Vienna Doctoral Programme on
Water Resource Systems
www.waterresources.at

Doctoral Thesis

Efficient Large-Scale Real-World Flood Simulations using the Shallow Water Equations on GPUs

A thesis submitted in fulfilment of the academic degree of

Doctor of Natural Sciences

under the supervision of

Prof. Dr. Günter Blöschl
Institute of Hydrology and Water Resource Management

Research conducted at TU Wien and VRVis Research Center

by

Ing. Zsolt Horváth

Student number 1327906
Mierová 1447/27
92401, Galanta
Slovakia

Vienna, June 1, 2017 .....

Examiner:	Prof. Dr. Günter Blöschl Inst. of Hydrology and Water Resource Management TU Wien Karlsplatz 13/222, A-1040 Vienna, Austria	.....
Examiner:	Prof. Dr. Christian Bucher Inst. of Building Construction and Technology TU Wien Karlsplatz 13/206, A-1040 Vienna, Austria	.....
Examiner:	Prof. Dr. Reinhard Hinkelmann Chair of Water Resources Management and Modeling of Hydrosystems Inst. of Civil Engineering TU Berlin Gustav-Meyer-Allee 25, 13355 Berlin, Germany	.....
Co-Supervisor:	Dr. Jürgen Waser Integrated Simulations Research Group VRVis Research Center Donau-City-Straße 11, A-1220, Vienna, Austria	.....

Parts of the present work have been accepted or published in peer-reviewed journals:

Zsolt Horváth, Jürgen Waser, Rui A.P. Perdigão, Artem Konev, Günter Blöschl. A Two-Dimensional Numerical Scheme of Dry/Wet Fronts for the Saint-Venant System of Shallow Water Equations. *International Journal for Numerical Methods in Fluids 77(3), pages 159-182, 2015.*

Zsolt Horváth, Rui A.P. Perdigão, Jürgen Waser, Daniel Cornel, Artem Konev, Günter Blöschl. Kepler Shuffle for Real-World Flood Simulations on GPUs. *International Journal of High Performance Computing Applications, 30(4), pages 379-395, 2016.*

Zsolt Horváth, Andreas Buttinger-Kreuzhuber, Daniel Cornel, Artem Konev, Jürgen Komma, Sebastian Noelle, Günter Blöschl, Jürgen Waser. Comparison and Validation of Three Shallow Water Schemes on Analytic, Laboratory and Real-World Cases. *International Journal for Numerical Methods in Fluids, Submitted.*

During the PhD I contributed to the following papers which were published in peer-reviewed journals and conference proceedings but are not included in this thesis:

Jürgen Waser, Artem Konev, Bernhard Sadransky, **Zsolt Horváth**, Hrvoje Ribičić, Robert Carnecky, Patrick Kluding, Benjamin Schindler. Many Plans: Multidimensional Ensembles for Visual Decision Support in Flood Management. *Computer Graphics Forum (Proceedings EuroVis 2014), 33(3), pages 281-290, 2014.*

Artem Konev, Jürgen Waser, Bernhard Sadransky, Daniel Cornel, Rui A.P. Perdigão, **Zsolt Horváth**, M. Eduard Gröller. Run Watchers: Automatic Simulation-Based Decision Support in Flood Management. *IEEE Transactions on Visualization and Computer Graphics, 20(12), pages 1873-1882 (Proceedings IEEE VAST), 2014.*

Günter Blöschl, **Zsolt Horváth**, Andrea Kiss, Jürgen Komma, Thomas Nester, Rui A.P. Perdigão, Alberto Viglione, Jürgen Waser. New Methods for Flood Risk Management. *Österreichische Ingenieur- und Architekten-Zeitschrift (ÖIAZ) Jg.160, pages 1-12, 2015.*

Daniel Cornel, Artem Konev, Bernhard Sadransky, **Zsolt Horváth**, Eduard Gröller, Jürgen Waser. Visualization of Object-Centered Vulnerability to Possible Flood Hazards. *Computer Graphics Forum (Proceedings EuroVis 2015), 34(3), pages 331-341, 2015.*

Daniel Cornel, Artem Konev, Bernhard Sadransky, **Zsolt Horváth**, Andrea Brambilla, Ivan Viola, Jürgen Waser. Composite Flow Maps. *Computer Graphics Forum (Proceedings EuroVis 2016), 35(3), pages 461-470, 2016.*

Daniel Cornel, Florian Schober, Artem Konev, **Zsolt Horváth**, Michael Wimmer, Jürgen Waser. Interactive Visualization of Adaptive Shallow Water Height Fields. *IEEE SciVis 2017, Submitted.*

This work has been conducted as part of the following research projects:

- Austrian Science Fund (FWF) project number W1219-N22 (Vienna Doctoral Programme on Water Resource Systems),
- European Research Council (ERC) Advanced Grant "FloodChange", project number 291152,
- Vienna Science and Technology Fund (WWTF) project number ICT12-009 (Scenario Pool),
- Austrian Funding Agency (FFG) within the scope of the COMET K1 program, project number 854174.

Author's Statement

I hereby declare that I independently drafted this manuscript, that all sources and references are correctly cited, and that the respective parts of this manuscript - including tables, maps, and figures - which were included from other manuscripts or the internet either semantically or syntactically are made clearly evident in the text and all respective sources are correctly cited.

Zsolt Horváth
Mierová 1447/27
92401, Galanta
Slovakia

.....

Acknowledgements

Many people deserve thanks for supporting me during my PhD and making it so enjoyable. First of all, my special gratitude goes to my supervisor Prof. Günter Blöschl, for his guidance, support and inspiration. I would also like to thank to all my colleagues from the Institute of Hydrologic Engineering at the TU Wien, especially Jürgen Komma and Rui Perdigaõ. I also want to thank all DK colleagues for the great time I had during my PhD. I enjoyed the symposia, courses and social activities.

Many thanks to all my colleagues at the VRVis Center, especially Andreas Buttinger-Kreuzhuber, Artem Konev, Daniel Cornel and of course my co-supervisor Dr. Jürgen Waser for his support and guidance.

I would like to thank to everyone who contributed to this work. I am grateful to Prof. Christian Bucher and Prof. Reinhard Hinkelmann for accepting the responsibility of evaluating my thesis.

I always had a lot of support from my friends who have persistently listened to all my problems I faced during this work. Special thanks go to Gabriel and Tunyó who where always there when I needed them.

Last but not least, I would like to thank to my parents Ildikó and József, my brother Balázs and of course to my wonderful and loving wife Zsuzsi for everything. I am very grateful to have such an amazing family and great friends.

Abstract

Climate change is one of the largest challenges humanity has to cope with today. Earth-orbiting satellites and other technological advances have enabled scientists to see the big picture, collecting many different types of information about our planet and its climate on a global scale. Data collected over many years reveals the signals of a changing climate. Europe and the northern hemisphere are warming at faster pace than the global average. Europe's Atlantic-facing countries are predicted to suffer heavier rainfalls, greater flood risk, more severe storm damage, according to the most comprehensive study of Europe's vulnerability to climate change. National Aeronautics and Space Administration (NASA) and National Oceanic and Atmospheric Administration (NOAA) confirmed that 2016 had broken the record for the hottest year ever previously held by 2015, which had itself broken the record that had been held by 2014. According to a climate change report from 2014, global sea level rose about 20 centimetres in the last century. The rate in the last two decades, however, is nearly double that of the last century. Moreover, in the last years a surprisingly large number of major floods happened around the world, which suggests that floods may have increased and will continue to increase in the near future.

Modern science in combination with the latest simulation technologies can help to understand the cause and the impact of the adverse phenomena related to climate change. Moreover, we can exploit our knowledge and simulation tools to prepare response measures which aim at reducing the risk associated with flood events. Today, a lot of effort is put into making flood simulations faster and more accurate to increase both computational efficiency and fidelity of the results. The aim of this thesis is to provide an efficient and robust simulation tool for large-scale flood simulations that can be used to support decision making. This goal is addressed by developing a new scheme for the shallow water equations (SWE), implementing it efficiently for graphics processing units (GPUs) and validating it on analytic, laboratory and real-world cases in comparison with other schemes.

Chapter I starts with a motivation for this thesis. This is followed by a general overview on fluid and flood simulations including the introduction of the SWE along with discretization methods for them. The next section gives a short insight into GPUs architectures and justifies the suitability of the SWE for parallel computations on these devices. In the last section of the chapter the main goals of this thesis are explained.

In Chapter II, we propose a new two-dimensional numerical scheme named HWP, to solve the SWE. The HWP scheme is an enhanced version of a scheme by Kurganov and Petrova (KP), which aims to improve the solution in the presence of partially flooded cells. The presented scheme is well-balanced, positivity preserving, and handles dry states. Mass conservation is ensured by using the draining time step (DTS) technique in the time integration process, which guarantees non-negative water depths. Unlike the KP scheme, our technique does not generate high velocities at the dry/wet boundaries, which are responsible for small time step sizes and slow simulation runs. We prove that the new scheme preserves "lake at rest" steady states and guarantees the positivity of the computed fluid depth in the partially flooded cells. We compare the new scheme, along with the KP scheme, against the analytical solution for a parabolic basin and show the improved simulation performance of the new scheme for two real-world scenarios.

Chapter III presents a new GPU implementation for the HWP and KP schemes on Cartesian grids. Previous implementations are not fast enough to evaluate multiple scenarios for a robust, uncertainty-aware decision support. To tackle this, we exploit the capabilities of the NVIDIA Kepler architecture and the new shuffle instructions. The KP scheme is simpler but suffers from incorrect high velocities along the wet/dry boundaries, resulting in small time steps and long simulation run-times. The HWP scheme resolves this problem but comprises a more complex algorithm, that represents an extra burden on the GPU. Here, an efficient and novel shuffle-based implementation is presented for both schemes. Moreover, a performance comparison is provided, in which we compare shuffle-based implementations with pure shared memory versions. The correctness and performance is validated on real-world scenarios.

In Chapter IV an exhaustive comparison and validation is performed and presented, which contains important use cases essential for developers and practitioners working with flood simulation tools. We discuss three state-of-the-art shallow water schemes, one by Kurganov and Petrova (KP), its successor by Horváth et al. (HWP), and our two-dimensional extension of the scheme by Chen and Noelle (CN). We analyse the advantages and disadvantages of each scheme on an extensive list of scenarios including several analytical and laboratory cases as well as a representative set of three historical floods. To enable the real-world studies, we address the implementation of the required boundary conditions (BCs), such as wall BCs, discharge BCs and water level BCs.

Chapter V contains a summary and the findings presented in this thesis, which advance the knowledge in simulating floods using the SWE on GPUs. The new HWP scheme tackles the non-physical velocities that appear along the dry/wet boundaries. This not only improves the numerical accuracy, but allows for faster simulation since there are no high velocity spots that act as a limiting factor on the time step sizes. Furthermore, an efficient GPU implementation is presented with focus on the reduction of the computational burden introduced by the HWP scheme. Finally, the validation cases give a comprehensive overview of three SWE schemes and reveal their strengths and weaknesses under various conditions. We observe that the KP and HWP schemes are more accurate than the CN scheme in some cases, however, in other cases they suffer from non-physical oscillations. Overall, good agreement is observed for all case studies rendering the presented shallow water schemes suitable for flood management applications.

Contents

List of Figures

List of Tables

List of Acronyms

AMR Adaptive mesh refinement

AoS Array of structures

API Application programming interface

BC Boundary condition

CFD Computational fluid dynamics

CFL Courant-Friedrichs-Lewy

CPU Central processing unit

DSP Digital signal processor

DTS Draining time step

FDM Finite difference method

FEM Finite element method

FPGA Field-programmable gate arrays

FV Finite volume

FVM Finite volume method

GPU Graphics processing unit

GTS Global time step

HLL Harten-Lax-van Leer

HPC High performance computing

HR Hydrostatic reconstruction

NSE Navier-Stokes equations

ODE Ordinary differential equation

PDE Partial differential equation

SB Shuffle–based

SM Streaming multiprocessor

SMO Shared–memory–only

SoA Structure of arrays

SPH Smoothed particle hydrodynamics

SV Saint–Venant

SWE Shallow water equations

Chapter I

Introduction

Floods are one of the main natural disasters responsible for human life losses and economic damages worldwide. There has been a surprisingly large number of major floods in the last years around the world [2], which suggests that floods may have increased and will continue to increase in the next decades [3]. The impacts of large-scale floods are rising mainly due to socio-economic development. Their frequency and intensity are showing signs of increase also due to climate change, as seen in the pan-European floods of 2002 and 2013 [4]. The European Community (EC) countries estimated that in the 1970-2006 period the average annual flood loss was about 4 billion dollars [5]. The ClimateCost project [6] estimates that, without improving the existing flood defense systems, the expected damages in 2050 will be worth 46 billion Euro/year (for the 27 countries of the European Union) with about 170,000 people being involved each year.

Flood risk assessment is of key importance in minimizing damages and economical losses caused by flood events. Studies on the impact of the climate change show that some regions in Europe are prone to rise in flood frequency. Northern and north-eastern parts of Europe [7] are highly affected areas and at a continental level 18.7 % of the territory is exposed to high flood hazard [8]. The first step in flood risk studies is the identification of flood prone areas [9]. This requires the implementation of hydrodynamic models that enable one to quantify the evolution of a flood and its hydraulic representative variables, e.g., water level and velocity [10]. Results of these models provide information to derive hazard maps which usually show potential flood extent, water height and sometimes velocity for predefined low, medium and high probability levels.

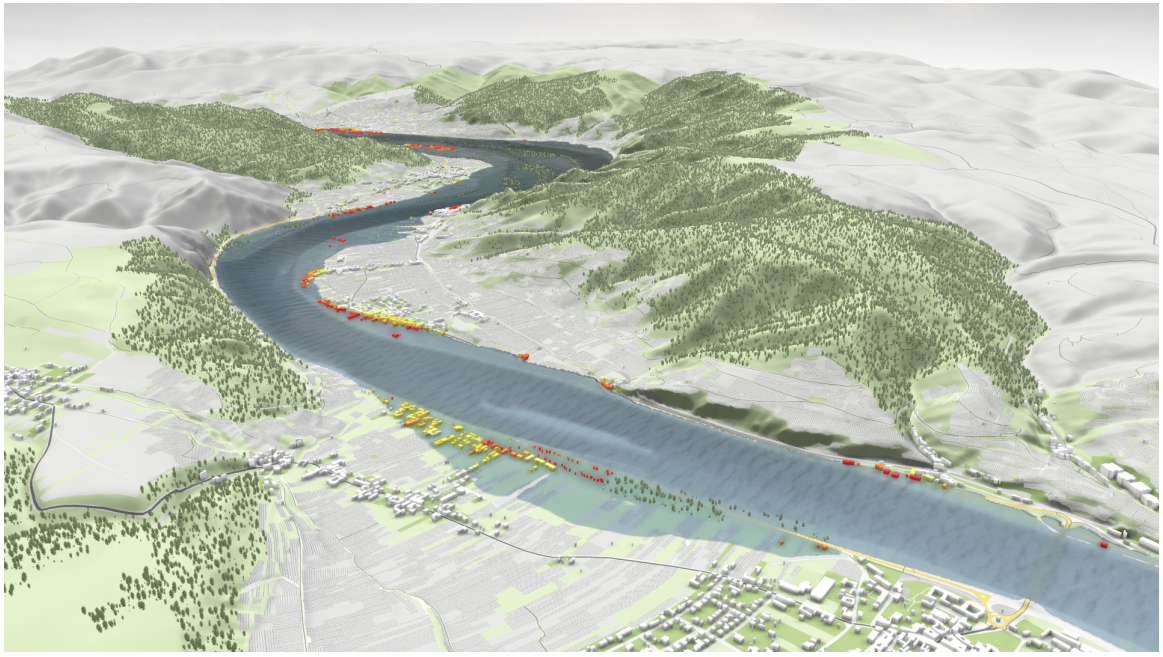


Figure I.1: Large-scale simulation of the 2013 June Danube flood in the Wachau valley in Austria, computed and visualized using the Visdom framework.

Floods differ by the processes that produce them [3]. One can distinguish between several types of floods, such as, coastal floods, river floods, dam breaks, flash floods, just to name a few. Here we focus mainly on river floods (see Figure I.1), which occur along small or big rivers and are usually triggered by rainfalls, sometimes in combination with snow-melt. Modern science in combination with the latest simulation technologies can help to understand the cause and the impact of these phenomena. Moreover, we can exploit our knowledge and simulation tools to prepare response measures which aim at reducing the risk associated with flood events. Today, a lot of effort is put into making flood simulations faster and more accurate to increase both computational efficiency and fidelity of the results.

1 Flood Simulations

We start with a brief overview of the techniques dealing with fluid simulations in general. In order to calculate the temporal evolution of fluids one has to solve the Navier-Stokes equations (NSE), which describe the motion of fluid substances. In flood simulations, where the water is in liquid form, NSE for incompressible flows are used [11]. Due to their complexity, the NSE are usually transformed into simpler forms by neglecting some properties, e.q., viscosity, vertical velocity, that are not important for the given problem. In order to solve the NSE, one has to use computational fluid dynamics (CFD). CFD provide a qualitative and quantitative prediction of fluid flows by means of mathematical modelling, numerical methods (discretization techniques), software tools (solvers, pre- and post-processing utilities). CFD enable scientists and engineers to perform numerical experiments by doing computer simulations. Traditionally, engineers use various techniques, like finite volume methods (FVMs), finite difference methods (FDMs), or finite element methods (FEMs) to solve the involved partial differential equations (PDEs).

Shallow Water Equations

In this work, we focus on the shallow water equations (SWE) and their numerical models, discretized on structured uniform grids, which serve as a basis for all simulations. The SWE, also called the Saint–Venant (SV) equations, are a set of hyperbolic PDEs if viscous and turbulent terms are neglected, which describe the flow of a fluid under certain assumptions. They were first introduced in 1871 by the French engineer Adhémar Jean Claude Barré de Saint Venant [12]. The underlying assumption is that the fluid flow is shallow, i. e., the horizontal length scale is greater than the vertical depth. The SWE can be derived by depth averaging the NSE and the continuity equation [11]. The flow is driven by a gravity-induced acceleration due to the slope of the free surface and the bathymetry. This criteria applies to many situations in hydrology and fluid dynamics, where the SWE are widely used. They can model various physical phenomena accurately, such as tsunamis, dam breaks, levee breaches or inundation, and provide also satisfactorily results for flood simulations regarding wave arrival times and flows velocities. For hyperbolic PDEs, the domain of dependence is always a bounded set, which means we can solve the system using an explicit scheme, since the waves travel at a finite speed. In addition, these properties make the explicit schemes particularly well suited for parallel architectures, such as modern graphics processing units (GPUs), which is also one of the key points of this work. For the sake of brevity, we refer the reader to Chapter II for a more comprehensive description of the SWE.

Lagrangian and Eulerian Discretization Methods

The most common techniques used to numerically approximate the solutions of the given equations are grid-based (Eulerian) simulations [13] and particle-based (Lagrangian) simulations.

First, we discuss the Eulerian or grid-based (mesh-based) description of the fluid flow for FVMs, which divides the domain into a number of control volumes. Each control volume is represented by a cell of the grid of the discretized domain. All flow properties are evaluated at specific points of the underlying simulation grid [14, 15, 16, 17, 18]. Cell-centered methods evaluate the physical quantities at the cell midpoints as cell averages, while in node-centered methods the physical quantities are evaluated as exact values at the grid point or cell vertices [19]. When discretizing the SWE other methods might be used as well, e. g.., FEM or FDM. A typical work out of the FEM involves dividing the domain of the problem into a collection of sub-domains, with each sub-domain represented by a set of element equations to the original problem, followed by systematically recombining all sets of element equations into a global system of equations for the final calculation [20]. The FDM works by approximating the differential equations with difference

equations, where the finite difference is like a differential quotient, except that it uses finite quantities instead of infinitesimal ones [21]. There are two main types of grids that are used in this representation, structured (regular) and unstructured (irregular). Both grid types have their benefits and drawbacks. Models defined on unstructured grid are more flexible when there is a need to follow a curve or structures, like buildings and walls [22]. Another important feature is that adaptive mesh refinement (AMR) comes implicitly for unstructured grids. Nevertheless, quad-tree based approaches exist also for the regular grids [23, 41], where each cell can be recursively subdivided or refined into four smaller ones - according to refinement criteria (velocity, bathymetry steepness variance, etc.) - until the desired resolution is reached. Even though irregular grids are very powerful because of the aforementioned features, they are not as suitable for parallel computations as grids with regular layout. The unstructured property of the simulation grid introduces some overhead that limits the computational performance due to multiple reasons, i.e., evaluation of trigonometric functions, higher branch divergency, uncoalesced memory reading and writing, and more burden on the memory bandwidth due to the structural information. When structured grids are used, parallel implementation becomes simpler and they have the advantage of lacking a time-consuming mesh generation. However, proper modelling of buildings and walls might need a special treatment, e. g., AMR [23, 24] or cut-cells methods [25, 26], if they are not aligned with with the main axes of the simulation grid.

Now, we shortly describe the particle based-simulations. A commonly used representative for the particle-based approach is the smoothed particle hydrodynamics (SPH) method. SPH is a Lagrangian mesh-free (mesh-less) method, where the coordinates move with the fluid. It was developed and introduced by Gingold and Monaghan [27] and Lucy [28] initially for astrophysical problems, but it has been heavily used in many fields of research, including fluid simulations. The SPH method works by dividing the fluid into a set of discrete elements, referred to as particles. The SPH method is extensively used in fluid simulations in general, and it is getting more attention lately also in flood simulations using the SWE [29, 30]. One major benefit of the SPH methods is that they provide more accurate solutions to moving and deforming boundaries compared to grid-based methods. However, there are multiple reasons that restrict their wider application. Most of the existing models have problems to fulfil the well-balanced property [31]. To overcome this problem, [32] recently reported a corrected SPH SWE model to provide well-balanced solutions for simulating complex shallow flows. Furthermore, they are too slow for time-critical applications as they require many-many particles, especially in urban environments with buildings, walls and other hydraulic structures that need high resolution. Computations of such simulations take between days and weeks, even more so if multiple scenarios are involved. Therefore, accurate prediction of urban inundations is still beyond the capability of most of the existing SPH models. For more information about SPH related methods and techniques we refer the reader to other works [33, 34, 35].

2 Parallel Computing on GPUs

GPUs were initially designed to accelerate computer graphics, namely updating the color of each pixel on the computer screen many times per second. This is an embarrassingly parallel task, which maps very well to stencil computations. GPUs went through an extensive evolution over the past decade. Nowadays, they are known as a very powerful tool which is used in the fastest supercomputers. These GPUs are widely used by the scientific computing community to solve various problem, e.g., to simulate blood flow, forming of the galaxies, climate models, or as in our case simulate floods. Today, GPUs are found in nearly every desktop and laptop computer, and they are used in supercomputers to accelerate computations.

One can pick from many frameworks, languages and standards when it comes to parallel programming on GPUs. The most common ones are CUDA, OpenCL, C++ AMP, OpenMP, and OpenACC. CUDA is a parallel computing platform and application programming interface (API)

model created by NVidia. It allows software developers and software engineers to use a CUDA-enabled GPUs for general purpose computing. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels. Open Computing Language (OpenCL) is a more general framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), GPUs, digital signal processors (DSPs), field-programmable gate arrayss (FPGAs) and other processors or hardware accelerators. C++ Accelerated Massive Parallelism (C++ AMP) is a native programming model that contains elements that span the C++ programming language and its runtime library. It provides an easy way to write programs that compile and execute on data-parallel hardware, such as GPUs. C++ AMP is a library implemented on DirectX 11 and an open specification from Microsoft for implementing data parallelism directly in C++. OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviour. OpenACC (for open accelerators) is a programming standard for parallel computing. The standard is designed to simplify parallel programming of heterogeneous CPU/GPU systems. Like in OpenMP, the programmer can annotate C, C++ and Fortran source code to identify the areas that should be accelerated using compiler directives and additional functions.

From this point on, we refer to NVidia GPUs and CUDA if not explicitly stated otherwise. The major difference between the CPUs and GPUs is that, GPUs have a parallel architecture that emphasizes executing many concurrent tasks slowly, rather than executing a single one very quickly. On a GPU, parallel tasks are called threads. These are scheduled and executed simultaneously in groups referred to as warps. One warp contains 32 threads, which is the number of threads effectively processed in parallel by one CUDA streaming multiprocessor (SM). GPUs have many SMs running in parallel to increase the effective parallelism. Each SM has its own resources, such as, cores, shared memory, registers, LD/ST units, warp schedulers, etc. SMs execute multiple groups of warps by switching between. Furthermore, threads are organized into larger structures called blocks, and blocks are organized into grids [36, 37]. Figure I.2 shows a schematics of the CUDA thread execution model of a conceptual 2D grid. In order to achieve peak performance on a GPU one should always strive maximizing the occupancy by optimizing the resource usage and hiding various sources of latencies by feeding the device with enough work.

