript clearer. This publication is part of the World of Wild Waters (WoWW) project number 949203100, which falls under the umbrella of the Norwegian University of Science and Technology's (NTNU) Digital Transformation initiative.

DATA AVAILABILITY STATEMENT

All relevant data are included in the paper or its Supplementary Information.

CONFLICT OF INTEREST

The authors declare there is no conflict.

REFERENCES

Ansell, L. & Dalla Valle, L. 2022 Social media integration of flood data: a vine copula-based approach. *Journal of Environmental Informatics* **39** (2), 97–110. https://doi.org/10.3808/jei.202200471.

Bellos, V. & Tsakiris, G. 2015 Comparing various methods of building representation for 2D flood modelling in built-up areas. *Water Resources Management* **29**, 379–397. https://doi.org/10.1007/s11269-014-0702-3.

Bermúdez, A., Dervieux, A., Desideri, J. A. & Vázquez, M. E. 1998 Upwind schemes for the two-dimensional shallow water equations with variable depth using unstructured meshes. *Computer Methods in Applied Mechanics and Engineering* **155** (1–2), 49–72. https://doi.org/10.1016/S0045-7825(97)85625-3.

Beven, K. J. 2012 *Rainfall-Runoff Modelling: The Primer*, 2nd edn. Wiley-Blackwell, John Wiley & Sons Ltd. https://doi.org/10.1002/9781119951001.

Beven, K., Lamb, R., Leedal, D. & Hunter, N. 2015 Communicating uncertainty in flood inundation mapping: a case study. *International Journal of River Basin Management* **13** (3), 285–295. https://doi.org/10.1080/15715124.2014.917318.

Bladé, E., Cea, L., Corestein, G., Escolano, E., Puertas, J., Vázquez-Cendón, E., Dolz, J. & Coll, A. 2014 Iber: herramienta de simulación numérica del flujo en ríos. *Revista Internacional de Métodos Numéricos Para Cálculo Y Diseño En Ingeniería* **30** (1), 1–10. https://doi.org/10.1016/j.rimni.2012.07.004.

Blöschl, G., Kiss, A., Viglione, A., Barriendos, M., Böhm, O., Brázdil, R., Coeur, D., Demarée, G., Llasat, M. C., Macdonald, N., Retsö, D., Roald, L., Schmocker-Fackel, P., Amorim, I., Bělínová, M., Benito, G., Bertolin, C., Camuffo, D., Cornel, D., Doktor, R., Elleder, L., Enzi, S., Garcia, J.C., Glaser, R., Hall, J., Haslinger, K., Hofstätter, M., Komma, J., Limanówka, D., Lun, D., Panin, A., Parajka, J., Petrić, H., Rodrigo, F.S., Rohr, C., Schönbein, J., Schulte, L., Silva, L.P., Toonen, W.H.J., Valent, P., Waser, J. & Wetter, O. 2020 Current European flood-rich period exceptional compared with past 500 years. *Nature* **583** (7817), 560–566. https://doi.org/10.1038/s41586-020-2478-3.

Bruland, O. 2020 How extreme can unit discharge become in steep Norwegian catchments? *Hydrology Research* **51** (2), 290–307. https://doi.org/10.2166/nh.2020.055.

Buttinger-Kreuzhuber, A., Konev, A., Horváth, Z., Cornel, D., Schwerdorf, I., Blöschl, G. & Waser, J. 2022 An integrated GPU-accelerated modeling framework for high-resolution simulations of rural and urban flash floods. *Environmental Modelling and Software* **156**, 105480. https://doi.org/10.1016/j.envsoft.2022.105480.

Casas-Mulet, R., Alfredsen, K., Boissy, T., Sundt, H. & Rüther, N. 2015 Performance of a one-dimensional hydraulic model for the calculation of stranding areas in hydropeaking rivers. *River Research and Applications* **31** (2), 143–155. https://doi.org/10.1002/rra.2734.

Caviedes-Voullième, D., Morales-Hernandez, M., Norman, M. R. & Ozgen-Xian, I. 2023 SERGHEI (SERGHEI-SWE) v1.0: a performance-portable high-performance parallel-computing shallow-water solver for hydrology and environmental hydraulics. *Geoscientific Model Development* **16** (3), 977–1008. https://doi.org/10.5194/gmd-16-977-2023.