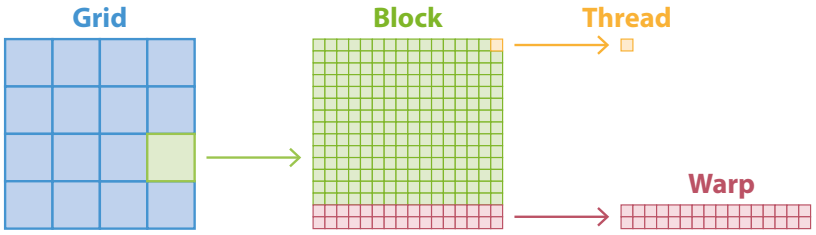


Figure I.2: Schematics of the CUDA thread execution model for a conceptual 2D problem.

As mentioned above, our primary interest lies in flood simulation using SWE, which are a set of hyperbolic PDEs, thus have a bounded set as a domain of dependence and they can be solved by explicit numerical schemes. Due to this property, explicit schemes typically rely on stencil computations, making them inherently parallel, and therefore a near perfect match for GPUs. Such schemes have an obvious and natural parallelism in the sense that each grid cell can be processed independently of its neighbours. In these schemes very little global communication is needed in the computational domain to advance the solution forward in time. Therefore, they are ideal to exploit the GPU's parallel nature and to be solved efficiently on it [38, 39, 40, 41, 42]. Unfortunately, a thorough introduction into high performance computing (HPC), GPU architectures and parallel computing on GPUs is beyond the scope of this work. Hence we refer the reader to the scientific literature [43, 36, 44, 45, 37]. More about the implementation of shallow water schemes on GPUs can be found in Chapter III.

3 Aim of the thesis

The aim of this thesis is to develop a robust and integrated flood simulation system capable of simulating real-world large-scale floods in real-time to support flood managers in decision making. An emphasis is put on the numerical stability of the delivered solutions, which is crucial for the users. Flood simulation systems often suffer from stability issues, that can significantly prolong the decision making process. Users have to adjust and tune the simulation parameters until a stable set-up is found, for which multiple simulation runs are needed. Another drawback of these systems is that they are not fast enough to deal with multi-dimensional (multi-parameter) ensembles of high resolution meshes where large-scale scenarios are simulated [46]. Ensemble simulations are of key importance in designing uncertainty-aware protection plans for various flood scenarios. In this work, our goal is to develop and implement a scheme for the SWE that is numerically stable, not only from the simulation point of view, but also from the physical point of view. This means, it should be possible to perform long run simulations (weeks, months) without a slow down due to numerical issues and the scheme should not produce non-physical states, like high velocities along the dry/wet boundaries. Moreover, the implemented solver has to be fast enough to deal with multi-dimensional ensembles for decision making in a reasonable time. All schemes in this work are implemented and optimized for parallel execution on NVidia GPUs using the CUDA toolkit and integrated into the simulation and visualization framework called Visdom [47].

The thesis consists of five chapters of which three chapters are based on manuscripts that have been published in or submitted to scientific peer-reviewed journals. Chapter I provides the motivation of this work along with a general introduction into flood simulations and parallel computations on GPUs. In Chapter II, a new two-dimensional numerical scheme, named HWP, is presented. The HWP scheme aims to solve the SWE in the presence of dry/wet boundaries. This is achieved, by applying a new reconstruction technique and using the draining time step technique in the time integration process, which guarantees non-negative water depths, and tackles the non-physical high velocities of its former scheme. In Chapter III, a new implementation of two second-order numerical schemes of the SWE for GPUs is discussed. We focus on the schemes of Kurganov and Petrova (KP) and Horváth et al. (HWP). This chapter provides detailed information on how to optimize the KP and HWP schemes for peak performance using the Kepler shuffle instructions. In addition, we discuss performance and memory considerations of the most time-consuming and resource-intensive computational steps in our algorithm. In Chapter IV, a comprehensive comparison and validation is presented for three state-of-the-art shallow water schemes, one by Kurganov and Petrova (KP), its successor by Horváth et al. (HWP), and our two-dimensional extension of the scheme by Chen and Noelle (CN). Various validation cases are investigated, such as analytic, laboratory and real-world, where we address the implementation of the required boundary conditions (BCs), such as wall BCs, discharge BCs and water level BCs. Finally, Chapter V presents the main conclusions of this thesis and possible future works.

Chapter II

A Two-Dimensional Numerical Scheme of Dry/Wet Fronts for the Saint-Venant System of Shallow Water Equations

Zsolt Horváth, Jürgen Waser, Rui A. P. Perdigão, Artem Konev, Günter Blöschl

This chapter is based on a paper which is published in
International Journal for Numerical Methods in Fluids 77(3), pages 159-182, 2015.

Abstract

We propose a new two-dimensional numerical scheme to solve the Saint-Venant system of shallow water equations in the presence of partially flooded cells. Our method is well-balanced, positivity preserving, and handles dry states. The latter is ensured by using the draining time step technique in the time integration process, which guarantees non-negative water depths. Unlike previous schemes, our technique does not generate high velocities at the dry/wet boundaries, which are responsible for small time step sizes and slow simulation runs. We prove that the new scheme preserves "lake at rest" steady states and guarantees the positivity of the computed fluid depth in the partially flooded cells. We test the new scheme, along with another recent scheme from the literature, against the analytical solution for a parabolic basin and show the improved simulation performance of the new scheme for two real-world scenarios.

1 Introduction

The shallow water equations are of great importance in many application areas, such as flood [38], or tsunami simulations [48] in urban and rural areas, in which waves propagate with a horizontal length scale much greater than the vertical length scale ("shallow waves"). Floods may produce enormous economic damage and human casualties, which have been recently reported to increase due to a number of reasons [49, 2]. In order to minimize the adverse effects of floods, flood mitigation measures are needed, such as adjusting regional planning, constructing levees and polders, establishing evacuation plans, and issuing timely flood warnings once the flood is imminent. All of these tasks rely on accurate and fast simulations of the flood wave propagation based on the shallow water equations. In this paper, we are mainly interested in large-scale flood simulations, for which the main challenge is the simulation time (see Figure. II.1).

In recent times, flood events have been reported to occur more frequently [2]. Moreover, their relation to climate change has been seen to embody emerging spatio-temporal features stemming from nonlinear landscape-climate dynamics [50]. These floods can contaminate the soil and our water sources, not to mention the human casualties and the financial costs of the caused damages. For this reason, we have to be well prepared and be able to act in time to minimize damages and losses. The shallow water equations serve as a fundamental and efficient tool for simulating floods and creating protection plans for such catastrophic events. Waser et al. [46] present an integrated solution based on the shallow water equations and on multidimensional, time-dependent ensemble simulations of incident scenarios and protective measures. They provide scalable interfaces which facilitate and accelerate setting up multiple time-varying parameters for generating a pool of pre-cooked scenarios.

Shallow water waves are described by the Saint–Venant (SV) system [51, 17], in which the motion of the fluid is introduced by the gravity. The equations are derived from depth-integrating the Navier-Stokes and the continuity equations [52, 11]. This leads to a vertically lumped description of the wave propagation, which assumes invariance in fluid properties with depth. If the fluid is stratified, i.e. features vertical layers with different properties (e.g. temperature, density), a system of multiple level shallow water equations (e.g. [53, 54]) can be used, with as many levels as there are the layers in the stratified fluid. In this paper, we focus on single-layer shallow waves. Note that the framework can be applied to multiple-level systems.

We are interested in a robust and fast numerical method for the SV system of the shallow water equations (SWE). We begin with a brief overview, and discuss the most important details, which are essential to completely understand the system and the proposed numerical scheme.

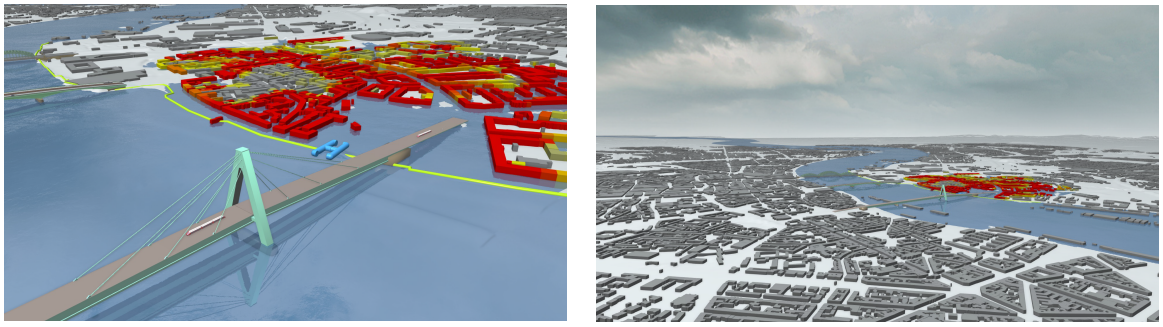
The two-dimensional shallow water waves can be described by the following SV system [55]:

$$\underbrace{\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t}_{\text{conserved variables}} + \underbrace{\begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y}_{\text{flux functions}} = \underbrace{\begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix}}_{\text{source terms}}, \tag{II.1}$$

where h represents the water height, hu is the discharge along the x-axis, hv is the discharge along the y-axis, u and v are the average flow velocities, g is the gravitational constant, and B is the bathymetry (see Figure II.4a). Subscripts represent partial derivatives, i.e., $\mathbf{U}_t$ stands for $\frac{\partial U}{\partial t}$. In vector form the system can be written down as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U}, B)_x + \mathbf{G}(\mathbf{U}, B)_y = \mathbf{S}(\mathbf{U}, B), \tag{II.2}$$

where $\mathbf{U} = [h, hu, hv]$ is the vector of conserved variables, $\mathbf{F}$ and $\mathbf{G}$ are flux functions, and $\mathbf{S}$ represents the source term function.



a b

Figure II.1: Real-world large-scale simulations of a breach in an urban area in Cologne, Germany. (a) Distant view of the city. (b) Closer view of the flooded area.

The method should be accurate on smooth parts of the solution and should not create spurious oscillations near discontinuities, i.e., at the dry/wet boundaries. These equations accurately capture both steady-states and quasi-steady flows [17, 56] in which the flux gradients are balanced by the source terms. Well-balanced numerical schemes have to be capable of exactly balancing the source terms and numerical fluxes, so that the "lake at rest" steady states are preserved:

$$u = 0, \quad v = 0, \quad w := h + B = \text{Const.}, \tag{II.3}$$

where w is the total elevation of the water surface. When $h = 0$, the previous state can be reduced to the "dry lake" steady state:

$$hu = 0, \quad hv = 0, \quad h = 0, \tag{II.4}$$

which means that no water is present, and the discharges are also zero. A good numerical scheme should be able to exactly preserve both "lake at rest" and "dry lake" steady states as well as their combinations. The methods that exactly preserve these solutions are termed "well-balanced" [16, 57, 58, 59, 60, 61, 62, 63]. Therefore, an ideal method should be well balanced in the sense that fluxes and source terms balance exactly and they result in zero velocities for "lake at rest" cases.

The simulation of water waves is particularly challenging near dry areas. Standard numerical methods may fail at the dry/wet fronts and produce negative water heights. If the water height becomes negative after the time integration, the whole computation breaks down. All computed water heights must be non-negative. To accomplish this, various positivity preserving methods are available [14, 16, 17, 18, 64]. The last major requirement is the stability of the scheme. In general, to fulfill this requirement, the Courant-Friedrichs-Lewy (CFL) condition [65, 66, 18] is applied. In case of a second order scheme the CFL condition allows for each wave to travel at

most one quarter of a grid cell per time step (CLF = 0.25), thus limiting the propagation of the information by limiting the time step.

In this paper, we present a new grid-based, central-upwind scheme that satisfies the criteria above. Following a new reconstruction of the water surface and the draining time step technique [67], we develop a well-balanced, positivity-preserving scheme for the dry/wet fronts. The new method is two-dimensional, which makes it suitable for real-world flood simulations by overcoming limitations of one-dimensional schemes. Furthermore, the scheme is well-balanced at the partially flooded cells. This allows for longer time steps which results in shorter simulation run time. We point out that our new two-dimensional scheme is not a direct dimension-by-dimension extension of the one-dimensional scheme presented by Bollerman et al. [16], since the latter does not contain some terms which appear only in the two-dimensional scheme. Our two-dimensional scheme is more than a juxtaposition of two one-dimensional schemes.

The paper is structured as follows. In Section 2, we discuss the existing solutions and their drawbacks. In Section 3, we show how to discretize the SWE on a regular grid. In Section 4, we describe the water surface reconstruction at partially flooded cells. In Section 5, we prove the positivity preserving property of the scheme with the new reconstruction and demonstrate the application of the draining time step technique. In Section 6, we present the evaluation of the proposed scheme. The scientific contributions of the proposed scheme span from fundamental numerical developments to an added practical value to engineering, environmental and hazard prevention applications. The paper thus contributes with the following key points:

- a physically consistent solution;
- no numerical artifacts at the boundaries;
- 2-10 times faster than previous schemes;
- evaluation on two large-scale, real-world scenarios relevant for society (urban and rural flooding);
- an improved numerical scheme for the SWE, based on the two-dimensional reconstruction at the dry/wet boundaries;
- well-balanced states at the dry/wet boundaries in partially flooded cells;
- application of the draining time step technique to preserve non-negative water heights while advancing the solution in time;
- avoiding spurious high velocities at the dry/wet boundaries;
- validation against an analytical solution.

2 Related Work

There are many schemes available in the literature that satisfy some of the criteria listed in the previous section. For other types of problems (including smooth phenomena, i.e., the formation of eddies [68, 69]), higher order schemes [61, 63] may be required. We are interested in urban and rural flood simulations, hence, we focus on the second-order schemes that produce sufficiently accurate results for these problems.

In the numerical treatment of the SWE, the spatial domain is discretized. For this purpose, one can use a structured or an unstructured mesh. Most of the numerical schemes developed for the SV system are based on the Eulerian approach [57]. This approach uses fixed points in space (grid points) where the fluid properties are evaluated. In general a uniform rectangular mesh or

a triangular mesh is used for this purpose [18]. The second approach is called Lagrangian, where the fluid is being tracked as it flows through space. This is a particle-based approximation of the fluid flow, where each fluid element or particle stores its own properties (e.g., mass, velocity, position).

We are interested in a two-dimensional, well-balanced, positivity preserving scheme discretized on a regular rectangular mesh, which is often referred to as regular grid. The scheme has to be able to handle dry and near dry states and solve accurately and efficiently problems characterized by strong discontinuities (e.g., dam breaks, flood breaches).

Kurganov and Levy [17] introduce a second-order scheme using discretization on a regular grid. They propose to use a different reconstruction of the water surface in near dry areas than in the wet zones. The resulting scheme is not well-balanced and violates mass conservation. Furthermore, spurious waves may emerge in the shoal zones. The technique assumes a continuous bathymetry, but a straightforward sampling of a discontinuous bathymetry can result in steep gradients of the bathymetry approximation. This will affect the CFL number [65], restricting the time steps toward very small values.

Kurganov and Petrova [18] improve the previous work by supporting a discontinuous bathymetry. They describe a reconstruction adjustment for the partially flooded cells, where values of the water depth become negative at the integration points. If the reconstructed water slope creates negative values at the integration points, they adjust the steepness of the slope so that the negative values become zero. Their correction solves the positivity problem by raising and lowering the water level at the left and right side of the cell according to the bathymetry function. This guarantees that all water heights are non-negative. However, at the partially flooded cells this can lead to large errors for small water heights and the flow velocity will grow smoothly in these formerly dry areas. Another issue related to this modification, i.e., the water climbs up on the shores at the dry/wet boundaries. Finally, if a cell becomes wet, it will almost never be completely dry again.

Bollermann [16] extends the Kurganov and Petrova [18] scheme and achieves well-balanced states in the partially flooded cells by constructing an alternative correction procedure, which is similar to the reconstruction used by Tai [70]. However, this modification works only for one dimension and can lead to infinitely small time steps. To overcome this, time step limitation, Bollermann uses the draining time technique introduced in [67]. This scheme is only one-dimensional and it is not sufficient for our simulations.

Apart from the regular grid-based schemes, various techniques exist for structured and unstructured meshes. For instance, Bryson [71] develops a well-balanced positivity preserving numerical scheme for triangular grids. This scheme can be applied to models with discontinuous bathymetry and irregular channel widths. Even though the method can be well adopted to irregular topographies, its implementation is more complex, and it has higher computational requirements than the scheme we introduce in this paper.

3 Two–Dimensional Central–Upwind Scheme

Our work is based on the two-dimensional, central-upwind scheme of Kurganov and Petrova [18]. In this section, we describe this technique for solving the SV system of shallow water equations on uniform grids. Table II.1 explains the notations used throughout the text.

We introduce a uniform grid $x_\alpha := \alpha\Delta x$ and $y_\beta := \beta\Delta y$, where Δx and Δy are small spatial scales (see Figure II.2), and we denote by $C_{j,k}$ the finite volume cells $C_{j,k} := [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}]$. The central-upwind semi-discretization (discretized only in space, while time remains continuous) of (II.2) can be written down as the following system of time-dependent, ordi-

Table II.1: Notations for the numerical scheme

$\mathbf{V}$	$\triangleq$	vector (e.g., $\mathbf{U} = [h, hu, hv]$)
$\bar{\mathbf{U}}$	$\triangleq$	cell average values
$\tilde{\mathbf{U}}$	$\triangleq$	approximated values
$\mathbf{U}_{j,k}$	$\triangleq$	vector at position $[j,k]$
$\mathbf{U}^{\pm}$	$\triangleq$	left- and right-sided point values
$\mathbf{U}^{n}$	$\triangleq$	vector at time t_n $(\mathbf{U}^n = \mathbf{U}(t_n))$
$\mathbf{H}^{\mathbf{x}}$	$\triangleq$	central-upwind flux function in x-dimension
$\mathbf{H}^{(1)}$	$\triangleq$	1st element of vector $\mathbf{H} = [\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \mathbf{H}^{(3)}]$
v	$\triangleq$	variable
v^*	$\triangleq$	special variable
$B_{j,k}$	$\triangleq$	value at position $[j,k]$

nary differential equations (ODE) [72, 17]:

$$\frac{d}{dt}\bar{\mathbf{U}}_{j,k}(t) = -\frac{\mathbf{H}^{x}_{j+\frac{1}{2},k}(t) - \mathbf{H}^{x}_{j-\frac{1}{2},k}(t)}{\Delta x} - \frac{\mathbf{H}^{y}_{j,k+\frac{1}{2}}(t) - \mathbf{H}^{y}_{j,k-\frac{1}{2}}(t)}{\Delta y} + \bar{\mathbf{S}}_{j,k}(t), \tag{II.5}$$

where $\mathbf{H}^{x}_{j\mp\frac{1}{2},k}$ and $\mathbf{H}^{y}_{j,k\mp\frac{1}{2}}$ are the central-upwind fluxes and $\bar{\mathbf{S}}_{j,k}$ is an appropriate discretization of the cell averages of the source term:

$$\bar{\mathbf{S}}_{j,k}(t) \approx \frac{1}{\Delta x \Delta y}\iint\limits_{C_{j,k}} \mathbf{S}(\mathbf{U}(x,y,t), B(x,y))dxdy, \tag{II.6}$$

which in our case only contains the bed source term. The friction term is omitted for simplification purposes without loss of generality.

We start by replacing the bathymetry function $B(x,y)$ with its continuous, piecewise bilinear approximation $\tilde{B}(x,y)$, which at each cell $C_{j,k}$ is given by a bilinear form. The vertex values $B_{j\pm\frac{1}{2},k\pm\frac{1}{2}}$ of the cell $C_{j,k}$ are computed based on the continuous bathymetry function (see Figure II.3).

The average value of $\tilde{B}$ over the cell $C_{j,k}$ is equal to its value $B_{j,k}$ at the center of this cell. Furthermore, it is equal to the average value of the values at cell interface midpoints:

$$B_{j,k} = \frac{1}{4}\left(B_{j+\frac{1}{2},k} + B_{j-\frac{1}{2},k} + B_{j,k+\frac{1}{2}} + B_{j,k-\frac{1}{2}}\right). \tag{II.7}$$

Further details on the piecewise bilinear approximation of the bathymetry can be found in [18].

Using the approximated bathymetry values at cell interface midpoints (see Figure II.4a-b) we have the discretized source terms in the following form [17, 18]:

$$\bar{\mathbf{S}}^{(1)}_{j,k}(t) := 0, \tag{II.8}$$

$$\bar{\mathbf{S}}^{(2)}_{j,k}(t) := -g\bar{h}_{j,k}\frac{B_{j+\frac{1}{2},k} - B_{j-\frac{1}{2},k}}{\Delta x}, \tag{II.9}$$

$$\bar{\mathbf{S}}^{(3)}_{j,k}(t) := -g\bar{h}_{j,k}\frac{B_{j,k+\frac{1}{2}} - B_{j,k-\frac{1}{2}}}{\Delta y}, \tag{II.10}$$

where we omit the time dependence t on the right hand side for simplification reasons without loss of generality, as we do in the following equations.

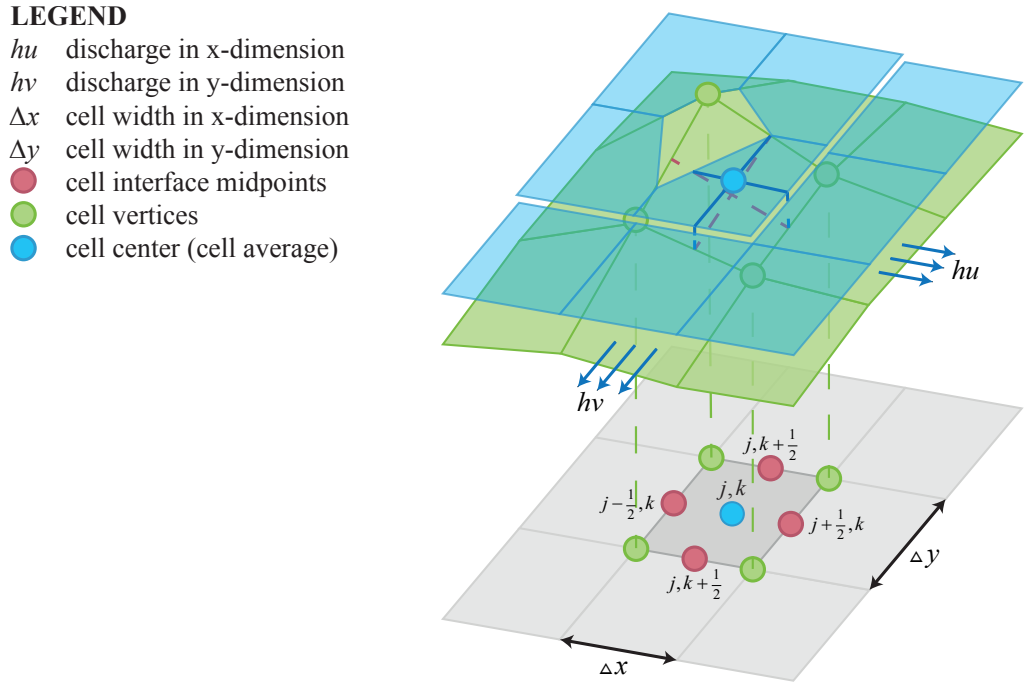


Figure II.2: Two-dimensional grid-based representation of average water elevations $\bar{w}$, discharges $\bar{hu}, \bar{hv}$, and bathymetry B. For a bilinear reconstruction, the cell averages coincide with the values at the cell centers. The bathymetry is approximated by its values at the cell vertices. In this figure, the middle cell is fully flooded in the y-dimension, while only partially flooded in the x-dimension. Waterlines are represented by the blue lines, red dashed lines mark the bathymetry slopes in both dimensions.

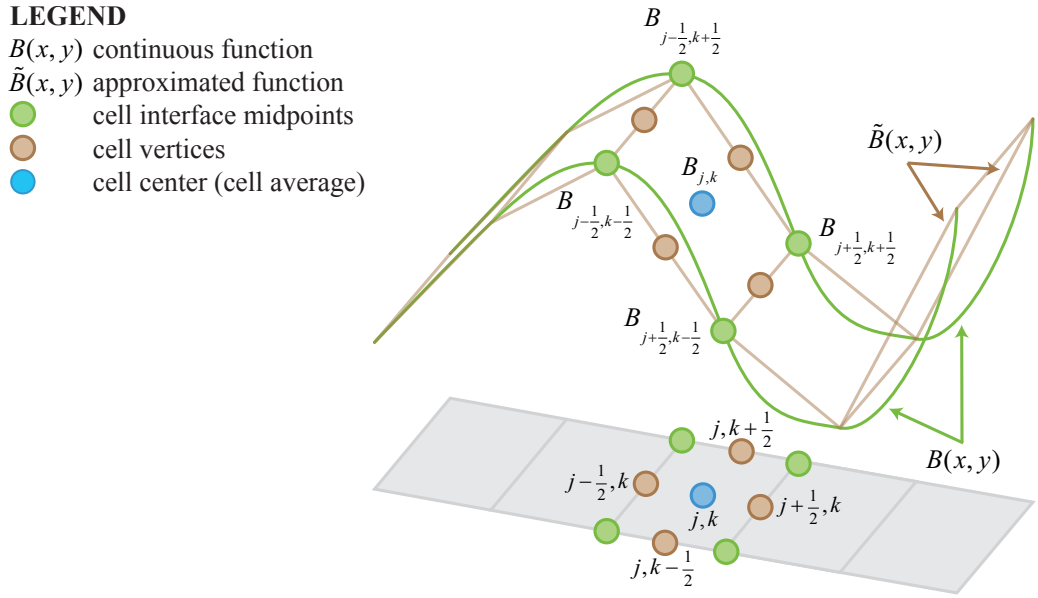


Figure II.3: Continuous bathymetry function $B(x,y)$ (green) and its piecewise linear approximation $\tilde{B}(x,y)$ (brown dots). The approximated function values equal to the continuous ones at the cell vertices (green dots). The cell average value (blue dot) equals both to the average value of the vertex values (green dots) and to the average value of the values at the cell interface midpoints (brown dots).

On the next step we reconstruct the slope of the water surface in the cells (see Figure II.4c). The reconstruction of the left- and right-sided point values (see Figure II.4d) is second-order accurate if the approximate values of the derivatives $(\mathbf{U}_x)_{j,k}$ and $(\mathbf{U}_y)_{j,k}$ are at least first-order componentwise approximations of $\mathbf{U}_x(x_j,y_k)$ and $\mathbf{U}_y(x_j,y_k)$. To preserve second-order accuracy, these values are computed using a non-linear limiter. In addition, to ensure non-oscillatory reconstruction and to avoid oscillation artifacts in the numerical solution, we use the generalized minmod limiter [73, 72, 74, 75, 76]:

$$\begin{aligned}(\mathbf{U}_x)_{j,k}&=\text{minmod}\left(\theta\frac{\bar{\mathbf{U}}_{j,k}-\bar{\mathbf{U}}_{j-1,k}}{\Delta x},\frac{\bar{\mathbf{U}}_{j+1,k}-\bar{\mathbf{U}}_{j-1,k}}{2\Delta x},\theta\frac{\bar{\mathbf{U}}_{j+1,k}-\bar{\mathbf{U}}_{j,k}}{\Delta x}\right),\\(\mathbf{U}_y)_{j,k}&=\text{minmod}\left(\theta\frac{\bar{\mathbf{U}}_{j,k}-\bar{\mathbf{U}}_{j,k-1}}{\Delta y},\frac{\bar{\mathbf{U}}_{j,k+1}-\bar{\mathbf{U}}_{j,k-1}}{2\Delta y},\theta\frac{\bar{\mathbf{U}}_{j,k+1}-\bar{\mathbf{U}}_{j,k}}{\Delta y}\right),\end{aligned}\tag{II.11}$$

where $\theta\in[1,2]$ is a parameter used to affect the numerical viscosity of the scheme. As suggested in [17], we set $\theta=1.3$, which is close to the optimal value. The minmod function is defined as

$$\text{minmod}(z_1,z_2,z_3):=\begin{cases}\min_j\{z_j\}, & \text{if } z_j>0 \quad \forall j,\\ \max_j\{z_j\}, & \text{if } z_j<0 \quad \forall j,\\ 0, & \text{otherwise,}\end{cases}\tag{II.12}$$

and is applied in a componentwise manner to all three elements $\left[\bar{w},\bar{hu},\bar{hv}\right]$ of vector $\bar{\mathbf{U}}$, where we reconstruct water levels $\bar{w}$ instead of water heights $\bar{h}$. Other non-linear limiters can be found in the literature [77, 78, 73, 74, 79, 75, 76].

The values $\mathbf{U}^{\pm}_{j+\frac12,k}=\left(w^{\pm}_{j+\frac12,k},hu^{\pm}_{j+\frac12,k},hv^{\pm}_{j+\frac12,k}\right)$ and $\mathbf{U}^{\pm}_{j,k+\frac12}=\left(w^{\pm}_{j,k+\frac12},hu^{\pm}_{j,k+\frac12},hv^{\pm}_{j,k+\frac12}\right)$ are referred to as the left- and right-sided point values (see Figure II.4d). They are obtained by the piecewise linear reconstruction $\tilde{\mathbf{U}}\equiv(\tilde{w},\tilde{hu},\tilde{hv})$ for $\mathbf{U}$ at cell interface midpoints $\left[x_{j+\frac12},y_k\right]$ and $\left[x_j,y_{k+\frac12}\right]$,

$$\tilde{\mathbf{U}}(x,y):=\bar{\mathbf{U}}_{j,k}+(\mathbf{U}_x)_{j,k}(x-x_j)+(\mathbf{U}_y)_{j,k}(y-y_k),\quad (x,y)\in C_{j,k}.\tag{II.13}$$

For cell $C_{j,k}$ we get the following four vectors describing the reconstructed point values:

$$\begin{aligned}\mathbf{U}^{-}_{j+\frac12,k}&=\mathbf{U}_{j,k}+\frac{\Delta x}{2}(\mathbf{U}_x)_{j,k},\quad \mathbf{U}^{+}_{j-\frac12,k}=\mathbf{U}_{j,k}-\frac{\Delta x}{2}(\mathbf{U}_x)_{j,k},\\ \mathbf{U}^{-}_{j,k+\frac12}&=\mathbf{U}_{j,k}+\frac{\Delta y}{2}(\mathbf{U}_y)_{j,k},\quad \mathbf{U}^{+}_{j,k-\frac12}=\mathbf{U}_{j,k}-\frac{\Delta y}{2}(\mathbf{U}_y)_{j,k}.\end{aligned}$$

We note that the reconstruction procedure (II.11)–(II.13) might produce negative water heights in the partially flooded cells [16, 18] (see Figure II.4d). Therefore, we need to correct them (see Figure II.4e). The correction technique proposed in [18] violates the well-balanced property of the scheme, and causes high velocities in these areas. Hence, we use a modified correction that has been first derived for the one-dimensional case [16]. Ensuring both well-balanced and positivity preserving properties for the two-dimensional version is not straightforward. Our new reconstruction affects only the partially flooded cells while maintaining the well-balanced property of the scheme. The derivation of the two-dimensional version is presented in Section 4.

Using the point values, we can calculate the fluxes needed for the computation of the next time step. The central-upwind numerical fluxes $\mathbf{H}^x_{j+\frac12,k}$ and $\mathbf{H}^y_{j,k+\frac12}$ (see Figure II.4f) are given by:

$$\begin{aligned}\mathbf{H}^x_{j+\frac12,k}&=\frac{a^+_{j+\frac12,k}\mathbf{F}\left(\mathbf{U}^-_{j+\frac12,k},B_{j+\frac12,k}\right)-a^-_{j+\frac12,k}\mathbf{F}\left(\mathbf{U}^+_{j+\frac12,k},B_{j+\frac12,k}\right)}{a^+_{j+\frac12,k}-a^-_{j+\frac12,k}}\\&\quad+\frac{a^+_{j+\frac12,k}a^-_{j+\frac12,k}}{a^+_{j+\frac12,k}-a^-_{j+\frac12,k}}\left[\mathbf{U}^+_{j+\frac12,k}-\mathbf{U}^-_{j+\frac12,k}\right],\end{aligned}\tag{II.14}$$

$$\mathbf{H}^{y}_{j,k+\frac{1}{2}} = \frac{b^{+}_{j,k+\frac{1}{2}}\mathbf{G}\left(\mathbf{U}^{-}_{j,k+\frac{1}{2}},B_{j,k+\frac{1}{2}}\right)-b^{-}_{j,k+\frac{1}{2}}\mathbf{G}\left(\mathbf{U}^{+}_{j,k+\frac{1}{2}},B_{j,k+\frac{1}{2}}\right)}{b^{+}_{j,k+\frac{1}{2}}-b^{-}_{j,k+\frac{1}{2}}} + \frac{b^{+}_{j,k+\frac{1}{2}}b^{-}_{j,k+\frac{1}{2}}}{b^{+}_{j,k+\frac{1}{2}}-b^{-}_{j,k+\frac{1}{2}}}\left[\mathbf{U}^{+}_{j,k+\frac{1}{2}}-\mathbf{U}^{-}_{j,k+\frac{1}{2}}\right], \tag{II.15}$$

where we use the following flux notations:

$$\mathbf{F}(\mathbf{U},B) := \left[hu, \frac{(hu)^2}{w-B}+\frac{1}{2}g(w-B)^2, huv\right]^T \tag{II.16}$$

$$\mathbf{G}(\mathbf{U},B) := \left[hv, huv, \frac{(hv)^2}{w-B}+\frac{1}{2}g(w-B)^2\right]^T. \tag{II.17}$$

We note that the central-upwind flux is a direct generalization of the well-known Harten-Lax-van Leer flux [80, 81].

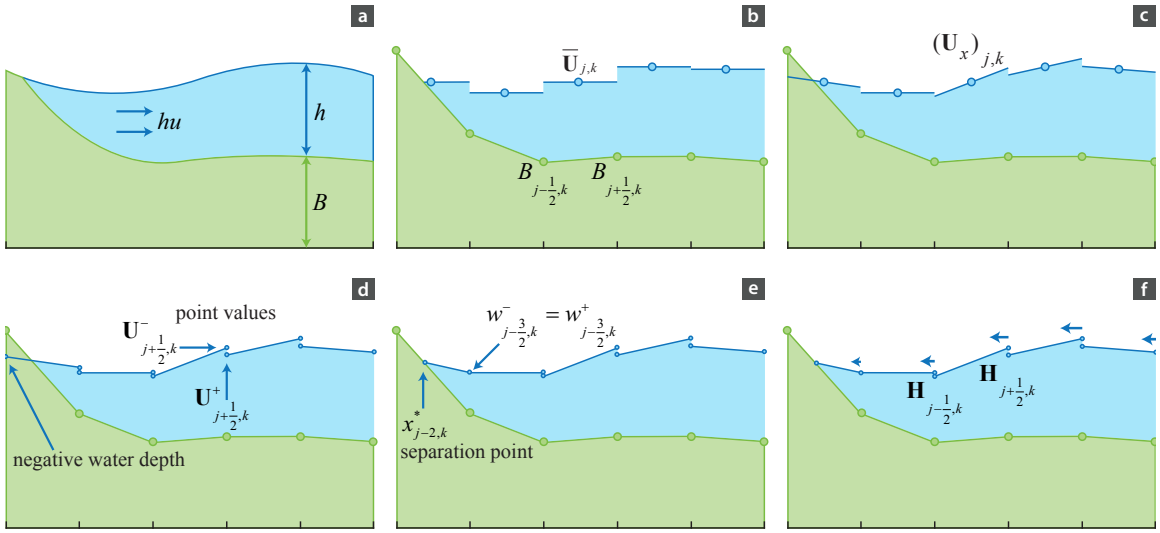


Figure II.4: (a) Schematic view of a shallow water flow at a dry/wet boundary and definition of the variables. (b) Conserved variables $\mathbf{U}$ are discretized as cell averages $\bar{\mathbf{U}}_{j,k}$. The bathymetry function B is computed at cell interface midpoints. (c) Slopes U_x are reconstructed using the minmod flux limiter. (d) Left- and right-sided point values are computed at cell interface midpoints. (e) At the almost dry cells the slope is modified to avoid negative water heights, and a separation point is generated. (f) Fluxes are computed using the central-upwind flux function at each cell interface.

The speed values $a^{\pm}_{j+\frac{1}{2},k}$ and $b^{\pm}_{j+\frac{1}{2},k}$ of propagation [72] are obtained using the eigenvalues of the Jacobian $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ as follows:

$$a^{+}_{j+\frac{1}{2},k} = \max\left\{u^{-}_{j+\frac{1}{2},k}+\sqrt{gh^{-}_{j+\frac{1}{2},k}}, u^{+}_{j+\frac{1}{2},k}+\sqrt{gh^{+}_{j+\frac{1}{2},k}}, 0\right\} \tag{II.18}$$

$$a^{-}_{j+\frac{1}{2},k} = \min\left\{u^{-}_{j+\frac{1}{2},k}-\sqrt{gh^{-}_{j+\frac{1}{2},k}}, u^{+}_{j+\frac{1}{2},k}-\sqrt{gh^{+}_{j+\frac{1}{2},k}}, 0\right\} \tag{II.19}$$

$$b^{+}_{j,k+\frac{1}{2}} = \max\left\{v^{-}_{j,k+\frac{1}{2}}+\sqrt{gh^{-}_{j,k+\frac{1}{2}}}, v^{+}_{j,k+\frac{1}{2}}+\sqrt{gh^{+}_{j,k+\frac{1}{2}}}, 0\right\} \tag{II.20}$$

$$b^{-}_{j,k+\frac{1}{2}} = \min\left\{v^{-}_{j,k+\frac{1}{2}}-\sqrt{gh^{-}_{j,k+\frac{1}{2}}}, v^{+}_{j,k+\frac{1}{2}}-\sqrt{gh^{+}_{j,k+\frac{1}{2}}}, 0\right\} \tag{II.21}$$

Using the semi-discrete equation (II.5) together with the forward Euler temporal discretization we obtain the discrete equation for the computation of the next time step as:

$$\bar{\mathbf{U}}^{n+1}_{j,k} = \bar{\mathbf{U}}^{n}_{j,k} - \lambda\left(\mathbf{H}^{x}_{j+\frac{1}{2},k}-\mathbf{H}^{x}_{j-\frac{1}{2},k}\right) - \mu\left(\mathbf{H}^{y}_{j,k+\frac{1}{2}}-\mathbf{H}^{y}_{j,k-\frac{1}{2}}\right), \tag{II.22}$$

where $\lambda := \Delta t/\Delta x$, $\mu := \Delta t/\Delta y$, and the numerical fluxes $\mathbf{H}^x_{j\pm\frac{1}{2},k}$ and $\mathbf{H}^y_{j,k\pm\frac{1}{2}}$ are evaluated at time $t = t^n$. In order to keep the numerical integration stable, the time step size Δt has to satisfy the CFL condition:

$$\Delta t \le \mathbf{CFL} \min\left\{\frac{\Delta x}{a}, \frac{\Delta y}{b}\right\}, \tag{II.23}$$

where $\mathbf{CFL} := \frac{1}{4}$ for the two-dimensional scheme, and a and b are given by

$$a := \max_{j,k}\left\{\max\{a^+_{j+\frac{1}{2},k}, -a^-_{j+\frac{1}{2},k}\}\right\}, \quad b := \max_{j,k}\left\{\max\{b^+_{j,k+\frac{1}{2}}, -b^-_{j,k+\frac{1}{2}}\}\right\}. \tag{II.24}$$

4 Reconstruction at Partially Flooded Cells

The central-upwind scheme described in the previous section may produce negative water values in the partially flooded cells on the reconstruction step (see Figure II.5a). Even if the total amount of water in the cell is positive ($\bar{w}_{j,k} > B_{j,k}$), the water level in the cell may intersect the bathymetry ($\bar{w}_{j,k} < B_{j-\frac{1}{2},k}$) and thus the point value at the cell interface becomes negative ($h_{j-\frac{1}{2},k} < 0$). One could try replacing the first-order, piecewise, constant reconstruction with a higher order, piecewise, linear reconstruction, but it will not guarantee positive reconstructed point values at the cell interfaces. Therefore, we need to correct these point values. The correction proposed in [18] solves the problem of the negative point values, but violates the well-balanced property of the scheme (see Figure II.5b). Based on the modified technique proposed in [16] for the one-dimensional scheme (see Figure II.5c), we extend this correction for two dimensions.

We assume that at a certain time t all computed water levels are higher or equal to the bathymetry elevation ($\bar{w}_{j,k} \ge B_{j,k}$). In addition, in the piecewise linear reconstruction (II.13), we use a non-linear limiter to compute the slopes $(\mathbf{U}_x)_{j,k}$ and $(\mathbf{U}_y)_{j,k}$. We also assume that at an arbitrary partially flooded cell $C_{j,k}$,

$$B_{j-\frac{1}{2},k} > \bar{w}_{j,k} > B_{j+\frac{1}{2},k} \quad \text{or} \quad B_{j,k-\frac{1}{2}} > \bar{w}_{j,k} > B_{j,k+\frac{1}{2}} \tag{II.25}$$

and that the reconstructed point values of w in cell $C_{j+1,k}$ and $C_{j,k+1}$ satisfy

$$\begin{aligned} w^+_{j+\frac{1}{2},k} &> B_{j+\frac{1}{2},k} \quad \text{and} \quad w^-_{j+\frac{3}{2},k} > B_{j+\frac{3}{2},k}, \\ w^+_{j,k+\frac{1}{2}} &> B_{j,k+\frac{1}{2}} \quad \text{and} \quad w^-_{j,k+\frac{3}{2}} > B_{j,k+\frac{3}{2}}. \end{aligned} \tag{II.26}$$

Symmetric cases can be treated the same way.

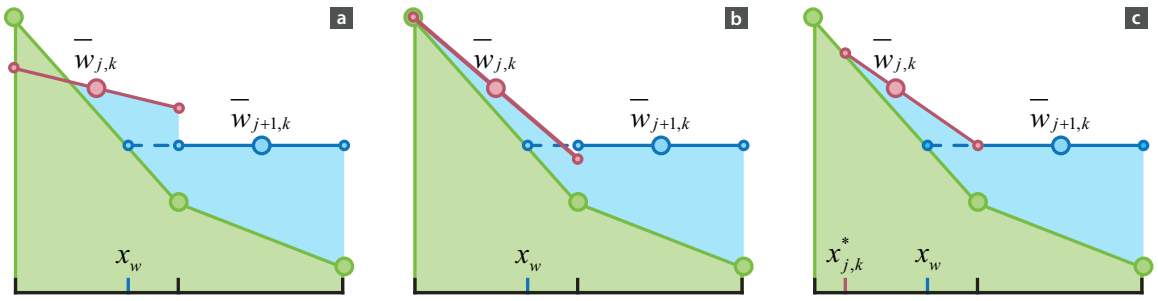


Figure II.5: Approximations of the wet/dry front reconstruction. The blue dashed line represents the waterline of the fully flooded cell. (a) Wrong approximation by the piecewise linear reconstruction, which produces a negative water value. (b) Positivity preserving, but unbalanced piecewise linear reconstruction. (c) Positivity preserving, well-balanced, piecewise linear reconstruction.

We start with computing the water surface $w_{j,k}$ in cell $C_{j,k}$. In order to distinguish between the two dimensions, $w^x_{j,k}$ marks the waterline for the x- and $w^y_{j,k}$ for the y-dimension. The water level is the average elevation of the water surface in a cell. The waterline is a horizontal line which represents the real water surface in a cell computed from the amount of water present in that cell, which means it can differ from the average water elevation. If the cell is fully flooded, we can represent the line by a linear function (see Figure II.6a). Otherwise, it has to be represented by a piecewise linear function, with a separation point $x^*_{j,k}$, which defines the location where the water height becomes zero (see Figure II.5c). We choose a water height in a way that the volume enclosed between the surface and the bathymetry equals to the amount of water in that cell (see Figure II.6b). The amount of water in cell $C_{j,k}$ is defined by $\Delta x \cdot \bar{h}_{j,k}$ and $\Delta y \cdot \bar{h}_{j,k}$ for each direction respectively, where $\bar{h}_{j,k} := \bar{w}_{j,k} - B_{j,k}$. These areas can be represented by trapezoids, if the cell is fully filled, or by triangular shapes, if the cell is partially flooded (see Figure II.6). We note that a cell can be fully flooded in one dimension, while only partially flooded in the other. If cell $C_{j,k}$ is fully flooded, the following conditions hold:

$$\bar{h}_{j,k} \geq \frac{\Delta x}{2}|(B_x)_{j,k}|, \quad \bar{h}_{j,k} \geq \frac{\Delta y}{2}|(B_y)_{j,k}|, \tag{II.27}$$

where $\bar{h}_{j,k}$ is the average water height of the cell, $(B_x)_{j,k}$ and $(B_y)_{j,k}$ are the batrymetry slopes in x and y directions. In this case, functions $s^x_{j,k}(x,y)$ and $s^y_{j,k}(x,y)$ represent the water surface for both dimensions:

$$s^x_{j,k}(x,y) = \bar{w}_{j,k}, \quad s^y_{j,k}(x,y) = \bar{w}_{j,k},$$

otherwise the free surface is a continuous piecewise linear function given by

$$s^x_{j,k}(x,y) = \begin{cases} B_{j,k}(x,y), & \text{if } x < x^*_w, \\ w^x_{j,k}, & \text{otherwise,} \end{cases} \quad s^y_{j,k}(x,y) = \begin{cases} B_{j,k}(x,y), & \text{if } y < y^*_w, \\ w^y_{j,k}, & \text{otherwise,} \end{cases} \tag{II.28}$$

where x^*_w and y^*_w are the boundary points separating the dry and wet parts in the cell $C_{j,k}$, and $B_{j,k}(x,y)$ is a two-dimensional plane representing the bathymetry approximation in the cell. The boundary separation points can be determined by the following equations:

$$\Delta x^*_w = \sqrt{\frac{2\Delta x \bar{h}_{j,k}}{|(B_x)_{j,k}|}} \quad \Delta y^*_w = \sqrt{\frac{2\Delta y \bar{h}_{j,k}}{|(B_y)_{j,k}|}} \tag{II.29}$$

where $\Delta x^*_w = x_{j+\frac{1}{2}} - x^*_w$ and $\Delta y^*_w = y_{j+\frac{1}{2}} - y^*_w$. More details on deriving this equation can be found in [16, 82]. The average total elevation of the water surface for the wet/dry cell is then computed as

$$\begin{aligned} w^x_{j,k} &= B_{j,k} + \left(\Delta x^*_w - \frac{\Delta x}{2}\right)|(B_x)_{j,k}| \\ w^y_{j,k} &= B_{j,k} + \left(\Delta y^*_w - \frac{\Delta y}{2}\right)|(B_y)_{j,k}| \end{aligned} \tag{II.30}$$

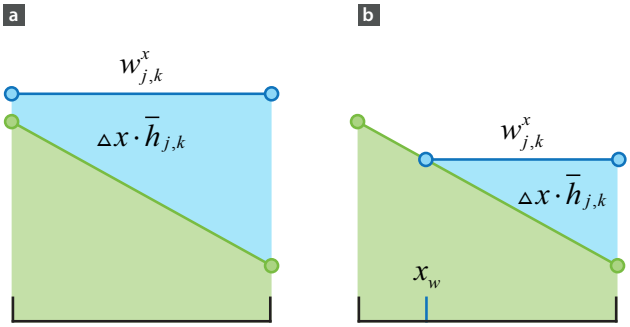


Figure II.6: Waterline $w^x_{j,k}$ computation using the conservation of the average water height $\bar{h}_{j,k}$, where $\Delta x \cdot \bar{h}_{j,k}$ equals to the amount of water in the cell. (a) In the fully flooded cell, the waterline does not intersect the bathymetry. (b) In the partially flooded cell, x_w marks the intersection point between the waterline and the bathymetry.

Note that if a cell satisfies at least one condition of (II.27), then it is a partially flooded cell with $\Delta x^*_w < \Delta x$ or $\Delta y^*_w < \Delta y$.