Cea, L., Bladé, E., Sanz-Ramos, M., Fraga, I., Sañudo, E., García-Feal, O., Gómez-Gesteira, M. & González-Cao, J. 2020 *Benchmarking of the Iber Capabilities for 2D Free Surface Flow Modelling*. Servizo de Pulicacións Universidade Da Coruña. https://doi.org/10.17979/spudc.9788497497640.

Courant, R., Friedrichs, K. & Lewy, H. 1967 On the partial difference equations of mathematical physics. *IBM Journal of Research and Development (Republication From the Original Paper in Mathematische Annalen 100, 32–74, 1928)* **11** (2), 215–234. https://doi.org/10.1007/BF01342943.

Dam, G. 2018 Simulating the flooding in Utvik on 24 July 2017 using a high-resolution 2D hydro- and morphological model. *Draft report O17017/A*, Dam Engineering, 38pp. Available from: www.damengineering.no.

Dimitriadis, P., Tegos, A., Oikonomou, A., Pagana, V., Koukouvinos, A., Mamassis, N., Koutsoyiannis, D. & Efstratiadis, A. 2016 Comparative evaluation of 1D and quasi-2D hydraulic models based on benchmark and real-world applications for uncertainty assessment in flood mapping. *Journal of Hydrology* **534**, 478–492. https://doi.org/10.1016/j.jhydrol.2016.01.020.

Echeverribar, I., Morales-Hernández, M., Brufau, P., García-Navarro, P., 2018 Numerical simulation of 2D real large scale floods on GPU: The Ebro River. In: *River Flow 2018 – Ninth International Conference on Fluvial Hydraulics. E3S Web of Conferences* (Paquier, A. & Rivière, N., eds). (Vol. 40, p. 06007). https://doi.org/10.1051/e3sconf/20184006007.

Echeverribar, I., Morales-Hernández, M., Brufau, P. & García-Navarro, P. 2019 2D numerical simulation of unsteady flows for large scale floods prediction in real time. *Advances in Water Resources* **134**, 103444. https://doi.org/10.1016/j.advwatres.2019.103444.

Ejigu, D. K. 2020 Vannlinjeberegning for Storelva i Utvik i Stryn kommune (Waterline calculation for Storelva in Utvik in Stryn municipality). *Report no. 8/2020*, Norwegian Water and Energy Directorate (NVE), 36pp. (In Norwegian).

Fernández-Nóvoa, D., García-Feal, O., González-Cao, J., de Gonzalo, C., Rodríguez-Suárez, J. A., del Portal, C. R. & Gómez-Gesteira, M. 2020 MIDAS: a new integrated flood early warning system for the Miño River. *Water (Switzerland)* **12** (9), 2319. https://doi.org/10.3390/W12092319.

FinansNorge 2023 NASK - Naturskadestatistikk (Natural damage statistics report). Available from: https://nask.finansnorge.no/ (Last accessed: 2nd May 2023).

García-Feal, O., González-Cao, J., Gómez-Gesteira, M., Cea, L., Domínguez, J. M. & Formella, A. 2018 An accelerated tool for flood modelling based on Iber. *Water (Switzerland)* **10**, 1459. https://doi.org/10.3390/w10101459.

González-Cao, J., García-Feal, O., Fernández-Nóvoa, D., Domínguez-Alonso, J. M. & Gómez-Gesteira, M. 2019 Towards an automatic early warning system of flood hazards based on precipitation forecast: the case of the Miño River (NW Spain). In: *Natural Hazards and Earth System Sciences*. https://doi.org/10.5194/nhess-19-2583-2019.

Kalyanapu, A. J., Shankar, S., Pardyjak, E. R., Judi, D. R. & Burian, S. J. 2011 Assessment of GPU computational enhancement to a 2D flood model. *Environmental Modelling and Software* **26** (8), 1009–1016. https://doi.org/10.1016/j.envsoft.2011.02.014.

Lacasta, A., Morales-Hernández, M., Murillo, J. & García-Navarro, P. 2014 An optimized GPU implementation of a 2D free surface simulation model on unstructured meshes. *Advances in Engineering Software* **78**, 1–15. https://doi.org/10.1016/j.advengsoft.2014.08.007.

Liu, Q., Qin, Y. & Li, G. 2018 Fast simulation of large-scale floods based on GPU parallel computing. *Water (Switzerland)* **10**, 589. https://doi.org/10.3390/w10050589.