In the following, we discuss the reconstruction of the waterline for the x dimension. It can be computed analogously for the y dimension. At this point we know which cells are partially flooded. For these cells we modify the reconstruction of the water height h to ensure the well-balanced property. We assign the value of the reconstructed water height of the next fully flooded cell $C_{j+1,k}$ to the value of the same interface at the partially flooded cell $C_{j,k}$, $w^-_{j+\frac{1}{2},k} := w^+_{j+\frac{1}{2},k}$. Using this value and the conservation of the average water height $\bar{h}_{j,k}$, we determine the waterline $w^x_{j,k}$ in cell $C_{j,k}$. This assignment implies that $h^-_{j+\frac{1}{2},k} := w^-_{j+\frac{1}{2},k} - B_{j+\frac{1}{2},k} = w^+_{j+\frac{1}{2},k} - B_{j+\frac{1}{2},k} =: h^+_{j+\frac{1}{2},k}$.

If the amount of water in cell $C_{j,k}$ is sufficiently large, then $h^+_{j-\frac{1}{2},k} \geq 0$ and satisfies

$$\bar{h}_{j,k} = \frac{1}{2}\left(h^-_{j+\frac{1}{2},k} + h^+_{j-\frac{1}{2},k}\right), \tag{II.31}$$

from which we obtain the total water level $w^+_{j-\frac{1}{2},k} = h^+_{j-\frac{1}{2},k} + B_{j-\frac{1}{2},k}$, and thus the well-balanced reconstruction for cell $C_{j,k}$ is completed.

If the value of $h^+_{j-\frac{1}{2},k}$ at the other interface computed from the conservation requirement (II.31) is negative, we replace the waterline $w^x_{j,k}$ in cell $C_{j,k}$ with two linear pieces. The breaking point between the "wet" and "dry" pieces is marked by $x^*_{j,k}$ and is determined by the conservation requirement, which in this case reads

$$\Delta x \cdot \bar{h}_{j,k} = \frac{\Delta x^*_{j,k}}{2} h^-_{j+\frac{1}{2},k}, \tag{II.32}$$

where

$$\Delta x^*_{j,k} = |x_{j+\frac{1}{2},k} - x^*_{j,k}|.$$

Using the idea from [16] and combining the above two cases, we obtain the reconstructed value

$$h^+_{j-\frac{1}{2},k} = \max\left\{0, 2\bar{h}_{j,k} - h^-_{j+\frac{1}{2},k}\right\}. \tag{II.33}$$

As in [16], we also generalize the definition of $\Delta x^*_{j,k}$ and set

$$\Delta x^*_{j,k} = \Delta x \cdot \min\left\{\frac{2\bar{h}_{j,k}}{h^-_{j+\frac{1}{2},k}}, 1\right\}. \tag{II.34}$$

The well-balanced reconstruction for the x dimension is completed. Following this technique, one can easily derive it for the second dimension.

We mentioned that the piecewise linear reconstuction (II.12)–(II.13) does not guarantee the positivity of the point values at the dry/wet fronts, thus they have to be corrected. This approach only modifies the reconstructed water values $\tilde{w}^\pm_{j\pm\frac{1}{2},k}$. Next, as in [16] we summarize the possible cases of the modified reconstruction, where $\bar{w}_{j,k}$ is the average water level in the cell, $\tilde{w}^\pm_{j\pm\frac{1}{2},k}$ are the point values reconstructed using the minmod flux limiter (II.11), and $w^\pm_{j\pm\frac{1}{2},k}$ are the corrected values:

Case 1. If $\bar{w}_{j,k} > B_{j-\frac{1}{2},k}$ and $\bar{w}_{j,k} > B_{j+\frac{1}{2},k}$ then cell $C_{j,k}$ is fully flooded.

1A. If $\tilde{w}^+_{j-\frac{1}{2},k} \geq B_{j-\frac{1}{2},k}$ and $\tilde{w}^-_{j+\frac{1}{2},k} \geq B_{j+\frac{1}{2},k}$ the cell is flooded and we set $w^\pm_{j+\frac{1}{2},k} := \tilde{w}^\pm_{j+\frac{1}{2},k}$ (see Figure II.7a).

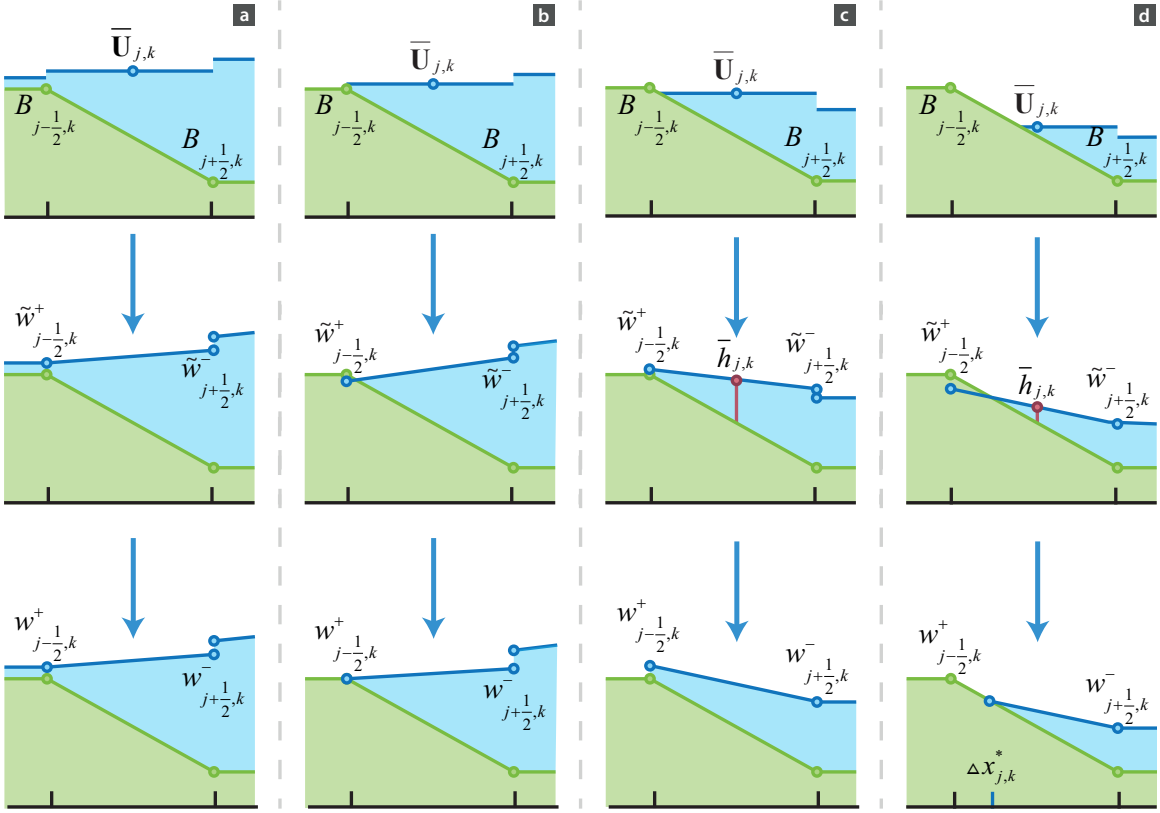


Figure II.7: Illustrations of reconstruction cases for the wet/dry fronts. The upper row shows the average water levels in the cells, the row in the middle shows the reconstructed point values, and the bottom row shows the modified point values. (a) The amount of water is enough to fill the cell, the reconstruction is correct. (b) The amount of water is enough to fill the cell, but a negative point value was produced, therefore we set it to zero, and the value on the right side requires correction due to the conservation criterion. (c) The cell is partially flooded, and after equalizing the water height between the current and the next fully flooded cell, both values become positive. (d) The cell is partially flooded, and there is not enough water to fill it after the equalization.

1B. Otherwise, as in [18], we redistribute the water (see Figure II.7b) in the following way:

$$\text{If} \quad \tilde{w}^{+}_{j-\frac{1}{2},k} < B_{j-\frac{1}{2},k}, \quad \text{then set} \quad (w_x)_{j,k} := \frac{\bar{w}_{j,k} - B_{j-\frac{1}{2},k}}{\Delta x/2},$$
$$\Longrightarrow \; w^{-}_{j+\frac{1}{2},k} = 2\bar{w}_{j,k} - B_{j-\frac{1}{2},k}, \quad w^{+}_{j-\frac{1}{2},k} = B_{j-\frac{1}{2},k}.$$

and symmetrically

$$\text{If} \quad \tilde{w}^{-}_{j+\frac{1}{2},k} < B_{j+\frac{1}{2},k}, \quad \text{then set} \quad (w_x)_{j,k} := \frac{B_{j+\frac{1}{2},k} - \bar{w}_{j,k}}{\Delta x/2},$$
$$\Longrightarrow \; w^{-}_{j+\frac{1}{2},k} = B_{j+\frac{1}{2},k}, \quad w^{+}_{j-\frac{1}{2},k} = 2\bar{w}_{j,k} - B_{j+\frac{1}{2},k},$$

Case 2. If $B_{j-\frac{1}{2},k} > \bar{w}_{j,k} > B_{j+\frac{1}{2},k}$, then cell $C_{j,k}$ is possibly partially flooded.

2A. If $\tilde{w}^{+}_{j+\frac{1}{2},k} > B_{j+\frac{1}{2},k}$ and $\tilde{w}^{-}_{j+\frac{3}{2},k} > B_{j+\frac{3}{2},k}$, then cell $C_{j+1,k}$ is fully flooded and $w^{+}_{j+\frac{1}{2},k} = \tilde{w}^{+}_{j+\frac{1}{2},k}$. We set $w^{-}_{j+\frac{1}{2},k} := w^{+}_{j+\frac{1}{2},k}$ and $h^{-}_{j+\frac{1}{2},k} := w^{-}_{j+\frac{1}{2},k} - B_{j+\frac{1}{2},k}$.

2A1. If $2\bar{h}_{j,k} - h^{-}_{j+\frac{1}{2},k} \geq 0$, then the amount of water in cell $C_{j,k}$ is sufficiently large, and we set $h^{+}_{j-\frac{1}{2},k} = 2\bar{h}_{j,k} - h^{-}_{j+\frac{1}{2},k}$, so $w^{+}_{j-\frac{1}{2},k} = h^{+}_{j-\frac{1}{2},k} + B_{j-\frac{1}{2},k}$ (see Figure II.7c).

2A2. Otherwise set $h^{-}_{j+\frac{1}{2},k} = 0$, $w^{+}_{j-\frac{1}{2},k} = B_{j-\frac{1}{2},k}$ and $\Delta x^{*}_{j,k}$ as in (II.34) (see Figure II.7d).

2B. Otherwise set $h^{-}_{j+\frac{1}{2},k} := w_{j,k} - B_{j+\frac{1}{2},k}$ and $\Delta x^{*}_{j,k} := \Delta x^{*}_{w}$. This situation is not generic and may occur only in the under-resolved computations [16].

Case 3. $B_{j-\frac{1}{2},k} < \bar{w}_{j,k} < B_{j+\frac{1}{2},k}$ is analogous to Case 2.

The correction cases for the second dimension can be derived analogously by changing indices from $j-\frac{1}{2},k$ and $j+\frac{1}{2},k$ to $j,k-\frac{1}{2}$ and $j,k-\frac{1}{2}$.

At this point, the corrected water heights of the reconstructed point values are non-negative. However, they may be very small or even zero. Since $u = \frac{hu}{h}$ and $v = \frac{hv}{h}$, these computations may lead to large errors in the partially flooded cells for small water heights and they have a singularity at zero water height ($h = 0$). To deal with this problem, we use the desingularization suggested in [18]:

$$u = \frac{\sqrt{2}h(hu)}{\sqrt{h^4 + \max(h^4, \epsilon)}}, \quad v = \frac{\sqrt{2}h(hv)}{\sqrt{h^4 + \max(h^4, \epsilon)}}, \tag{II.35}$$

where ϵ is a small apriori chosen positive number. This has a dampening effect on the velocities as the water height approaches zero. Determining a proper value for ϵ is very difficult. High values lead to large errors in the simulation results, while low values give small time steps. In our simulation, we used the suggestion of Brodtkorb [39]:

$$\epsilon = E_0 \max\left(1, \min\left(\Delta x, \Delta y\right)\right), \tag{II.36}$$

with $E_0 = 10^{-2}$ m.

5 Positivity Preserving in Time Integration

In the previous section, we have discussed the well-balanced spatial reconstruction of the point values for partially flooded cells. In this section, we continue with the time-quadrature for the fluxes at partially flooded cells. This means the discretization of the semi-discrete scheme in time, and advancing by Δt. We follow the technique used in [16] and start with modifying the time integration of the water height so that it remains positive after its computation. Assuming that

the water height is positive for all $\bar{h}^n_{j,k}$ at time step n, it has to remain positive for the next time step $n+1$:

$$\bar{h}^{n+1}_{j,k} = \bar{h}^n_{j,k} - \Delta t \frac{(\mathbf{H}^x)^{(1)}_{j+\frac{1}{2},k} - (\mathbf{H}^x)^{(1)}_{j-\frac{1}{2},k}}{\Delta x} - \Delta t \frac{(\mathbf{H}^y)^{(1)}_{j,k+\frac{1}{2}} - (\mathbf{H}^y)^{(1)}_{j,k-\frac{1}{2}}}{\Delta y} \geq 0, \tag{II.37}$$

where we use (II.22) and subtract $B_{j,k}$ from both sides, since $\bar{w}_{j,k} = \bar{h}_{j,k} + B_{j,k}$.

Using (II.37) we introduce the draining time step [67] for the two-dimensional scheme:

$$\Delta t^{drain}_{j,k} = \frac{\Delta x \Delta y \bar{h}^n_{j,k}}{\Delta y \left((\mathbf{H}^x)^{(1)}_{j+\frac{1}{2},k} + (\mathbf{H}^x)^{(1)}_{j+\frac{1}{2},k} \right) + \Delta x \left((\mathbf{H}^y)^{(1)}_{j,k+\frac{1}{2}} + (\mathbf{H}^y)^{(1)}_{j,k+\frac{1}{2}} \right)}, \tag{II.38}$$

which describes how long it takes for cell $C_{j,k}$ to become dry due to the outflow fluxes. Now we modify the evolution step (II.22) using the draining time step:

$$\bar{h}^{n+1}_{j,k} = \bar{h}^n_{j,k} - \Delta t_{j,k} \frac{(\mathbf{H}^x)^{(1)}_{j+\frac{1}{2},k} - (\mathbf{H}^x)^{(1)}_{j-\frac{1}{2},k}}{\Delta x} - \Delta t_{j,k} \frac{(\mathbf{H}^y)^{(1)}_{j,k+\frac{1}{2}} - (\mathbf{H}^y)^{(1)}_{j,k-\frac{1}{2}}}{\Delta y}, \tag{II.39}$$

where the time steps at the cell interfaces are computed by:

$$\Delta t_{j,k} = \begin{cases} \min(\Delta t, \Delta t^{drain}_{j,k}), & \text{if } \nabla \cdot \mathbf{H}_{j,k} > 0, \\ \Delta t, & \text{if } \nabla \cdot \mathbf{H}_{j,k} \leq 0, \end{cases} \tag{II.40}$$

which means that the draining time step is only used if the water height decreases during the integration and the cell is at risk of drying out. If the flux divergence $\nabla \cdot \mathbf{H}_{j,k} > 0$ for a fully flooded cell $C_{j,k}$, then the draining time can have higher values than the global time step and we select the smallest of the two. If $\nabla \cdot \mathbf{H}_{j,k} \leq 0$, i.e., there is more water entering than leaving the cell, then we use the global time step. This means that the draining time step is applied only to the partially flooded cell in case of the positive divergence. Hence, the time evolution of the fully flooded cells remains as in [18].

Since we use the global time step Δt for the source terms $\bar{S}^{(2)}_{j,k}$ and $\bar{S}^{(3)}_{j,k}$ in the time evolution step, we have to split the momentum fluxes $F^{(2)}(\mathbf{U}, B)$ and $G^{(3)}(\mathbf{U}, B)$ into advection (a) and gravity (g) driven parts to ensure the well-balanced property:

$$\begin{aligned} F^{(2),a}(\mathbf{U}, B) &:= \tfrac{(hu)^2}{w-B} \quad \text{and} \quad F^{(2),g}(\mathbf{U}, B) := \frac{g}{2}(w-B)^2 \\ G^{(3),a}(\mathbf{U}, B) &:= \tfrac{(hv)^2}{w-B} \quad \text{and} \quad G^{(3),g}(\mathbf{U}, B) := \frac{g}{2}(w-B)^2 \end{aligned}$$

Due to this modification, the source terms exactly balance the gravity-driven part of the flux. Using the split fluxes, the modified central-upwind fluxes read:

$$\begin{aligned} \mathbf{H}^{(3),g}_{j,k+\frac{1}{2}}(t) = &\frac{a^+_{j,k+\frac{1}{2}} \mathbf{G}^{(3),g}\left(\mathbf{U}^-_{j,k+\frac{1}{2}}\right) - a^-_{j,k+\frac{1}{2}} \mathbf{G}^{(3),g}\left(\mathbf{U}^+_{j,k+\frac{1}{2}}\right)}{a^+_{j,k+\frac{1}{2}} - a^-_{j,k+\frac{1}{2}}} \\ &+ \frac{a^+_{j,k+\frac{1}{2}} a^-_{j,k+\frac{1}{2}}}{a^+_{j,k+\frac{1}{2}} - a^-_{j,k+\frac{1}{2}}} \left[\mathbf{U}^{(2),+}_{j,k+\frac{1}{2}} - \mathbf{U}^{(2),-}_{j,k+\frac{1}{2}} \right], \end{aligned} \tag{II.41}$$

and

$$\mathbf{H}^{(3),a}_{j,k+\frac{1}{2}}(t) = \frac{a^+_{j,k+\frac{1}{2}} \mathbf{G}^{(3),a}\left(\mathbf{U}^-_{j,k+\frac{1}{2}}\right) - a^-_{j,k+\frac{1}{2}} \mathbf{G}^{(3),a}\left(\mathbf{U}^+_{j,k+\frac{1}{2}}\right)}{a^+_{j,k+\frac{1}{2}} - a^-_{j,k+\frac{1}{2}}}. \tag{II.42}$$

The modified central-upwind fluxes $\mathbf{H}^{(2),a}_{j+\frac{1}{2},k}(t)$ and $\mathbf{H}^{(2),g}_{j+\frac{1}{2},k}(t)$ can be derived analogously, and their one-dimensional forms can be found in [16], which are very similar to the two-dimensional ones.

Using the new fluxes, we get the new update of discharges hu and hv:

$$\begin{aligned}\bar{hu}^{n+1}_{j,k} =& \bar{hu}^{n}_{j,k} - \Delta t \bar{S}^{(2),n}_{j,k} - \Delta t \frac{(\mathbf{H}^x)^{(2),g}_{j+\frac{1}{2},k} - (\mathbf{H}^x)^{(2),g}_{j-\frac{1}{2},k}}{\Delta x} \\ & - \Delta t_{j,k} \left(\frac{(\mathbf{H}^y)^{(2)}_{j,k+\frac{1}{2}} - (\mathbf{H}^y)^{(2)}_{j,k-\frac{1}{2}}}{\Delta y} - \frac{(\mathbf{H}^x)^{(2),a}_{j+\frac{1}{2},k} - (\mathbf{H}^x)^{(2),a}_{j-\frac{1}{2},k}}{\Delta x} \right)\end{aligned} \tag{II.43}$$

$$\begin{aligned}\bar{hv}^{n+1}_{j,k} =& \bar{hv}^{n}_{j,k} - \Delta t \bar{S}^{(3),n}_{j,k} - \Delta t \frac{(\mathbf{H}^y)^{(3),g}_{j,k+\frac{1}{2}} - (\mathbf{H}^y)^{(3),g}_{j,k-\frac{1}{2}}}{\Delta y} \\ & - \Delta t_{j,k} \left(\frac{(\mathbf{H}^x)^{(3)}_{j+\frac{1}{2},k} - (\mathbf{H}^x)^{(3)}_{j-\frac{1}{2},k}}{\Delta x} - \frac{(\mathbf{H}^y)^{(3),a}_{j,k+\frac{1}{2}} - (\mathbf{H}^y)^{(3),a}_{j,k-\frac{1}{2}}}{\Delta y} \right)\end{aligned} \tag{II.44}$$

This new finite volume scheme consisting of (II.39), (II.43) and (II.44) is both well-balanced and positivity preserving even in the presence of partially flooded cells.

As in [16] for the one-dimensional scheme, we prove that the new two-dimensional central-upwind finite volume scheme remains well-balanced for both "lake at rest" and "dry lake" states.

Theorem 5.1. *Consider the system* (II.1) *and the fully discrete central-upwind scheme* (II.39), (II.43) *and* (II.44). *Assume that the numerical solution* $\mathbf{U}(t^n)$ *corresponds to the steady state which is a combination of the "lake at rest"* (II.3) *and "dry lake"* (II.4) *states in the sense that* $w_{j,k} = Const.$ *and* $u = 0, v = 0$ *whenever* $h_{j,k} > 0$. *Then* $\mathbf{U}(t^{n+1}) = \mathbf{U}(t^n)$, *that is, the scheme is well-balanced.*

Proof. We have to show that in all cells the fluxes and the source term discretization cancel exactly. First, we mention the fact that the reconstruction procedure derived in Section 4 preserves both the "lake at rest" and "dry lake" steady states and their combinations. For all cells where the original reconstruction is not corrected, the resulting slopes are zero and therefore the reconstructed point values equal to the average water level there, $w^{\mp}_{j\pm\frac{1}{2}} = w_{j,k}$. As $hu = 0$ and $hv = 0$ in all cells, the reconstructions for hu and hv reproduce the constant point values $(hu)^{\mp}_{j\pm\frac{1}{2},k} = (hu)^{\mp}_{j,k\pm\frac{1}{2}} = (hv)^{\mp}_{j\pm\frac{1}{2},k} = (hv)^{\mp}_{j,k\pm\frac{1}{2}} = 0, \forall j,k$. Thus, the draining time is equal to the global time step, i.e., $\Delta t^{drain}_{j,k} = \Delta t$.

First, we show that the update of the water levels satisfy the criteria above:

$$\begin{aligned}\mathbf{H}^{(1),x}_{j+\frac{1}{2},k} =& \frac{a^+_{j+\frac{1}{2},k}(hu)^-_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}(hu)^+_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} \\ & + \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} \left[(h+B)^+_{j+\frac{1}{2},k} - (h+B)^-_{j+\frac{1}{2},k} \right] = 0\end{aligned}$$

as $B^+_{j+\frac{1}{2},k} = B^-_{j+\frac{1}{2},k}$, $h^+_{j+\frac{1}{2},k} = h^-_{j+\frac{1}{2},k}$ and $(hu)^+_{j+\frac{1}{2}} = (hu)^-_{j+\frac{1}{2},k} = 0$. The same holds for $\mathbf{H}^{(1),y}_{j,k+\frac{1}{2}}$ with hv. From this we get:

$$\bar{w}^{n+1}_{j,k} = \bar{h}^{n+1}_{j,k} + B_{j,k} = \bar{h}^{n}_{j,k} + B_{j,k} = \bar{w}^{n}_{j,k}.$$

Second, we analyze the update of the discharge using (II.43) and (II.44). Using the same argument and setting $u^{\pm}_{j+\frac{1}{2},k} = u^{\pm}_{j,k+\frac{1}{2}} = v^{\pm}_{j+\frac{1}{2},k} = v^{\pm}_{j,k+\frac{1}{2}} = 0$ at the points $p^x = p_{j+\frac{1}{2},k}$ and $p^y = p_{j,k+\frac{1}{2}}$ where $h^+_{j+\frac{1}{2},k} = h^-_{j+\frac{1}{2},k} = h^+_{j,k+\frac{1}{2}} = h^-_{j,k+\frac{1}{2}} = 0$, for the second and third component we obtain:

$$\begin{aligned}
\mathbf{H}^{(2),a,x}_{j+\frac{1}{2},k} + \mathbf{H}^{(2),g,x}_{j+\frac{1}{2},k} &= \frac{a^+_{j+\frac{1}{2},k}(hu)^-_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}(hu)^+_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} + \frac{a^+_{j+\frac{1}{2},k}(\frac{g}{2}h^2)^-_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}(\frac{g}{2}h^2)^+_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} \\
&\quad + \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}\left[(hu)^+_{j+\frac{1}{2},k} - (hu)^-_{j+\frac{1}{2},k}\right] \\
&= \frac{\left(u_{j+\frac{1}{2},k} + \sqrt{gh_{j+\frac{1}{2},k}}\right)(\frac{g}{2}h^2)^-_{j+\frac{1}{2},k} - \left(u_{j+\frac{1}{2},k} - \sqrt{gh_{j+\frac{1}{2},k}}\right)(\frac{g}{2}h^2)^+_{j+\frac{1}{2},k}}{\left(u_{j+\frac{1}{2},k} + \sqrt{gh_{j+\frac{1}{2},k}}\right) - \left(u_{j+\frac{1}{2},k} - \sqrt{gh_{j+\frac{1}{2},k}}\right)} \\
&= \frac{g}{2}h^2_{j+\frac{1}{2},k} \qquad \text{(II.45)} \\
\mathbf{H}^{(2),y}_{j+\frac{1}{2},k} &= \frac{a^+_{j+\frac{1}{2},k}(huv)^-_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}(huv)^+_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} \\
&\quad + \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}\left[(huv)^+_{j+\frac{1}{2},k} - (huv)^-_{j+\frac{1}{2},k}\right] = 0, \qquad \text{(II.46)}
\end{aligned}$$

where

$$\begin{aligned}
h_{j+\frac{1}{2},k} &:= h^+_{j+\frac{1}{2},k} = h^-_{j+\frac{1}{2},k} \quad \text{and} \quad h_{j,k+\frac{1}{2}} := h^+_{j,k+\frac{1}{2}} = h^-_{j,k+\frac{1}{2}}, \\
u_{j+\frac{1}{2},k} &:= u^+_{j+\frac{1}{2},k} = u^-_{j+\frac{1}{2},k} \quad \text{and} \quad u_{j,k+\frac{1}{2}} := u^+_{j,k+\frac{1}{2}} = u^-_{j,k+\frac{1}{2}}.
\end{aligned}$$

Following the same rules, it can straightforwardly be proven that $\mathbf{H}^{(3),a,y}_{j,k+\frac{1}{2}} + \mathbf{H}^{(3),g,y}_{j,k+\frac{1}{2}} = \frac{g}{2}h^2_{j,k+\frac{1}{2}}$ and $\mathbf{H}^{(3),x}_{j,k+\frac{1}{2}} = 0$. Therefore, the finite update (II.43) and (II.44) for the studied steady state after substituting the source quadrature (II.9)–(II.10) reads as,

$$\begin{aligned}
(\bar{hu})^{n+1}_{j,k} &= (\bar{hu})^n_{j,k} - \frac{\Delta t}{\Delta x}\left[\frac{g}{2}\left(h_{j+\frac{1}{2},k}\right)^2 - \frac{g}{2}\left(h_{j-\frac{1}{2},k}\right)^2\right] + \Delta t\,\bar{\mathbf{S}}^{(2),n}_{j,k} \\
&= (\bar{hu})^n_{j,k} - \frac{\Delta t}{\Delta x}\left[\frac{g}{2}\left(h_{j+\frac{1}{2},k}\right)^2 - \frac{g}{2}\left(h_{j-\frac{1}{2},k}\right)^2\right] + \frac{\Delta t}{\Delta x}g\bar{h}_{j,k}\left(B_{j+\frac{1}{2},k} - B_{j+\frac{1}{2},k}\right) \\
&= (\bar{hu})^n_{j,k}
\end{aligned}$$

where we have used

$$\frac{\left(h_{j+\frac{1}{2},k}\right)^2 - \left(h_{j-\frac{1}{2},k}\right)^2}{2} = -\bar{h}^n_{j,k}\left(B_{j+\frac{1}{2},k} - B_{j-\frac{1}{2},k}\right), \qquad \text{(II.47)}$$

$$\frac{\left(h_{j,k+\frac{1}{2}}\right)^2 - \left(h_{j,k-\frac{1}{2}}\right)^2}{2} = -\bar{h}^n_{j,k}\left(B_{j,k+\frac{1}{2}} - B_{j,k-\frac{1}{2}}\right). \qquad \text{(II.48)}$$

We have to verify (II.47) and (II.48). In the fully flooded cells, where $w_{j,k} > B_{j\pm\frac{1}{2},k}$, we have

$$\begin{aligned}
\frac{\left(h_{j+\frac{1}{2},k}\right)^2 - \left(h_{j-\frac{1}{2},k}\right)^2}{2} &= \frac{h_{j+\frac{1}{2},k} + h_{j-\frac{1}{2},k}}{2}\left(h_{j+\frac{1}{2},k} - h_{j-\frac{1}{2},k}\right) \qquad \text{(II.49)} \\
&= \bar{h}^n_{j,k}\left(w_{j,k} - B_{j+\frac{1}{2},k} - w_{j,k} - B_{j-\frac{1}{2},k}\right) = -\bar{h}^n_{j,k}\left(B_{j+\frac{1}{2},k} - B_{j-\frac{1}{2},k}\right)
\end{aligned}$$

and thus (II.47) is satisfied. One can easily prove the same for (II.48). It remains to verify the solution for

$$\begin{aligned}(\bar{hv})_{j,k}^{n+1} &= (\bar{hv})_{j,k}^{n} - \frac{\Delta t}{\Delta y}\left[\frac{g}{2}\left(h_{j,k+\frac{1}{2}}\right)^2 - \frac{g}{2}\left(h_{j,k-\frac{1}{2}}\right)^2\right] + \Delta t\,\bar{\mathbf{S}}_{j,k}^{(3),n} \\ &= (\bar{hv})_{j,k}^{n} - \frac{\Delta t}{\Delta y}\left[\frac{g}{2}\left(h_{j,k+\frac{1}{2}}\right)^2 - \frac{g}{2}\left(h_{j,k-\frac{1}{2}}\right)^2\right] + \frac{\Delta t}{\Delta y} g\bar{h}_{j,k}\left(B_{j,k+\frac{1}{2}} - B_{j,k+\frac{1}{2}}\right) \\ &= (\bar{hv})_{j,k}^{n}\end{aligned}$$

which can be satisfied by using (II.48). In the partially flooded cells, where $w_{j,k} < B_{j-\frac{1}{2},k}$, we have $h_{j-\frac{1}{2},k} = 0$, and thus using (II.32) and (II.47) yields

$$\frac{\left(h_{j+\frac{1}{2},k}\right)^2}{2} = \frac{\Delta x_{j,k}^* h_{j+\frac{1}{2},k}}{2\Delta x}\left(B_{j+\frac{1}{2},k} - B_{j+\frac{1}{2},k}\right) = -\frac{h_{j+\frac{1}{2},k}}{2}\Delta x_{j,k}^*(B_x)_{j,k}, \tag{II.50}$$

which is true since at the studied-steady situation, $x_{j,k}^* = x_w^*$, which implies that $\Delta x_{j,k}^* = \Delta x_w^*$, and hence, $-\Delta x_{j,k}^*(B_x)_{j,k} = h_{j+\frac{1}{2},k}$. Finally, we show the same for $w_{j,k} < B_{j,k-\frac{1}{2}}$, $h_{j,k-\frac{1}{2}} = 0$ using (II.48):

$$\frac{\left(h_{j,k+\frac{1}{2}}\right)^2}{2} = \frac{\Delta y_{j,k}^* h_{j,k+\frac{1}{2}}}{2\Delta y}\left(B_{j,k+\frac{1}{2}} - B_{j,k+\frac{1}{2}}\right) = -\frac{h_{j,k+\frac{1}{2}}}{2}\Delta y_{j,k}^*(B_y)_{j,k}, \tag{II.51}$$

which is also true since $y_{j,k}^* = y_w^*$. This implies that $\Delta y_{j,k}^* = \Delta y_w^*$, and hence, $-\Delta y_{j,k}^*(B_y)_{j,k} = h_{j,k+\frac{1}{2}}$.

This concludes the proof of the theorem. □

6 Evaluation

The new scheme proposed in this paper is denoted as HWP scheme. In order to test the scheme and compare it with an existing scheme, we implemented the HWP and KP [18] schemes for computation on graphics processing units (GPUs). We then performed a number of numerical experiments. As a first step, we analyzed a wave run-up on a slope. This simulation is followed by a verification against a two-dimensional parabolic basin benchmark for which an analytical solution exists. To demonstrate the stability and to measure the performance of the schemes, we created two real-world scenarios. The first is a breach-initiated flood in an urban area, and the second is a flood event in a rural area.

6.1 Wave Run-Up on a Slope

In this comparison, we compare the schemes on a small domain, where we simulate a wave run-up on a slope. We let the water oscillate for 1000 seconds and visualize the results (see Figure II.8). For the KP scheme, the upper part of the domain does not dry out. If a cell becomes wet, it will always contain a very thin layer of water, and will never dry out again. For the HWP scheme, the upper part of the slope dries out due to the new reconstruction and draining time step technique.

6.2 Parabolic Basin Benchmark

The analytical solution of the parabolic basin case was first introduced by Thacker [83]. It describes time-dependent oscillations of planar water surface in a parabolic basin. It serves as a good basis to

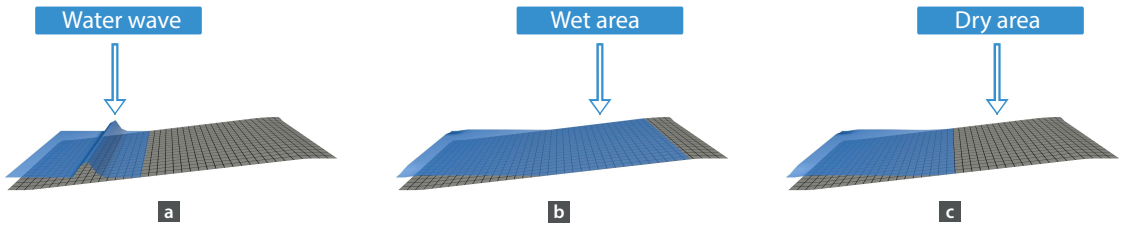


Figure II.8: Comparison of drying of the KP and the HWP schemes. We simulate a wave run-up on a slope and visualize the solution after 1000 seconds. (a) Initial condition. (b) Solution of the KP scheme, where the upper part of the simulation domain is wet. There is a thin layer of water, which is incorrect. (c) Solution of the HWP scheme, the upper part of the domain is dry.

compare different numerical schemes [14, 84, 85, 86, 87]. Recently, Sampson et al. [87] extended the solution of Thacker to support bed friction. However, their solution is limited to one dimension. In this two-dimensional case, we use the same setup as Holdahl et al. [88]. The bathymetry of the two-dimensional parabolic basin is given by:

$$B(x,y) = D_0\left(\frac{x^2+y^2}{L^2} - 1\right). \tag{II.52}$$

The water surface elevation and the velocities are given by:

$$\begin{aligned} w(x,y) &= 2AD_0\left(x\cos\omega t \pm y\sin\omega t + LB_0\right) \\ u(x,y) &= -A\cos\omega t \\ u(x,y) &= \pm A\sin\omega t \end{aligned} \qquad \omega = \sqrt{\frac{2D_0}{L^2}}, \tag{II.53}$$

where we set $D_0 = 1$, $L = 2500$, $A = \frac{L}{2}$, $B_0 = -\frac{A}{2L}$, the gravitational constant $g = 1$, the desingularization $\epsilon = 0.01$, and use 100×100 grid cells with a second-order accurate Runge-Kutta time integrator. Figure II.9 shows our simulation results for two snapshots in time, where we compare the solutions of the numerical solvers to the analytical one. We visualize a one-dimensional slice in x-dimension in the middle of the computation domain. We plot values $w_{j,k}$ at cell centers, which are the average values. Both the KP and the HWP schemes capture well the water levels of the analytical solution. However, there is an error accumulating for the velocities along the wet/dry boundaries. This can be found in many schemes [88, 17, 18], and it is difficult to avoid. We can reduce the error by setting the desingularization ϵ to a higher value. However, higher ϵ values cause spurious oscillations in the KP scheme, which were not observed in the HWP scheme. Furthermore, for small water heights, numerical errors can be introduced when calculating the velocity $u = hu/h$. This is related to a limited floating-point precision [89].

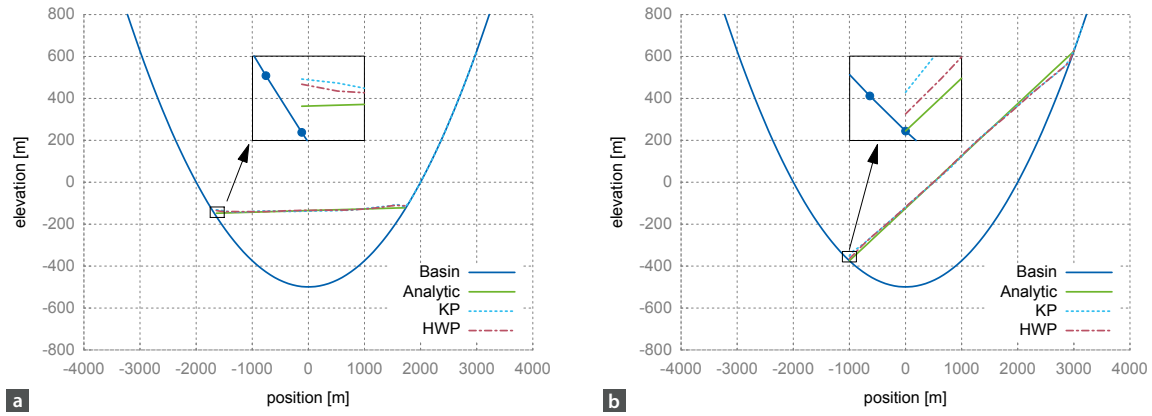


Figure II.9: Simulation of oscillating water in a parabolic basin, compared to the analytical solution after (a) 300 seconds and (b) 400 seconds. Values are plotted at the cell centers. Blue dots in the inlay windows represent positions of the cell centers.

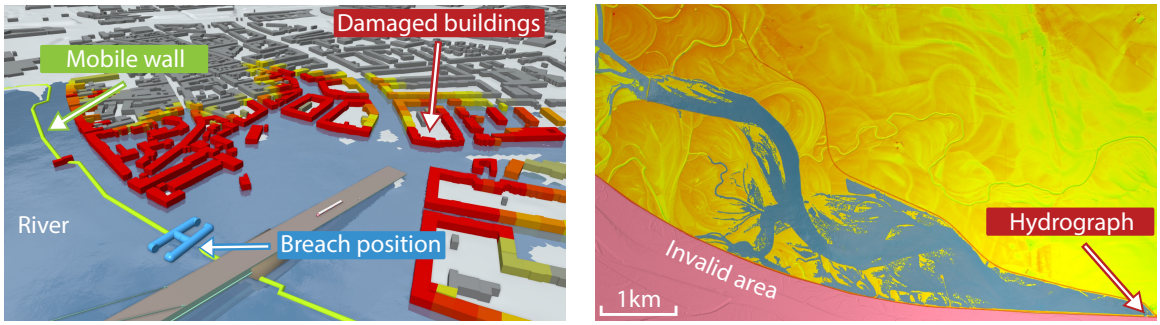


Figure II.10: Real-world case studies. (a) Simulation of a levee breach caused by a failure of the mobile protection walls in the city of Cologne, Germany. The green line along the riverside represents the mobile protection wall. We removed a short section of the wall to simulate a breach (indicated in blue). The buildings are colored according to the damage, where red denotes high damage and grey no damage. (b) Results of a 5 days long hydrograph-based flood simulation in Lobau, Austria. The bathymetry is colored according to the elevation values. Red color represents higher altitudes, green color represents lower altitudes. Wet areas are shown in blue. The hydrograph is supplied for the short section at the bottom right corner.

6.3 Real-World Performance Benchmark in Cologne

In the first real-world scenario, we simulate a levee breach in the city of Cologne, Germany. This scenario models a failure of the mobile flood protection walls, which are installed along the river to protect the city. If the mobile wall collapses or the water height becomes high enough to overtop the walls, the city gets flooded. These two cases can happen in real-world situations, and it is important to understand their impacts.

In our case study, we simulate a one hour long levee breach. We simulate the breach by opening a short section of the mobile walls and letting the water flow into the city (see Figure II.10a). The buildings are colored based on the computed damage, where grey represents no damage, yellow is the middle damage, and red is the highest damage [90].

To assess the performance of our numerical scheme, we carried out a benchmark and compared the number of executed time steps (iterations) of the two schemes. A longer time step size requires fewer iterations to simulate the same duration and thus increases the performance. Figure II.11 shows our performance results for a 60 minute long simulation run. The size of the simulation domain is 1.4×1.6 kilometers, it contains 277×329 cells, where each cell is 5×5 meters large. In this case study the KP scheme executes $\approx 4\times$ more time steps than the HWP scheme. The average time step per second is 0.08678 for the HWP scheme and 0.02253 for the KP scheme. The reason for this is that the KP scheme is not well-balanced at the dry/wet boundaries, thus high velocities appear at these locations. To keep the scheme stable, the CFL condition is applied. This means that the size of the actual time step is computed based on the highest velocity in the computation domain. Therefore, it acts as a limiting factor for the time step size. The HWP scheme does not suffer from this problem and can perform longer jumps in time while preserving numerical stability. We show both the instantaneous and cumulative number of time steps. Figure II.11a shows the average number of time steps per minute. Figure II.11b shows the total number of time steps for the 60 minutes long simulation run. During the first five minutes, the number of time steps is increasing, as the water starts to flow into the city. After five minutes, as the flow starts to stabilize, the number of time steps in the HWP scheme is stabilizing, too. In the KP scheme, the number of time steps continues to increase. Compared to the KP scheme, the HWP scheme is more stable with respect to the number of time steps.

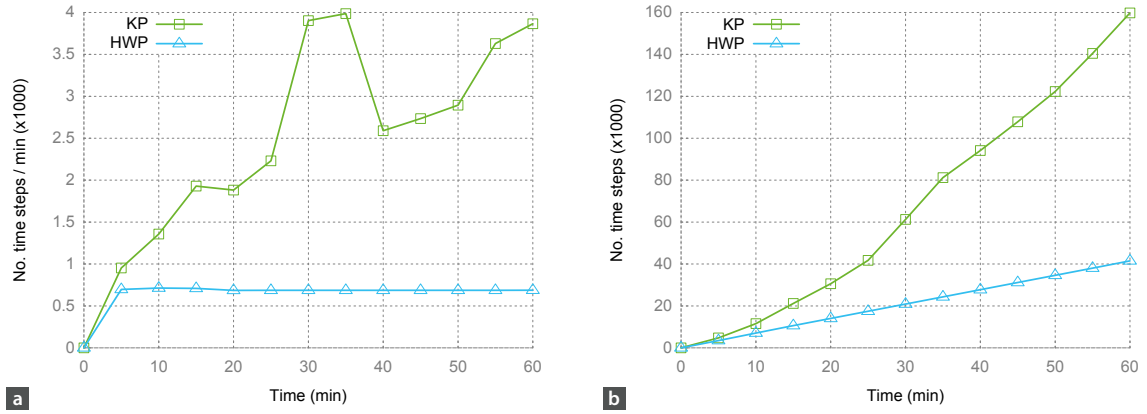


Figure II.11: Real-world performance benchmark of a breach flood in the city of Cologne, Germany. We compare the time step performance of the KP and the HWP schemes. The figure shows the number of time steps as a function of the simulated time. Lower numbers corresponds to longer time step sizes in the simulation and thus to increased performance. (a) Average number of time steps per minute. (b) Cumulative number of time steps during the simulated 60 minutes.

6.4 Real-World Performance Benchmark in Lobau

The second case study involves the Lobau area, which is the alluvial backwater and floodplain of the Danube-Auen National Park. It extends on the left bank of the river Danube from river kilometer (rkm) 1918 to rkm 1908 downstream of the city of Vienna (see Figure II.10b). If the water level in the Danube rises, water flows from the river into the floodplain, causing regular flooding events. The size of the area is 1474 ha and it consists of floodplain forests and surface water bodies.

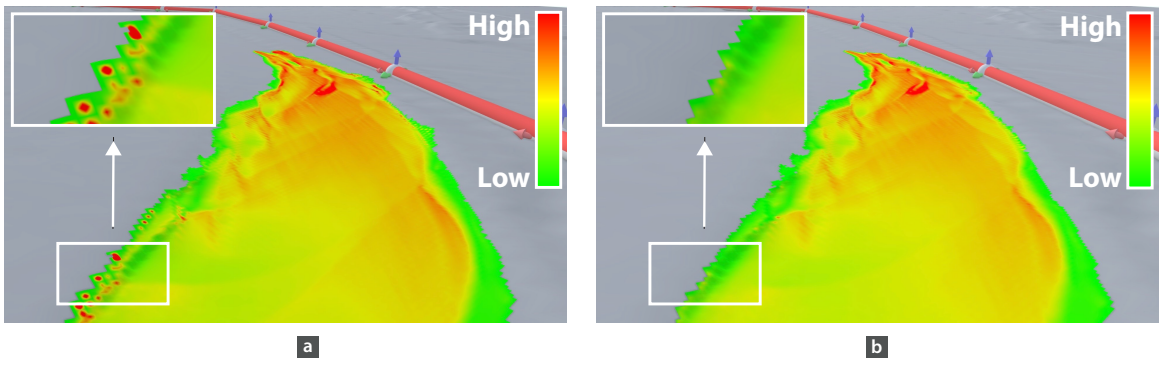


Figure II.12: Velocity profile of the real-world simulation in Lobau. The bathymetry is colored according to the velocity magnitude. (a) Simulation results of the KP scheme. High velocity spots (red) appear at the dry/wet boundaries. (b) Simulation results of the HWP scheme. No high velocity spots, the velocity profile is consistent.

In this case study, we use a hydrograph from 13.01.2011 and simulate a 12 hours long flooding. The size of the simulation domain is 7.5×5 kilometers, it contains 2508×1682 cells, where each cell is 3×3 meters large. We simulate a period on 13.01.2011 for which we prescribe the hydrograph in the bottom right corner of the domain as a boundary condition. Both water level and discharge are prescribed. Discharge values describe the inflow from the Danube into the Lobau. Even though this is a rural area and does not contain any buildings, it is challenging to perform simulations for this region, since it has a very complex bathymetry (lots of small channels and steep slopes). Here, we present only the first 12 hours of the simulation, since the KP scheme slows down dramatically after 12 hours and 50 minutes. At this point, the actual time step size is 0.000156 seconds and continues to decrease. The reason for this is that the velocities are accumulating and abnormally rising at the steep dry/wet boundaries limiting the time step size.

Figure II.12 shows the high velocity spots, which are responsible for the simulation slow down. In the HWP scheme, the velocity profile is more consistent. For this special case, the HWP scheme was $\approx 8\times$ faster compared to the KP scheme during the 12 hours long simulation. This difference gets higher as the time advances. Using the HWP scheme, we have successfully performed a 168 hours long simulation, where the solver remained stable.

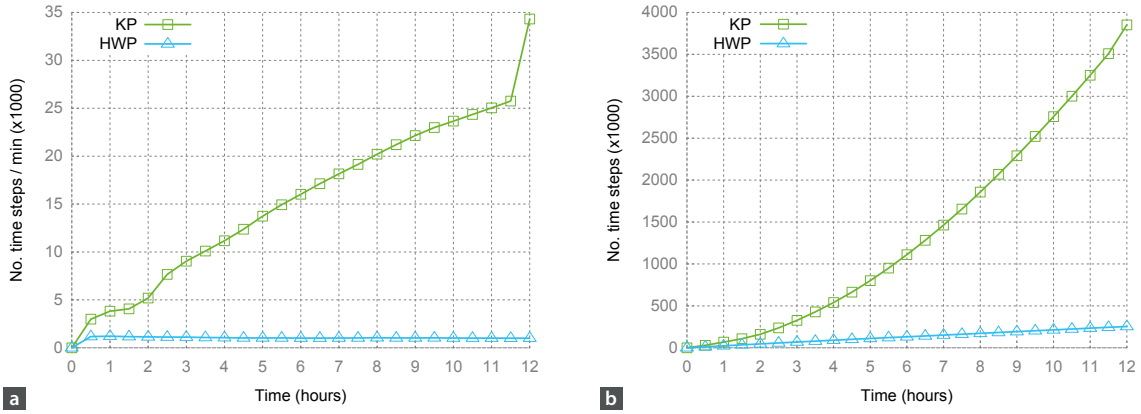


Figure II.13: Real-world performance benchmark of a flood event in the Danube-Auen National Park, in Lobau, Austria. The figure shows the number of time steps as a function of the simulated time. Lower numbers correspond to longer time steps in the simulation and thus to increased performance. (a) Average number of time steps per minute. (b) Total number of time steps during the simulated 12 hours.