Liu, Z., Merwade, V. & Jafarzadegan, K. 2019 Investigating the role of model structure and surface roughness in generating flood inundation extents using one- and two-dimensional hydraulic models. *Journal of Flood Risk Management* **12**, e12347. https://doi.org/10.1111/jfr3.12347.

Morales-Hernández, M., Sharif, M. B., Gangrade, S., Dullo, T. T., Kao, S. C., Kalyanapu, A., Ghafoor, S. K., Evans, K. J., Madadi-Kandjani, E. & Hodges, B. R. 2020 High-performance computing in water resources hydrodynamics. *Journal of Hydroinformatics* **22** (5), 1217–1235. https://doi.org/10.2166/hydro.2020.163.

Moraru, A., Rüther, N., Bruland, O., 2020 Current trends in the optimization of hydraulic flood simulations in ungauged steep rivers. In: *10th International Conference on Fluvial Hydraulics (River Flow 2020)* (Uijttewaal, W., Franca, M., Valero, D., Chavarrias, V., Ylla Arbos, C., Schielen, R. & Crosato, A., eds). Taylor & Francis Group, pp. 1231–1238. https://doi.org/10.1201/b22619-170

Moraru, A., Pavlíček, M., Bruland, O. & Rüther, N. 2021 The story of a steep river: causes and effects of the flash flood on 24 July 2017 in western Norway. *Water (Switzerland)* **13** (12), 1688. https://doi.org/10.3390/w13121688.

Norwegian Mapping Authority 2013 Digital Elevation Model (DEM) Stryn, resolution 0.5m. Available from: https://hoydedata.no/LaserInnsyn/ (Last accessed: 18 November 2018).

Norwegian Mapping Authority 2015 Orthophotographs Sogn, pixel size 0.25m (online). Available from: https://norgeibilder.no/ (Last accessed: 8 February 2021).

Paprotny, D., Sebastian, A., Morales-Nápoles, O. & Jonkman, S. N. 2018 Trends in flood losses in Europe over the past 150 years. *Nature Communications* **9**, 1985. https://doi.org/10.1038/s41467-018-04253-1.

Pavlíček, M. & Bruland, O. 2019 Numerical modelling of flash flood event in steep river using Telemac 2D and Sisyphe. In: *XXVIth Telemac & Mascaret Users Conference (TUC2019)* (Ricci, S., ed.). CNRS/CERFACS, Toulouse, France, pp. 25–31. https://doi.org/10.5281/zenodo.3611581.

Roe, P. L. 1981 Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* **43**, 357–372. https://doi.org/10.1016/0021-9991(81)90128-5.

Roe, P. L. 1986 Discrete models for the numerical analysis of time-dependent multidimensional gas dynamics. *Journal of Computational Physics* **63** (2), 458–476. https://doi.org/10.1016/0021-9991(86)90204-4.

Rozos, E., Dimitriadis, P. & Bellos, V. 2022 Machine learning in assessing the performance of hydrological models. *Hydrology* **9** (1), 1–17. https://doi.org/10.3390/hydrology9010005.

Son, S. 2020 Optimization of hydraulic flood simulations in steep rivers using GPU and Machine Learning. MSc thesis, Universitat Politècnica de Catalunya - Norwegian University of Science and Technology, 78pp.

Sunnmørsposten. 2017 *Prislapp for flommen i Sogn og Fjordane: 120 millioner kroner (The Price tag for the Flood in Sogn og Fjordane: 120 Million NOK)*. Available from: https://www.smp.no/nyheter/i/QxwJa4/prislapp-for-flommen-i-sogn-og-fjordane-120-millioner-kroner

Tomczak, T., Zadarnowska, K., Koza, Z., Matyka, M. & Mirosław, Ł. 2013 Acceleration of iterative Navier-Stokes solvers on graphics processing units. *International Journal of Computational Fluid Dynamics* **27** (4–5), 201–209. https://doi.org/10.1080/10618562.2013.804178.

Vacondio, R., Dal Palù, A., Ferrari, A., Mignosa, P., Aureli, F. & Dazzi, S. 2017 A non-uniform efficient grid type for GPU-parallel Shallow Water Equations models. *Environmental Modelling and Software* **88**, 119–137. https://doi.org/10.1016/j.envsoft.2016.11.012.

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

First received 20 January 2023; accepted in revised form 18 August 2023. Available online 7 September 2023