Our new HWP scheme executes approximately the same number of time steps in both case studies, which is below 1000 time steps per minute. The KP scheme is not as stable regarding the number of time steps. For the first case study, the average number of time steps is between $1000 - 4000$ time steps per minute, while for the second case study, it is continuously increasing (see Figure II.13).

7 Summary

We presented a two-dimensional numerical scheme for the SV system of the shallow water equations. We used a new two-dimensional reconstruction and a special correction procedure to ensure positive water heights and well-balanced states at partially flooded cells. The positivity of the water height is guaranteed by the draining time step, which is activated in the partially flooded cells if the divergence is positive. We proved that the scheme is well-balanced and positivity preserving in the presence of partially flooded cells. Furthermore, it preserves "lake-at-rest" and "dry lake" steady states, as well as their combinations. The scheme was verified against the analytical solution of the parabolic basin benchmark. We measured and compared the time step performance of the KP and HWP schemes for two real-world scenarios. The first scenario was a flood event caused by a breach in the flood protection walls of the city of Cologne. The second scenario considered a flooding in the national park of Lobau near Vienna, where we used a hydrograph as a boundary condition to simulate the incoming flood from the Danube river. The new scheme proved its stability and it also proved to be faster than the KP scheme, as it needs fewer time steps to simulate the same time span due to the removal of spurious high velocities at the wet/dry boundaries.

Acknowledgements

This work was supported by grants from the Austrian Science Fund (FWF) project number W1219-N22 (Vienna Doctoral Programme on Water Resource Systems) and from the Vienna Science and Technology Fund (WWTF) project number ICT12-009 (Scenario Pool), and from the European Research Council (ERC) Advanced Grant "FloodChange", project number 291152. We thank the flood protection centre of Steb Cologne, AöR and the ViaDonau, Vienna for supplying the terrain data.

Chapter III

Kepler Shuffle for Real-World Flood Simulations on GPUs

Zsolt Horváth, Rui A. P. Perdigão, Jürgen Waser, Daniel Cornel, Artem Konev, Günter Blöschl

This chapter is based on a paper which is published in
International Journal of High Performance Computing, 30(4), pages 379-395, 2016.

Abstract

We present a new GPU implementation of two second-order numerical schemes of the shallow water equations on Cartesian grids. Previous implementations are not fast enough to evaluate multiple scenarios for a robust, uncertainty-aware decision support. To tackle this, we exploit the capabilities of the NVIDIA Kepler architecture. We implement a scheme developed by Kurganov and Petrova (KP), and a newer, improved version by Horváth et al. (HWP). The KP scheme is simpler but suffers from incorrect high velocities along the wet/dry boundaries, resulting in small time steps and long simulation runtimes. The HWP scheme resolves this problem but comprises a more complex algorithm. Previous work has shown that HWP has the potential to outperform KP, but no practical implementation has been provided. The novel shuffle-based implementation of HWP presented here takes advantage of its accuracy and performance capabilities for real-world usage. The correctness and performance is validated on real-world scenarios.

1 Introduction

Flood risk assessment is of key importance in minimizing damages and economical losses caused by flood events. These are challenging to predict due to nonlinear interactions between driving processes [50]. The first step in flood risk studies is the identification of flood prone areas [9]. This requires the implementation of hydrodynamic models that enable one to quantify the evolution of a flood and its hydraulic representative variables, e.g., water level and velocity.

The shallow water equations (SWE) are capable of modeling many flood phenomena such as levee breaches, flood plain inundations, tsunamis, dam breaks, or river flows in both rural and urban areas. Our primary interest is in decision making systems, where we evaluate many scenarios and select the solution with the best outcome [46]. Modern approaches allow the user to evaluate the uncertainties of the parameters considered in the simulation, requiring the simulation of many different scenarios, which is usually computationally expensive. Therefore, simulation runs have to be as fast as possible to reduce the overall time of finding the best solution.

The equations are

The SWE are a set of hyperbolic partial differential equations, described by the Saint–Venant (SV) system, where the fluid motion is introduced by the gravitational acceleration. They are derived from depth-integrating the Navier-Stokes and the continuity equations, and represent wave propagation with a horizontal length scale much greater than the vertical length scale.

In this paper, we focus on schemes which are defined on Cartesian grids and present a new implementation of two shallow water schemes which are both able to handle “dry lake” and “lake at rest” steady state solutions. We discuss how to exploit the capabilities of modern graphics processing units (GPUs) using the CUDA programming model. We focus on the NVIDIA Kepler architecture and its newest features to achieve peak performance. We build upon the scheme of Kurganov and Petrova [18] (KP), and Horváth et al. [1] (HWP), which improves the former by avoiding spurious high velocities at the dry/wet boundaries. The HWP scheme is more complex, and hence it requires more GPU resources. Nevertheless, previous work has shown that the HWP scheme has the potential to outperform KP, since it can use longer time step sizes [1]. Until now, no implementation has been given to exploit this potential in practice. To accomplish this, we present a new implementation based on recent advancements in the GPU technology. In summary, this paper contributes the following:

- a new implementation of the HWP scheme which outperforms the less accurate KP scheme,
- exploitation of the Kepler shuffle instructions, which allows for an increased GPU occupancy

and speed-up by requiring less shared memory and less explicit block synchronizations,

- extensive performance benchmarks of the implemented schemes and validation against real-world scenarios.

2 Related Work

In the numerical treatment of the SWE, the spatial domain is discretized. For this purpose, one can use an unstructured mesh or a structured mesh (Cartesian). Unstructured meshes have demonstrated good properties for the simulation of rainfall/runoff events [91]. Since cells do not establish preferential directions, this type reduces the mesh influence on the numerical results. Also, it is easy to adapt unstructured cells to capture local structures. The disadvantage of using unstructured meshes on the GPU is related to the mesh disorder, which makes it expensive to load and share data between threads [92]. Structured meshes have proved to be suitable for GPU computations [39]. Indeed, the GPUs have been optimized for memory access when Cartesian grids are used. The main advantage of structured meshes is the inherent order existent in their creation. On Cartesian grids, every cell knows its neighbors implicitly, no information on the topography is needed to maintain data dependencies. To overcome the limitation regarding capturing local structures, adaptive mesh refinement (AMR) [42] or the cut-cell approach [93] can be used.

Explicit numerical schemes are well suited for parallel execution on the GPU to solve hyperbolic partial differential equations [94, 58, 95, 96]. Hagen et al. [51] were among the first to deliver a GPU solver for the shallow water equations based on the first-order Lax-Friedrichs and the second-order central-upwind schemes. They achieve a 15-30 times speedup compared to an equivalently tuned central processing unit (CPU) version. Lastra et al. [60] implement a first-order well-balanced finite volume solver for one-layer SWE using OpenGL and Cg languages. Acuña and Aoki [48] propose a multi-GPU solution for tsunami simulations. They use overlapping kernels to compute the solution for x and y directions concurrently. Liang et al. [97] add a friction slope to the conservation of momentum to simulate tsunami inundation by using the MacCormack method. De la Asunción et al. [53, 59] demonstrate solvers for one-dimensional and two-dimensional SWE using the CUDA Toolkit. They show that an optimized CUDA solver is faster than a GPU version which is based on a graphics-specific language. Brodtkorb et al. [38] implement three second-order schemes on the GPU. They discuss how single- and double-precision arithmetics affect accuracy, efficiency, scalability, and resource utilization. They show that double precision is not necessary for the second-order SWE [38]. Their implementation demonstrates that all three schemes map very well to the GPU hardware. Simulating real-world dam-break scenarios, de la Asunción et al. [84] and Brodtkorb et al. [39] report relevant computational speed-ups in comparison to sequential codes. Moreover, Sætra and Brodtkorb use a multiple GPU solver to tackle large-domain simulations and reduce both the memory footprint and the computational burden by using sparse computation and sparse memory approaches. Sætra et al. [42] are able to increase the grid resolution locally for capturing complicated structures or steep gradients in the solution. They employ adaptive mesh refinement which recursively refines the grid in parts of the domain. Vacondio et al. [98] implement an implicit local ghost cell approach, that enables the discretization of a broad spectrum of boundary conditions. They also present an efficient block deactivation procedure in order to increase the efficiency of the numerical scheme in the presence of wetting-drying fronts.

3 Numerical Schemes

In this section we summarize the underlying numerical theory of the implemented schemes. The hyperbolic conservation law described by the two-dimensional shallow water equations of the SV system can be written as:

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix} + \begin{bmatrix} 0 \\ -gu\sqrt{u^2+v^2}/C^2 \\ -gv\sqrt{u^2+v^2}/C^2 \end{bmatrix}, \tag{III.1}$$

where h represents the water height, hu is the discharge along the x-axis, hv is the discharge along the y-axis (Figure III.1a), u and v are the average flow velocities, g is the gravitational constant, B is the bathymetry, and C is the Chézy friction coefficient. We use the relation $C = \frac{1}{n}h^{1/6}$, where n is Manning's roughness coefficient. Subscripts represent partial derivatives, i.e., $\mathbf{U}_t$ stands for $\frac{\partial \mathbf{U}}{\partial t}$.

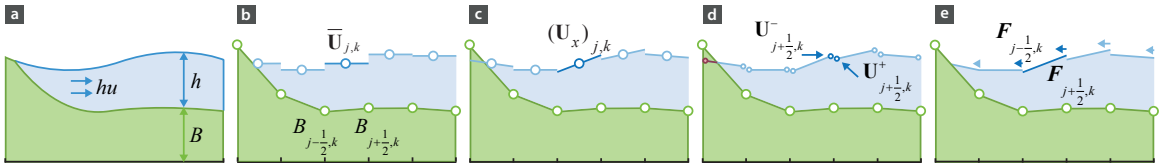


Figure III.1: Schematic view of a shallow water flow, definition of the variables, and flux computation. a) Continuous variables. b) The conserved variables $\mathbf{U}$ are discretized as cell averages $\bar{\mathbf{U}}_{j,k}$. The bathymetry function B is approximated at cell interface midpoints. c) Slopes of the water surface $(\mathbf{U}_x)_{j,k}$ are reconstructed using the minmod flux limiter. d) Left- and right-sided point values are computed at cell interface midpoints. The red circle indicates that a negative water height is computed. Since water heights cannot be negative, they are corrected before the flux computation. e) Fluxes are computed using the central-upwind flux function at the cell center interfaces.

In vector form the system can be written as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U}, B)_x + \mathbf{G}(\mathbf{U}, B)_y = \mathbf{S}_B(\mathbf{U}, B) + \mathbf{S}_f(\mathbf{U}), \tag{III.2}$$

where $\mathbf{U} = [h, hu, hv]$ is the vector of conserved variables, $\mathbf{F}$ and $\mathbf{G}$ are flux functions, $\mathbf{S}_B$ and $\mathbf{S}_f$ represent the bed slope and bed friction source terms respectively.

Herein, we focus on two numerical schemes, the first one developed by Kurganov and Petrova [18] and its improved version developed by Horváth et al. [1]. A good numerical scheme should be able to exactly preserve both "lake at rest" and "dry lake" steady states as well as their combinations. The methods that exactly preserve these solutions are termed "well-balanced" [99, 100, 101, 16, 61, 62, 63].

3.1 Kurganov-Petrova Scheme (KP)

The KP scheme is based on a two-dimensional central-upwind second-order numerical scheme developed by Kurganov and Petrova [18]. It allows for a discontinuous bathymetry and is more suitable for GPU implementation than its predecessor [17]. To avoid negative water heights h and preserve well-balancedness, the KP scheme changes from the variables $[h, hu, hv]$ to $[w, hu, hv]$, where $w = h + B$ represents the water surface (Figure III.1a). Kurganov and Levy use a non-oscillatory conservative piecewise linear reconstruction of w, which is properly corrected near dry areas, without switching to a reconstruction of h there. The correction relies on the fact that

the original bottom function B is replaced with its continuous piecewise linear approximation. However, near dry areas and at the dry-wet boundaries the scheme can still create large errors in the flux calculations, since the water height can become very small or even zero. Due to $u = \frac{hu}{h}$ and $v = \frac{hv}{h}$, these computations may lead to large errors in the partially flooded cells for small water heights and they have a singularity at zero water height ($h = 0$). To deal with this problem, Kurganov and Levy use the desingularization:

$$u = \frac{\sqrt{2}h\,(hu)}{\sqrt{h^4 + \max\left(h^4, \epsilon\right)}} \quad v = \frac{\sqrt{2}h\,(hv)}{\sqrt{h^4 + \max\left(h^4, \epsilon\right)}}, \tag{III.3}$$

where ϵ is a small apriori chosen positive number. This has a dampening effect on the velocities as the water height approaches zero. Determining a proper value for ϵ is a difficult task. High values lead to large errors in the simulation results, while low values give small time steps. There is one more problem related to the partially flooded cells. A correction to the reconstruction has to be applied. This correction solves the positivity problem and guarantees that all water heights are non-negative. However, at the partially flooded cells, it can lead to large errors for small water heights and the flow velocity will grow smoothly in these formerly dry areas, since the correction is not well-balanced there. High velocities in the domain cause small time steps, thus poor performance of the scheme and slower simulation. Another issue related to this correction is that the water climbs up on the shores at the dry/wet boundaries. Finally, if a cell becomes wet, it will almost never be completely dry again.

3.2 Horváth-Waser-Perdigão Scheme (HWP)

The HWP scheme is an improved version of the KP scheme. It introduces a new reconstruction and the draining time step (DTS) technique to restore the well-balancedness for the partially flooded cells. This scheme is well-balanced, positivity preserving, and handles dry states. The latter is ensured by using the DTS technique in the time integration process, which guarantees non-negative water depths. Unlike the KP scheme, the new technique does not generate high velocities at the dry/wet boundaries, which are responsible for small time step sizes and slow simulation runs. The new scheme preserves ”lake at rest“ steady states and guarantees the positivity of the computed fluid depth in the partially flooded cells. Due to the new reconstruction procedure, this scheme has some additional computations compared to the KP, and thus it is computationally more expensive. It detects partially flooded cells and computes a new waterline for them. This information is used to reconstruct proper point values at the cell interfaces and to ensure well-balancedness. It reduces the non-physical high velocities at the dry-wet boundaries and allows for longer time steps which results in shorter simulation runtimes.

3.3 Spatial Discretization

The same spatial discretization is applied to both schemes on a uniform grid (Figure III.2), where the conserved variables $\mathbf{U}$ are defined as cell averages (Figure III.1b). The bathymetry is given as a piecewise bilinear surface defined by the values at the cell vertices. The fluxes are computed at the integration points, i.e., at the midpoints of the cell interfaces. The central-upwind semi-discretization of (III.1) can be written down as the following system of time-dependent, ordinary differential equations (ODE):

$$\begin{aligned}\frac{d}{dt}\bar{\mathbf{U}}_{j,k} = &- \left[\mathbf{F}\left(\mathbf{U}_{j+\frac{1}{2},k}\right) - \mathbf{F}\left(\mathbf{U}_{j-\frac{1}{2},k}\right)\right] - \left[\mathbf{G}\left(\mathbf{U}_{j,k+\frac{1}{2}}\right) - \mathbf{G}\left(\mathbf{U}_{j,k-\frac{1}{2}}\right)\right] \\ &+ \mathbf{S}_B\left(\bar{\mathbf{U}}_{j,k}, B_{j,k}\right) + \mathbf{S}_f\left(\bar{\mathbf{U}}_{j,k}\right) \\ = &\,\mathbf{R}_{F+G}\left(\bar{\mathbf{U}}\right)_{j,k} + \mathbf{S}_{B+f}\left(\bar{\mathbf{U}}_{j,k}, B_{j,k}\right)\end{aligned} \tag{III.4}$$

where $\mathbf{U}_{j\pm\frac{1}{2},k}$ and $\mathbf{U}_{j,k\pm\frac{1}{2}}$ are the reconstructed point values at the cell interface midpoints.

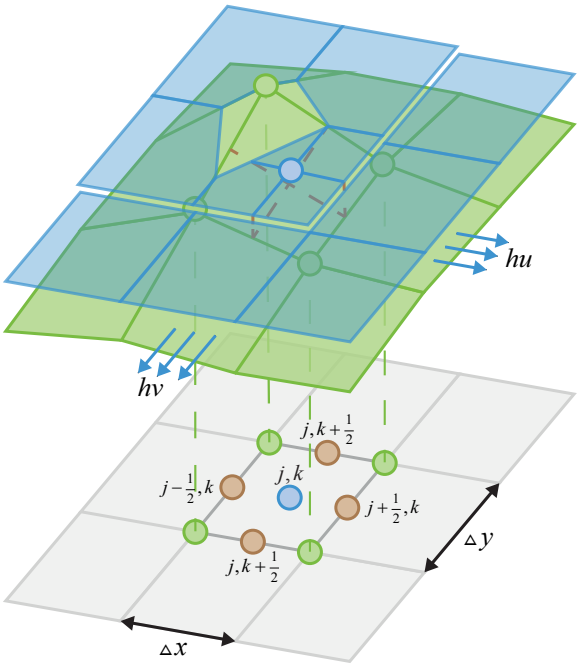


Figure III.2: Two-dimensional grid-based representation of the discretized variables of the shallow water equations. Cell averages $\bar{\mathbf{U}}_{j,k}$ are defined at cell centers (blue dots). Green dots indicate the sampling points of the bathymetry function B. Brown dots indicate the approximated values of the bathymetry function at the cell interface midpoints [1].

In order to compute the fluxes $\mathbf{F}$ and $\mathbf{G}$ across the cell interfaces, we start with reconstructing the water surface. A planar surface is computed for each cell using the cell averages $\bar{\mathbf{U}}$ and a piecewise bilinear approximation, which determines the slope of the water in the cells (Figure III.1c). This slope reconstruction is performed using the generalized minmod flux limiter:

$$(\mathbf{U}_d)_{j,k} = \text{minmod}\left(\theta b, c, \theta f\right), \tag{III.5}$$

where $(\mathbf{U}_d)_{j,k}$ are the derivatives of the conserved variables in x or y direction, b, c, f are the backward, central, and forward difference approximations of the derivative, that is the slope of the water surface within cell $C_{j,k}$. The parameter $\theta \in [1,2]$ is used to control the numerical viscosity of the scheme. Here we use $\theta = 1.3$ as suggested by Kurganov and Levy [17]. The index symbol d indicates the direction of the derivation, which can be x or y in our case. The minmod flux limiter is defined as:

$$\text{minmod}(z_1,z_2,z_3) := \begin{cases} \min_j\{z_j\}, & \text{if } z_j > 0 \quad \forall j, \\ \max_j\{z_j\}, & \text{if } z_j < 0 \quad \forall j, \\ 0, & \text{otherwise,} \end{cases} \tag{III.6}$$

and is applied in a componentwise manner to all three elements $[\bar{w}, \bar{hu}, \bar{hv}]$ of the vector $\bar{\mathbf{U}}$.

In general, the slope reconstruction produces negative water heights h at the integration points in the partially flooded cells. If the water height becomes negative, the computation breaks down since the eigenvalues of the system are $u \pm \sqrt{gh}$ and $v \pm \sqrt{gh}$. Hence, these values have to be corrected while maintaining mass conservation. The KP correction solves the positivity problem by raising and lowering the water level at the left and right side of the cell according to the bathymetry function. This guarantees that all water heights are non-negative. However, this violates the well-balancedness of the scheme at the dry/wet boundary causing high velocities to appear. The HWP scheme uses a more complex correction, which preserves the well-balancedness and reduces the spurious velocities. The slope reconstruction is followed by the point value

computation. For every cell interface at the integration points, point values are reconstructed (Figure III.1d). Each cell interface has two point values, one from the cell on the left and one from the right. Based on these point values, we compute the fluxes using the central-upwind flux function (Figure III.1e). In order to do so, we need to calculate the local speed values $u = hu/h$ and $v = hv/h$ at the integration points. This leads to large errors as the water height h approaches zero and can produce high velocities and instabilities. Since the time step size is calculated using the maximum velocity component in the domain (III.8), it is also affected by this problem. To reduce the effect of these high velocities, we have to apply the desingularization (III.3). Finally, we compute the bed slope source term S_B and the friction source term S_f.

3.4 Temporal Discretization

We have discussed the spatial reconstruction of the point values and the flux computation. In the following, we continue with the time-quadrature for the fluxes applied to both schemes. We discretize the semi-discrete scheme (III.1) in time, and advance by Δt using a standard second-order accurate total variation diminishing (TVD) Runge-Kutta scheme [102, 103]:

$$\begin{aligned}
\bar{\mathbf{U}}^{*}_{j,k} &= \bar{\mathbf{U}}^{n}_{j,k} + \Delta t \left(\mathbf{R}_{F+G} \left(\bar{\mathbf{U}}^{n} \right)_{j,k} + \mathbf{S}_{B+f} \left(\bar{\mathbf{U}}^{n}_{j,k}, B_{j,k} \right) \right), \\
\bar{\mathbf{U}}^{n+1}_{j,k} &= \frac{1}{2} \bar{\mathbf{U}}^{n}_{j,k} + \frac{1}{2} \left[\bar{\mathbf{U}}^{*}_{j,k} + \Delta t \left(\mathbf{R}_{F+G} \left(\bar{\mathbf{U}}^{*} \right)_{j,k} + \mathbf{S}_{B+f} \left(\bar{\mathbf{U}}^{*}_{j,k}, B_{j,k} \right) \right) \right],
\end{aligned} \tag{III.7}$$

where Δt is the time step. In order to keep the numerical integration stable, the time step size is limited by the Courant-Friedrichs-Lewy (CFL) condition [18, 65, 66]:

$$\Delta t \leq \frac{1}{4} \min \left(\frac{\Delta x}{\max_{\Omega} \left(u \pm \sqrt{gh} \right)}, \frac{\Delta y}{\max_{\Omega} \left(v \pm \sqrt{gh} \right)} \right), \tag{III.8}$$

where Ω represents the whole simulation domain.

Replacing the second-order Runge-Kutta scheme with the first-order Euler scheme, the time integration (III.6) reduces to:

$$\bar{\mathbf{U}}^{n+1}_{j,k} = \bar{\mathbf{U}}^{n}_{j,k} + \Delta t \left[\mathbf{R}_{F+G} \left(\bar{\mathbf{U}}^{n} \right)_{j,k} + \mathbf{S}_{B+f} \left(\bar{\mathbf{U}}^{n}_{j,k}, B_{j,k} \right) \right] \tag{III.9}$$

We note that the HWP scheme applies a different time step computation for the partially flooded cells. It is based on the flux divergence of these cells. For more details on the draining time step computation we refer the reader to the related work [1].

4 Graphics Processing Units and the CUDA Platform

In this section, we introduce the concepts of the CUDA programming model which are most relevant for our implementation of the aforementioned schemes. We discuss performance and memory considerations from the perspective of our flux computation kernel, which is by far the most time-consuming and resource-intensive computational step in our algorithm.

On a GPU, parallel tasks are called threads. These are scheduled and executed simultaneously in groups referred to as warps. One warp contains 32 threads, which is the number of threads effectively processed in parallel by one CUDA streaming multiprocessor (SM). GPUs have many SMs, e.g., a GeForce Titan has 14 SMs running in parallel to increase the effective parallelism. Furthermore, threads are organized into larger structures called blocks, and blocks are organized into grids [37].

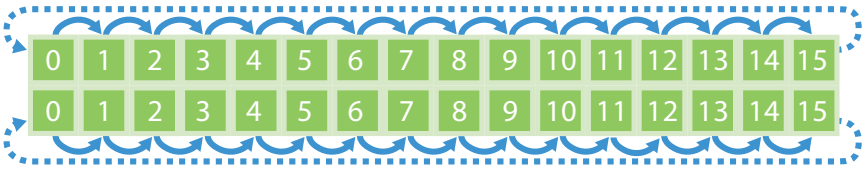


Figure III.3: Illustration of the Kepler shuffle up instruction of width 16, used in the flux kernel of block size 16×16. The shuffle up instruction is applied to a single warp executed on two rows of a block. We use it to shift data from cell $C_{j,k}$ to the next cell $C_{j+1,k}$.

4.1 Memory Usage

Understanding the CUDA memory model is often the key to achieving high performance on an NVIDIA GPU. The model consists of different regions with varying scopes, latencies and bandwidths. All threads have access to the same global memory. Shared memory is visible to all threads within a block with the same life time as the block. Each thread has its private registers and local memory.

Global memory resides in the device memory, which is the slowest, but the largest one. It is allocated and freed by the programmer. The device memory is accessed via 32, 64, or 128 byte memory transactions. These memory transactions must be aligned. Only the 32, 64 or 128 byte segments of device memory, all of them of the same size, can be read or written by memory transactions. In our implementation, we use a structure of arrays (SoA) to store elements of a vector, instead of using an array of structures (AoS), since performance is significantly affected by memory access patterns [37, 36, 45]. Therefore, when using SoA, all threads in a warp access the same relative address when loading and storing the same element of a vector.

Local memory is often mentioned as virtual memory. It is allocated by the compiler, e.g., if one uses large structures or an array that would consume too much register space. It resides in the global memory, and it is cached in L1 and L2 caches. If done right, one can use local memory to avoid costly memory transfers, but if the caches are overused and data are often evicted, the memory traffic and instruction count will be increased with a negative effect on the performance. Therefore, it is always necessary to profile the application. The best practice is to avoid the usage of the local memory whenever possible.

In this work, we focus on the on-chip memory, that is, registers and shared memory. *Shared memory* has much higher bandwidth and much lower latency than the local or the global memory. To achieve high bandwidth, it is divided into equally-sized memory modules, called banks, which can be accessed simultaneously by the threads. On the Kepler architecture, the shared memory has 48KB and it is organized into 32 banks. However, if two threads in a warp access different memory from the same bank, a bank conflict occurs, and the access is serialized. To get maximum performance we have to minimize these bank conflicts. In our flux kernel, we store temporal variables in shared memory using 2D-arrays. Bank conflicts can happen when accessing elements in the y direction, but not in the x direction. To circumvent this, we allocate arrays of $blocksize * (blocksize + 1)$. By adding a padding element of 1, we avoid the bank conflicts.

With the introduction of the *Kepler shuffle* instructions, programmers have been provided with a new memory feature to increase performance. As a faster alternative to shared memory, the Kepler shuffle can be used to efficiently share values between threads in a warp-synchronous manner. On earlier hardware, this could only be done by using shared memory. This involved writing data to the shared memory, synchronizing threads, and then reading the data back from the shared memory. The Kepler shuffle enables a thread to directly read a register from another thread in the same warp. This allows threads in a warp to collectively exchange or broadcast data. In our case, every warp is split into two rows of 16 grid cells (Figure III.3). The shuffle up instruction sends a value of a cell to the next cell on the right. Since the right most cell does not

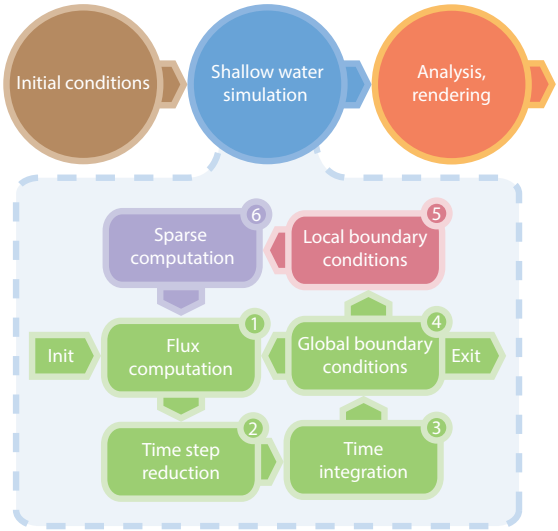


Figure III.4: Data flow of the simulation system and steps of the simulation loop. Green boxes ①-④ are the main steps of the simulation. Boxes ⑤-⑥ are optional. Box ⑤ is activated if there is a hydrograph attached. Box ⑥ is an optional optimization that skips dry cells.

have any cell on the right, it will send its values to the first one, indicated by the dashed lines in Figure III.3. By applying the shuffle down instruction, we can shift data in the opposite direction.

4.2 Block Size and Occupancy

Choosing a good block size is essential for achieving high performance. The number of threads in the block should be a multiple of the warp size, which is currently 32. Blocks should contain at least 192 threads to hide arithmetic latency [37]. One of the keys to good performance is to keep the multiprocessors on the device as busy as possible. The occupancy value is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Higher occupancy does not always imply higher performance. Above a certain level, additional occupancy does not improve performance. Factors that determine occupancy are the number of registers used per thread and the amount of shared memory used per block. Excess consumption of these resources limits the number of blocks and warps per multiprocessor. Low occupancy always interferes with the ability to hide memory latency, resulting in performance degradation. In our flux kernel, we use 16×16 threads per block. On devices with compute capability 3.5, each multiprocessor is able to schedule 2048 threads simultaneously, i.e., 8 blocks per multiprocessor in our case. Furthermore, we require 32 registers per thread and, consequently, 8096 registers per block. Since devices with compute capability 3.5 have 65536 registers per multiprocessor, 8 blocks could be scheduled in a single multiprocessor. Regarding shared memory, when using the shuffle instructions the flux kernel requires 4 temporary floating-point variables, which leads to a total of $16 \times 17 \times 4 \times 4 = 4352$ bytes of shared memory per block (the factor 17 stems from the padding of 1 to avoid bank conflicts). This means, we can achieve nearly 100% occupancy as we are limited neither by shared memory nor by register usage. However, if we do not use the shuffle instructions, we need 18 temporary floating-point variables to store in the shared memory, which results in $16 \times 17 \times 18 \times 4 = 19584$ bytes, and the occupancy drops to 25%.

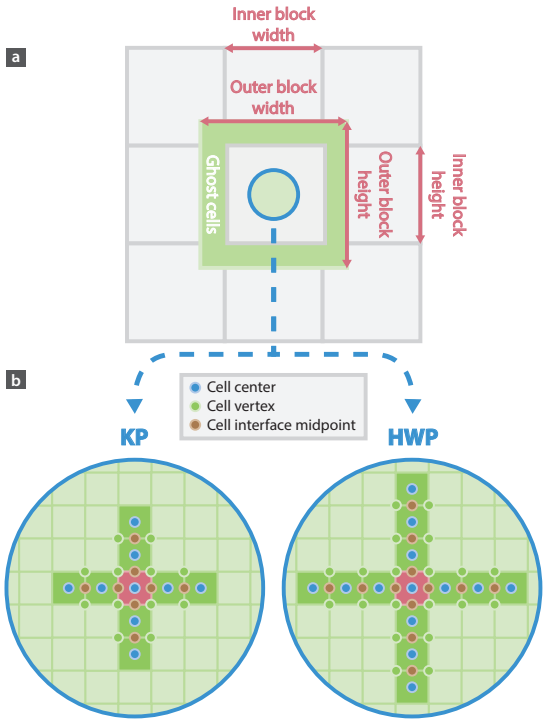


Figure III.5: Domain partitioning and the computational stencil of the flux kernel. a) Domain decomposition into blocks processed independently on the GPU. Fluxes are computed for inner block cells only. The number of ghost cells differs for the KP and HWP schemes. b) Computation stencils for the pink cells. KP requires two cells in each direction, HWP needs three. Blue dots are variables at the cell center, i.e., $\bar{\mathbf{U}}_{j,k}$ and $B_{j,k}$, brown dots are values at the cell interface midpoints $\mathbf{U}_{j\pm\frac{1}{2},k}$ and $\mathbf{U}_{j,k\pm\frac{1}{2}}$, and the bathymetry values $B_{j\pm\frac{1}{2},k\pm\frac{1}{2}}$ are defined at the green dots.

5 Implementation

We implemented a shallow water simulator as a plugin node for the Visdom framework [47]. Visdom is a software framework that integrates simulation, visualization and analysis to assist decision making. The simulator was implemented with C++ and the NVIDIA CUDA Toolkit 6.5. The framework comprises multiple modules responsible for processing various tasks. Each module can be a node of a generic data-flow network [104]. Nodes can directly import data or access the results produced by other nodes. In our case, the bathymetry node loads and samples the terrain for the shallow water node. The shallow water node simulates the fluid flow, and the simulation results are visualized using the OpenGL-based view node.

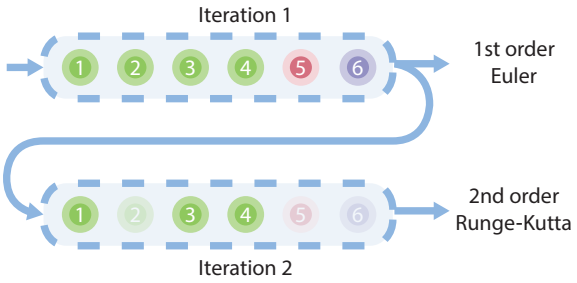


Figure III.6: Simulation steps for the first and second-order time integrations. ① Flux computation, ② Time step reduction, ③ Time integration, ④ Global boundary conditions, ⑤ Local boundary conditions, ⑥ Sparse computation. In the first iteration, all computations are active, in the second, some steps are skipped (desatured circles).

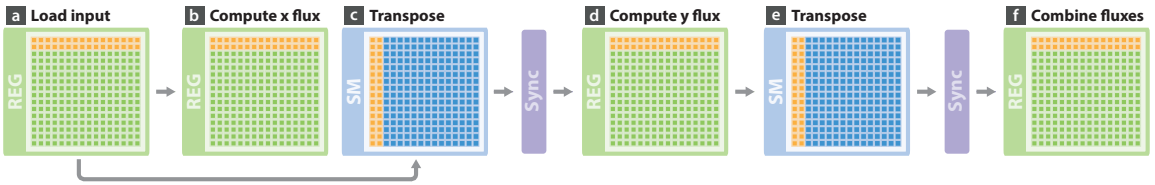


Figure III.7: Overview of the flux kernel code from the perspective of shared memory (SM) and register (REG) spaces of the block. Orange squares show the memory occupation for the current warp. Shuffle instructions operate on registers only in x-dimension. To exploit them for the computation of the fluxes in both directions, we require data transpositions via the shared memory. Explicit thread synchronization barriers are shown in purple.

Our simulation node comprises several classes and uses C++ templates on both CPU and GPU sides to achieve the highest runtime performance. Being supplied with all the inputs, the node computes the required amount of memory needed for the simulation and allocates the buffers. The simulation parameters such as the domain and cell size are uploaded to the constant memory of the GPU, where all kernels can access them. Our simulator supports dynamic modification of the input parameters, e.g., to simulate the effects of barriers that can be placed into the scene. For the shallow water simulation, we need 11 buffers of the size of the simulation domain, 2 to hold the bathymetry values, 3 + 3 to hold the values of $\bar{\mathbf{U}}_{j,k}$ and $\bar{\mathbf{U}}^*_{j,k}$, and 3 to hold the flux values combined with the source terms. We need one more floating-point value per block to hold the maximum speed value of a block. In addition, smaller buffers are used to provide other features, but their sizes are insignificant compared to the domain size. The shallow water node supports hydrograph-based simulations. A hydrograph is a function describing the development of local water level and discharges over time, often used for specifying time-dependent boundary conditions. The hydrographs are stored as indices of the affected cells, along with time-varying data of the water elevations and velocities.

5.1 Domain Partitioning

We coarsely decompose the problem into sub-problems that can be solved independently in parallel by blocks of threads. Each sub-problem is then split into finer pieces that can be solved cooperatively in parallel by all threads within a block. In case of our flux kernel, each block contains 16×16 threads or cells. The flux kernel loads data for all cells, but produces fluxes only for the inner cells (inner block), as shown in Figure III.5. The inner block size is defined by the computational stencil which is the number of neighboring cells needed to compute the fluxes at a particular cell. The computation of the flux for the inner block requires data from more cells than it contains. Additional cells from adjacent inner blocks, called ghost or halo cells, are needed. Therefore, the outer blocks must be bigger and overlap each other. The stencil differs for the KP and HWP schemes. In case of KP, the flux kernel outputs fluxes for the inner 12×12 cells. When using the HWP scheme, the inner block is smaller and contains 10×10 cells, since it requires one more ghost cell in every direction.

5.2 Simulation Steps

Figure III.4 shows the main computation steps of our simulation process. The computation starts by calculating the fluxes ① for all cells in the domain. The time step ② is computed next. The maximum speed value in the domain is used in order to compute the time step size according to (III.8). After computing the fluxes and the maximum time step size, the time integration ③ follows. In addition, each block containing wet cells is marked as active. The last of the four main simulation steps is the application of the boundary conditions ④. At this point, the process can

```
// Allocate shared memory for temporary variables
// A padding of 1 avoids bank conflicts
__shared__ float smWaterLevels [blockSize][blockSize + 1];
__shared__ float smBathY [blockSize][blockSize + 1];
…

// (a) Load input. Store conserved variables and bathymetry in variables (registers)
float waterLevel = waterLevels[globalIdx];                          ← w_{j,k}
…
float bathRightInterfaceY = bathymetriesY [globalIdx];              ← B_{j,k+1/2}
float bathRightInterfaceX = bathymetriesX [globalIdx];              ← B_{j+1/2,k}

// (b) Compute x flux. No need for prior thread synchronization because we only
//     require data loaded within the same warp
float3 fluxX = calcFlux (waterLevel, dischargeX, dischargeY, bathRightInterfaceX );

// (c) Transpose. Store conserved variables and bathymetry (in y-direction only)
//     inside shared memory in a transposed way
smWaterLevels [threadIdx.x][threadIdx.y] = waterLevel;
…
smBathY [threadIdx.x][threadIdx.y] = bathRightInterfaceY;

// Make sure that all transposed data is available from other threads
__syncthreads();

// Load data from shared memory
waterLevel = smWaterLevels [threadIdx.y][threadIdx.x];
…
bathRightInterfaceY = smBathY [threadIdx.y][threadIdx.x];

// (d) Compute y flux. The previous transposition allows us to use the same
//     shuffle-based technique
float3 fluxY = calcFlux (waterLevel, dischargeX, dischargeY, bathRightInterfaceY);

// (e) Transpose. Store fluxes in y-direction at the correct cells in shared memory
//                (we re-use the previous buffers which are no longer needed)
smWaterLevels [threadIdx.x][threadIdx.y] = fluxY.x;
…

// Synchronize after transposing back to shared memory
__syncthreads();

// (f) Combine fluxes. Write the results into global memory
```

Figure III.8: Overview of the flux kernel code.

exit the simulation loop if the desired simulation time is reached. Otherwise, the computation proceeds. If no hydrograph is supplied to the simulation node and the sparse computation is disabled, the simulation skips the steps ⑤-⑥ and jumps to the flux computation ① again. However, if the hydrograph input is present, its properties are computed for the current time step ⑤. Another feature of the simulation node is the sparse block computation, adopted from [41]. Initially, all blocks are marked as wet. After evolving the solution in time, the integration kernel checks if there is at least one wet cell in the block and stores the information in the global memory. If a hydrograph is supplied and the sparse computation is enabled, the blocks containing hydrograph cells should be activated as well. If a block is active, its neighbors could be flooded in the next time step, thus, all adjacent blocks are activated, too. Indices of active blocks are compacted ⑥ and used by the flux and time integration kernels to process only wet blocks.

The system can perform the first-order Euler or second-order Runge-Kutta time integrations. For the second-order accuracy, we make two simulation iterations, whilst for the first-order Euler method, only the first iteration is needed. The first iteration follows all computation steps of the simulation loop. The second iteration requires only three computation steps (Figure III.6).

① **Flux computation** The flux kernel is computationally the most expensive of our parallel routines. It is responsible for computing the source terms and the fluxes over all four interfaces of each cell. We have identified the six most important steps in the code and labeled them from **(a)** to **(f)** in Figure III.7 and Figure III.8. Figure III.8 provides an overview of the code. Figure III.7 illustrates the role of the register and shared memory spaces of a block while the kernel is being executed. Our code design is guided by the principle of achieving a maximum amount of warp-synchronous computations. On the GPU, each warp processes 2×16 cells, i.e., two rows of a block at once. Due to this, threads of different columns are able to share data with each other without any performance loss. In contrast, the communication between threads across different rows is slower since it requires explicit warp synchronizations. Whenever possible, threads should exchange data across columns only. We suggest a method of data *transposition* to layout the required data in x-direction before the actual computations are done.

The kernel starts by allocating in shared memory four 2D-arrays required to hold the temporary variables. **(a)** The conserved variables $\bar{\mathbf{U}}_{j,k}$ and the bathymetry values at the right cell interface midpoints $B_{j,k+\frac{1}{2}}$, $B_{j+\frac{1}{2},k}$ are loaded from global memory into registers, totalling to five floating-point numbers per cell. **(b)** At this stage, we can directly proceed with the warp-synchronuous computation of the fluxes in x-direction without waiting for thread synchronization, since all required data has been loaded by the same warp. The computation exploits multiple shuffle instructions, factored out into the re-usable *calcFlux* subroutine (details follow later). **(c)** Next, we copy the bathymetry value in y-direction and the conserved variables to the shared memory. Notice, that they are assigned to the 2D-array element by swapping the x and y indices associated with the thread (transposition). Now we have to wait until this process has been completed by all threads from all warps within the block. An explicit synchronization barrier assures that all data is available for computing the flux in y-direction. Then we simply load the data from shared memory without swapping indices. **(d)** The stencil for the y-flux is now arranged in x-direction of the block, and we can re-use our efficient shuffle-based *calcFlux* function. One might argue that the necessary data in y-direction could be loaded from global memory again. However, this would significantly lower the performance, not only because the same values have to be loaded twice, but also due to coalescing rules. Parallel threads running the same access instruction to consecutive locations in global memory achieve the most efficient memory transfer. Since the data in global memory is organized row after row in a linear array, the access in y-direction is not favorable. The flux kernel proceeds with **(e)** transposing the computed y-fluxes. This way, the y-fluxes are now stored at the correct cell locations in shared memory. Next, we need to synchronize the threads to ensure that all threads have finished flux computations. **(f)** The kernel concludes by loading these values without swapping indices, and combines them with the x-flux and the source terms. The result is stored as a three-element vector in global memory.

In the *calcFlux* subroutine, the differences between KP and HWP become apparent (Figure III.9). HWP exhibits a more elaborate behavior from the GPU perspective. The relevant code sections are marked in red. HWP has to detect the partially flooded cells and treat them differently according to a set of rules. This results in a more complex branching pattern that slows down the process. For the computation of the point values at the cell interfaces (Figure III.1d), the subroutine acquires all the necessary data from the neighboring cells through shuffling the input values by one index, and in case of HWP, also by two indices. In KP, point values are reconstructed in the same way at all cells. HWP makes a distinction between partially flooded cells and fully wet cells. Moreover, the treatment of the partially flooded cells depends on the local bathymetry steepness and the amount of water contained in the neighboring cells. Consequently, the execution path divergence is greatly influenced by the current simulation status. There are multiple if-else blocks where we cannot guarantee that half-warps take the same execution path. This breaks the parallel execution of the warp and introduces some overhead compared to KP. After the reconstruction of all point values, the subroutine proceeds with computing the flux at the right cell interface (Figure III.1e). Here, the two schemes exhibit slight differences which are not crucial from the implementation point of view. The subroutine continues by acquiring the flux at the left interface through shuffling. The use of the shuffle instruction avoids to compute the same flux twice, first time for cell $C_{i,j}$ and second time for cell $C_{i+1,k}$. Finally, the fluxes at both interfaces are used to evaluate the flux at the current cell according to the central upwind flux function.

```
SUB: float3 calcFlux (w = waterLevel, dischargeX, dischargeY, bathRightInterfaceX)

// Fetch bathymetry at left interface by shuffling
float bathLeftInterfaceX = __shfl_up(bathRightInterfaceX , 1, blockSize); <- B_{j-1/2,k}

// Calculate average water level and determine whether the cell is partially flooded
// HWP specific
bool isPartiallyFlooded = waterDepth < (cellSize * 0.5f) *
                          abs(bathLeftInterfaceX  - bathRightInterfaceX ) / cellSize;

// Reconstruct water levels and discharges at cell interfaces

// Use shuffle  to fetch water levels needed for minmod reconstruction
float wCellLeft  = __shfl_up(w, 1, blockSize);   <- w_{j-1,k}
float wCellRight = __shfl_down(w, 1, blockSize); <- w_{j+1,k}
…

// Variables at left-left / right-right cells could be needed and shuffling must be
// done outside the following if-blocks to make sure that all data is available
float wCellLeftLeft      = __shfl_up(w, 2, blockSize);   <- w_{j-2,k}
float wCellRightRight  = __shfl_down(w, 2, blockSize);   <- w_{j+2,k}
…

if ( ! isPartiallyFlooded )
{
  // Compute point values for the conserved variables at the inner cell interfaces
  float2 wPointVal = reconstruct (wCellLeft , w, wCellRight);
  …
}
else  // HWP specific
{
  // Bathymetry steepness needed for cases below
  float bathSteepness = (bathRightInterfaceX - bathLeftInterfaceX) / cellSize;

  // At the partially flooded cells, branching occurs
  if ( isFullyFlooded(leftCell) && bathSteepness > 0 )
  {
    float2 wPointVal = reconstructAtPartial
                       (wCellLeftLeft, wCellLeft , w, bathLeftInterfaceX );
  }
  else if ( isPartiallyFlooded(leftCell) && bathSteepness > 0 )
  …
  …  // For the logic of all cases
}

// Equivalent reconstruction for point values of discharges
float dischargeXPointVal, dischargeYPointVal  = … ;

// Compute flux at right interface using the point values
// KP and HWP differ slightly here
float fluxRightInterfaceX = interfaceFlux
                    (wPointVal.y, wPointValCellRight.x, dischargeXPointVal.y, …);

float fluxLeftInterfaceX = __shfl_up(fluxRightInterfaceX, 1, blockSize);

// Conclude by computing the flux at the current cell
return  centralUpwind (fluxLeftInterfaceX, fluxRightInterfaceX);
```

Figure III.9: Subroutine *calcFlux* to evaluate the flux in any direction. The function assumes that the stencil is organized in x-direction, no matter if invoked for the computation of the y-flux. HWP requires additional work for the correct treatment of partially flooded cells (red parts).

As a side effect, the flux kernel evaluates the maximum speed value inside a block. This value is later needed to compute the time step size for the time integration. Our calculation of the

maximum speed value is based on the commonly used butterfly-reduction pattern. We exploit the exclusive-or shuffle instructions to accelerate this process.

② **Time step reduction** The time step reduction kernel is responsible for finding the highest speed value within the whole domain. Based on the values obtained for each block, this kernel employs a similar butterfly reduction to evaluate the maximum speed value for the entire grid. The maximum allowed time step size is then calculated according to the CFL condition (III.8).

③ **Time integration** The time integration kernel advances the solution in time and performs one sub-step of the Runge-Kutta integrator. Since the kernel does not require any ghost cells, its block size has the same dimensions as the inner block of the flux kernel. The kernel starts by reading the conserved variables $\bar{\mathbf{U}}_{j,k}$, the average bathymetry value $B_{j,k}$ at the cell center, and the combined fluxes $\mathbf{R}_{F+G}\left(\bar{\mathbf{U}}\right)_{j,k}$. After this, the solution advances in time. In addition, one has to ensure that no negative water heights are produced due to floating-point round-off errors.

④ **Global boundary conditions** Our global boundary condition kernel implements four types of boundary conditions using global ghost cells. The supported types are reflective or wall boundaries, inlet discharge, free outlet, and fixed water elevation. To implement these boundaries, we have to manually set both the cell averages and the reconstructed point values at the cell interface midpoints. Similarly to Brodtkorb et al. [39], we exploit the property of the minmod limiter. We recall that the minmod limiter uses the forward, central and backward difference approximations to the derivative, and always selects the least steep slope, or zero if any of them have opposite signs. We are allowed to set the reconstructed point values to arbitrary values as long as we ensure that the least steep slope is zero. To fulfill this condition, we need two ghost cells in each direction at the boundaries of the domain. Global ghost cells are updated at every step and thus they do not need any special treatment different from the handling of the interior cells.

The wall boundary condition is implemented by mirroring the last two interior cells at the boundary and changing the sign of the velocity component perpendicular to the boundary. The input discharge and fixed water elevations are implemented similarly. For the former, we set a fixed discharge value, whereas for the latter, a fixed water elevation. The free outlet is implemented by copying the values of the cell averages of the last interior cell to the two ghost cells.

⑤ **Local boundary conditions** Inside the domain, we support local boundary conditions to simulate phenomena such as sewer overflow scenarios or to apply a hydrograph. A hydrograph, describing a local water level and discharge over time, can be attached to multiple cells to impose time-varying inlet boundaries. At every time step, we compute the water level and velocity vectors for each affected cell by interpolating the hydrograph function values.

⑥ **Sparse computation** The sparse computation is used to exclude blocks that do not contain water. The time integration kernel marks the active blocks that contain wet cells or have cells with non-zero discharges. After this, we compact the indices of the active blocks. On the next iteration, these indices are loaded by the flux and integration kernels to find the offsets of the active blocks. If the sparse block computation is enabled, we have to activate blocks which contain hydrograph cells, since the hydrograph water depths could start at zero and increase later during the simulation. Therefore, it is necessary to track if the hydrograph cells become active. If they become inactive again, they will be excluded from the computation.

6 Evaluation

In this section, we provide a performance analysis of our latest, shuffle–based (SB) implementations of HWP and KP compared to previous, shared–memory–only (SMO) versions. The comparison is done using real-world scenarios. Our benchmarks run on a desktop PC equipped with a 3.4 GHz quad core Intel Xeon CPU and 16 GB RAM. We use an NVIDIA GeForce Titan graphics card which has 6 GB GDDR5 memory. GPU sources are compiled with CUDA v6.5 using the compiler flags shown in Table III.1.

`-arch=sm_35`	*NVIDIA GPU architecture to generate.*
`-maxrregcount=32`	*Max. amount of registers per thread.*
`-ftz=true`	*Flush denormals to 0.*
`-fmad=true`	*Force fused multiply-add operations.*
`-prec-div=false`	*Faster, but less accurate division.*
`-prec-sqrt=false`	*Faster, but less accurate square root.*
`-Xptxas -dlcm=ca`	*Increases the L1 cache to 48KB.*

Table III.1: NVCC flags used to compile CUDA source files.

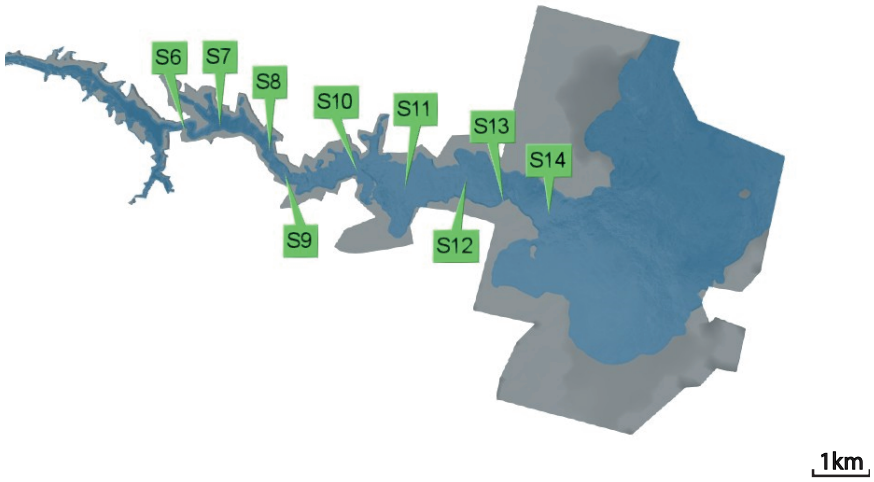


Figure III.10: Simulated dam break of Malpasset. The water extent at the time step 4000 s is shown (blue). The simulation results are verified with the experimental data obtained from laboratory models at the displayed locations (green labels).

6.1 Validation: Malpasset Dam Break

Before testing the simulation speed, we evaluate the correctness of our new solver using the well-known historical Malpasset dam break event. Our validation is based on the dataset available from the [105] samples. The original dataset consists of 104 000 unstructured points. The corresponding simulation grid contains 1149×612 cells, each cell of the size of 15×15 m. We set the desingularization constant $\epsilon = 0.40$ m and use two Manning coefficients $n = 0.025\,\mathrm{m}^{1/3}/\mathrm{s}$, and $n = 0.033\,\mathrm{m}^{1/3}/\mathrm{s}$. We use CFL = 0.25 in all our simulations and simulate the first 4000 seconds after the actual breach.

We employ the outcome of laboratory experiments on a 1:400 scale model to verify the correctness of our numerical results. In these experiments, researchers have recorded wave front arrival times [106] and maximum water elevations [107] at 14 wave gauge locations (S1-S14). Our verification uses only locations S6-S14, since no data is available for the other gauge locations. Figure III.10 displays the measurement points and the water extent after 4000 seconds of simulated model time.

Figures III.11-III.12 show our simulation results compared to the physical data acquired by the laboratory model. Overall, there is good agreement with the measurements. Small discrepancies between the scale model and the numerical results were also reported by others [39, 108, 101], and our results are consistent with these. We simulated two different roughness values for the terrain

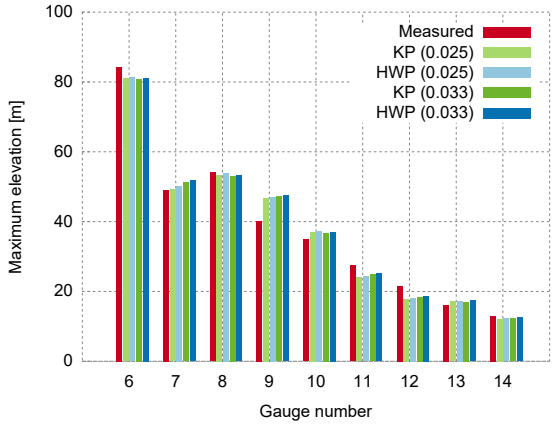


Figure III.11: Verification of the maximum water elevations during the Malpasset dam break event at nine gauge locations (S6-S14) for two roughness values (0.025 and 0.033).

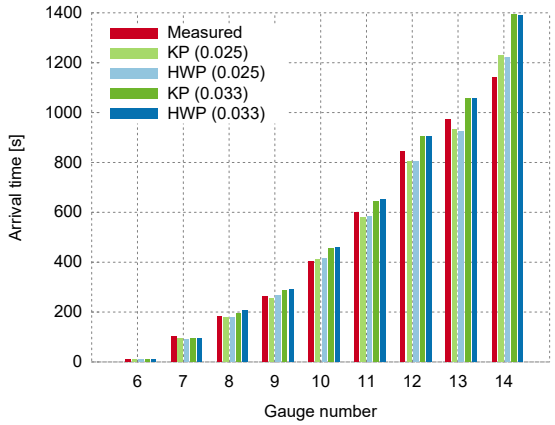


Figure III.12: Verification of the wave arrival times during the Malpasset dam break event at nine gauge locations (S6-S14) for two roughness values (0.025 and 0.033).

and compared the maximum water elevations and wave arrival times. The roughness value 0.025 is associated with gravelly earth channels, and the roughness value 0.033 corresponds to weedy, stony earth channels and floodplains with pasture and farmland. The higher the roughness value, the slower the flow velocity, which causes the water level to increase for the same discharge. This effect is visible in Figure III.11, where the elevation values are slightly increased when using the higher roughness value. The wave arrival times are more affected by the roughness value than the water elevations. A higher roughness value results in slower wave propagation. We note that, with a uniform roughness value across the whole domain, we can approximate the real-world process to a certain extent only. Usually, the roughness values vary spatially across the domain.

6.2 Performance Measurements

To evaluate the performance of the implemented solvers, we first examine the number of gigacells that can be processed per second. This quantity is computed by counting the number of wet cells that have been processed in one second of computational time. Thus, the quantity is independent of the actual number of time steps needed and provides a measure for the kernel performance. Figure III.13 shows how the performance decreases over time as the water spreads through the domain and the number of dry blocks decreases. The dashed lines show the percentage of the dry cells during the simulation. The HWP solver has smaller blocks, hence, it needs more blocks

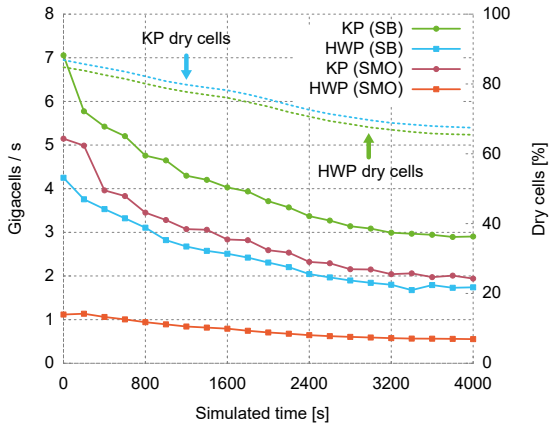


Figure III.13: Estimated solver performance, measured in gigacells per second, for the Malpasset dam break scenario (solid lines). The dashed lines show the percentage of dry blocks within the simulation domain, which is different for KP and HWP.

than the KP solver to cover the same domain. This explains the difference in the percentage of dry cells in the graph. One can notice that, due to simpler computations, the KP solver can process more cells per second than the HWP solver. Later in the text, we will show that the KP solver *has to* process much more cells for real-world scenario modeling, since it requires more time steps to simulate the same duration in long simulation runs. In the end, this fact makes the HWP solver significantly more efficient.

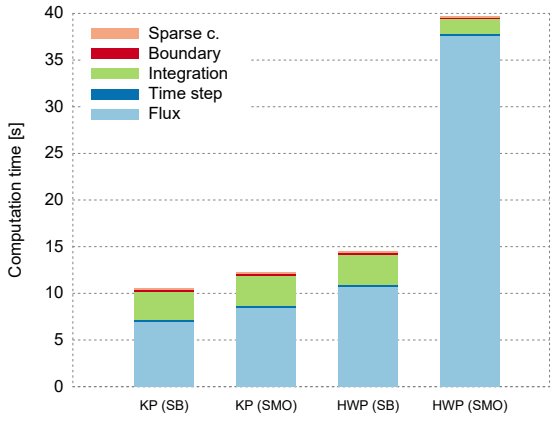


Figure III.14: Overall GPU runtime of the implemented solvers for the simulation of 4000 seconds of the Malpasset dam break event, including the time distribution over five phases of a single computation iteration.

Figure III.14 displays the absolute GPU runtime of the solvers required to simulate the first 4000 seconds of the Malpasset dam break event. While the previous SMO implementation of the HWP scheme is obviously slower, the new shuffle-based approach for HWP exhibits clear improvements. Its running times are similar to those of the KP implementations. Even though the newly implemented SB solver for KP completes the 4000 seconds simulation slightly faster, the situation changes in real-world scenarios, where hours or days of flooding have to be simulated. The difference becomes dramatic when ensembles of such scenarios are used for uncertainty treatment.

In Figure III.14 we see that the computation times for the two KP solvers are only slightly different, while there is a significant difference for the HWP solvers. For the KP (SB) solver we cannot further improve the GPU utilization by reducing the shared memory footprint, since we are already limited by the number of registers per block, i.e., the GPU cannot launch more blocks

on a multiprocessor. However, the HWP (SMO) solver requires more shared memory than KP (SMO), and more synchronization barriers, which are responsible for the performance loss. The use of the shuffle instructions (SB) avoids the need for some of the synchronization barriers, since threads in the same warp are always executed in a synchronous way. Moreover, this leads to a reduced shared memory footprint, thus to improved GPU utilization and a high performance gain.

Figures III.15-III.16 show the computation time and the average number of time steps per second for the first 10000 seconds of the Malpasset dam break event. In Figure III.15, we see that the KP solver is faster for the first 5600 seconds. However, over time, it has to perform more and more time steps (Figure III.16). Thus, it requires more computation time than HWP as the simulation progresses.

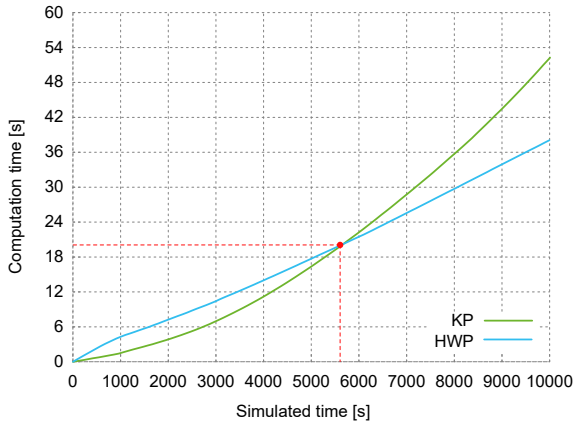


Figure III.15: Computation time of the two shuffle-based solvers for the simulation of 10000 seconds of the Malpasset dam break event. The red dot shows the intersection of the simulated and the computation time of the solvers.

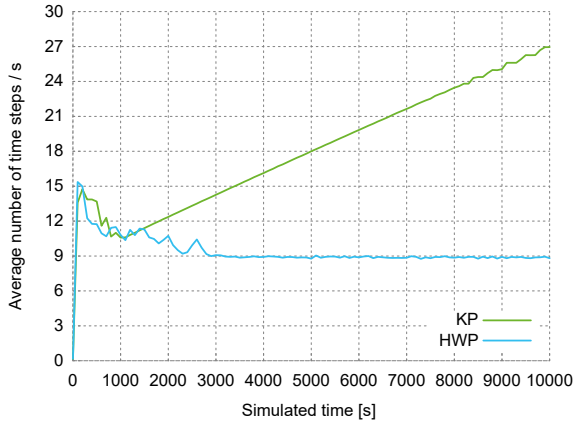


Figure III.16: Average number of time steps per second required by the two shuffle-based solvers for the simulation of 10000 seconds of the Malpasset dam break event.

We now show that for real-world simulations, our shuffle-based implementation of HWP outperforms KP, since it actually requires considerably fewer time steps to solve the same problem. As a test case, we use the overtopping of mobile flood protection walls in Cologne, Germany (Figure III.17b). Such overtopping happens when the water in the Rhine river raises above the level of 11.9 m, marked with a red line in Figure III.17a. A standard way to use uncertain overtopping predictions for action planning is to simulate ensembles of overtopping events. In our test case, we simulate an ensemble of 10 overtopping events corresponding to water levels from 11.95 m to 12.95 m, displayed by a set of hydrographs in Figure III.17a. The simulation domain of approximately 3.1×7.5 km is shown in Figure III.17c. The corresponding grid contained approximately

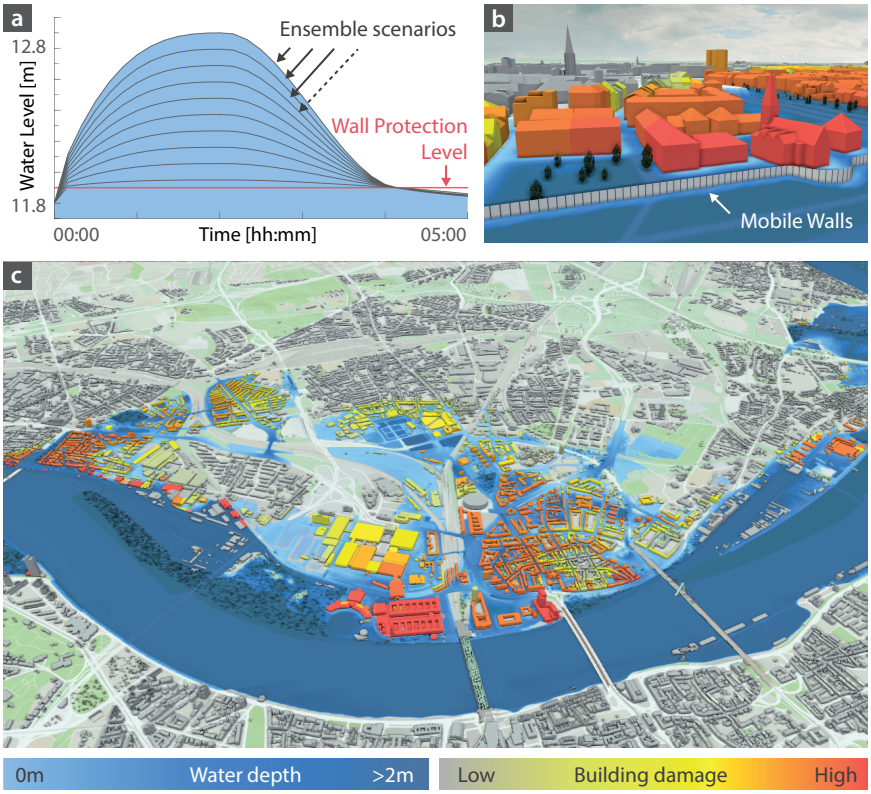


Figure III.17: Uncertainty-aware prediction of mobile flood protection wall overtopping in Cologne. (a) Input hydrographs forming an ensemble of 10 different scenarios with varying peak levels. (b,c) Visualization of ensemble results. Buildings are colored according to the expected damage. The terrain is colored according to the average water depth.

2.6 million cells of 3×3 m. For each ensemble member, we decided to simulate only the first 2 hours of overtopping, since the KP solver was too slow. In Figure III.17b,c, the expected building damage and water depths, aggregated over the whole ensemble, are displayed with colors. Figure III.18a shows the computation runtimes for each simulated scenario. These are measured for the SMO and SB implementations of the KP and HWP schemes. Clearly, the shuffle-based implementation of HWP exhibits the best performance, closely followed by the shared-memory-only implementation of the same scheme. Both implementations of KP turn out to be considerably slower. This is due to high velocities eventually generated at dry/wet boundaries and, consequently, smaller time step sizes. In turn, a larger number of smaller time steps imply more computation effort to advance the simulation up to the required model time. This computation overhead multiplies with the size of the ensemble. For illustration, we compare the number of time steps required by both schemes to simulate every scenario (Figure III.18b). Note that the number of time steps depends solely on the scheme and not on the implementation (SB or SMO), hence two groups of bar charts instead of four. As one can see in Figure III.18b, there is a large, sometimes up to an order of magnitude, difference between the numbers of time steps required by the two schemes, which proves our assertion.

To sum up, we discuss the main factors responsible for the differences in the number of time steps between the presented case studies. The first factor is the flow velocity. In case of the Malpasset dam break, the flow velocity is very high due to the high water level at the dam. At the beginning of the simulation, the error in the velocities along the dry/wet boundaries is negligible compared to the high flow velocity in the middle of the stream. As time advances, the reservoir is draining and the flow velocity decreases. This allows for a smaller number of time steps (see the first 1000 seconds in Figure III.16). However, for KP, the error is growing along the boundaries, causing an increase in the number of time steps again. In the Cologne case, the water depth is very shallow with low flow velocities, which means that the effects of the error

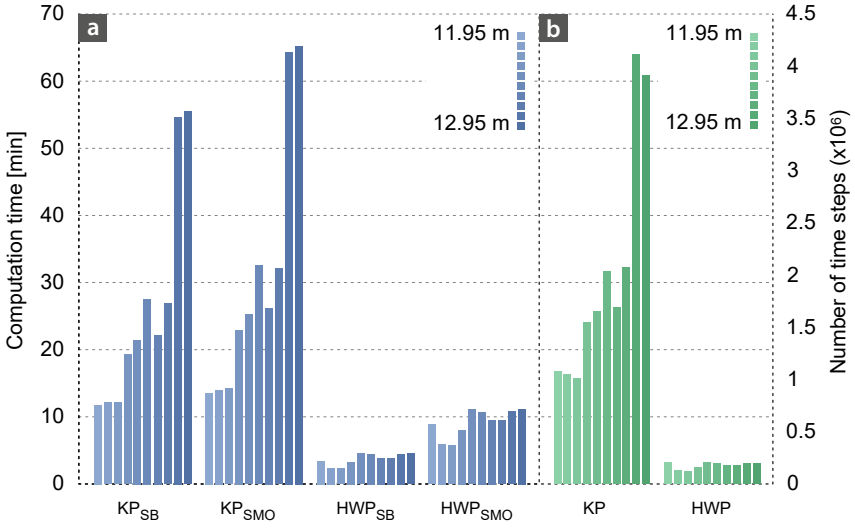


Figure III.18: Runtimes and number of time steps for the first 2 hours of model time for each overtopping scenario of the simulated ensemble for Cologne.

accumulation along the dry/wet boundaries occurs sooner. The second factor is the resolution of the bathymetry. It is 3×3 meters in Cologne, which is $5\times$ higher than in the Malpasset case, where we use a cell size of 15×15 meters. This increases the rate of the accumulation of the error, since the computation of the time step size depends not only on the maximum velocity but also on the grid resolution (III.8).

7 Conclusion

In recent years, a lot of effort has been made towards efficient GPU-based solvers for shallow water equations on Cartesian grids. However, their application to real-world scenarios is often plagued with significant slow-downs. This happens due to spurious high velocities generated by the corresponding numerical schemes at dry/wet boundaries. These high velocities sometimes lead to time steps so small that the simulation barely advances in time. Such slow-downs are unacceptable for real-world decision making in flood management, where large ensembles of flood scenarios have to be simulated to handle the prediction uncertainty. Keeping this problem in mind, we developed the HWP scheme [1] that stabilizes the velocities and increases the time step size but requires a more complex algorithm. It should also be noted that, even in shorter run times where KP outperforms HWP in runtime efficiency, the HWP scheme has a crucial advantage over KP: it produces simulations that are more physically consistent. For instance, it avoids the buildup of unrealistically high velocities close to boundaries, which actually ends up contributing to the loss of efficiency of KP for longer runs.

In this paper, we present a novel GPU-based implementation that fully reveals the potential of the HWP scheme. The implementation is optimized for the Kepler GPU architecture. In order to achieve high utilization, we exploit the capabilities of the shuffle instructions. Due to their warp-synchronous nature, we can share data between the threads and thus avoid explicit synchronizations. However, we have to re-organize data using the shared memory to be able to use them for the second dimension efficiently. As demonstrated by the evaluation, our shuffle-based implementation of HWP outperforms the existing alternatives significantly when applied to real-world scenarios. This allows for the use of the presented GPU solver for real-world decision making, where, for instance, imminent floods can be modeled and analyzed in time-critical situations. Moreover, it can be efficiently used for planning purposes where a large number of scenarios need

to be explored prior to any flood events.

Nevertheless, open questions remain, motivating further improvements. Reconstruction cases in the presented solver lead to a conditional branch divergence in the code. To avoid performance loss, these need to be carefully optimized. Furthermore, with the new Thrust library v1.8, which will support CUDA streams, we will be able to increase the GPU utilization. This will further reduce the overall simulation time. Another interesting direction for future work is to provide the solver with the support for adaptive grids for handling even larger simulation domains.

Acknowledgments

This work was supported by grants from the Austrian Science Fund (FWF) project number W1219-N22 (Vienna Doctoral Programme on Water Resource Systems) and from the Vienna Science and Technology Fund (WWTF) project number ICT12-009 (Scenario Pool), and from the European Research Council (ERC) Advanced Grant ”FloodChange”, project number 291152. We thank the flood protection centre of Steb Cologne, AöR.

Chapter IV

Comparison and Validation of Shallow Water Schemes on Analytic, Laboratory and Real-World Cases

Zsolt Horváth, Andreas Buttinger-Kreuzhuber, Daniel Cornel, Artem Konev, Jürgen Komma, Günter Blöschl, Sebastian Noelle, Jürgen Waser

This chapter is based on a paper which is submitted to the *International Journal for Numerical Methods in Fluids*.

Abstract

Two-dimensional shallow water schemes on Cartesian grids are amendable for graphics processing units and thus a convenient choice for fast flood simulations. A comparison of recent schemes and an exhaustive validation on important use cases is essential for developers and practitioners working with flood simulation tools. In this paper, we discuss three state-of-the-art shallow water schemes, one by Kurganov and Petrova (KP), its successor by Horváth et al. (HWP), and our two-dimensional extension of the scheme by Chen and Noelle (CN). We analyse the advantages and disadvantages of each scheme on an extensive list of scenarios including several analytical and laboratory cases as well as a representative set of three historical floods. To enable the real-world studies, we address the implementation of the required boundary conditions (BCs), such as wall BCs, discharge BCs and water level BCs. We observe that the KP and HWP schemes are more accurate than the CN scheme in low-resolution real-world cases. However, at higher resolution and at laboratory cases with curved walls they suffer from unphysical oscillations. Overall, good agreement is observed for all case studies rendering these shallow water schemes suitable for flood management applications.

1 Introduction

There is a high demand for appropriate simulation tools that are capable of providing accurate and fast predictions for flood management. In most flood simulation tools, such as HEC-RAS [109], TELEMAC [110] and Visdom [47, 111], the shallow water equationss (SWEs) are the underlying model equations. They provide plausible and reliable results of water levels for events, such as river floods, dam breaks, levee breaches [14, 39, 112, 107, 86, 62]. The SWEs are based on the assumption that the horizontal length scale is large compared to the vertical length scale and they can be derived by depth-averaging the Navier-Stokes and the continuity equations [11].

A common numerical method for solving hyperbolic partial differential equations (PDEs), such as the homogeneous SWEs, is the finite volume method (FVM). A computational grid has to be chosen as a basis for the spatial discretization of the SWEs. Unstructured triangular meshes are able to incorporate complex geometries, however they require time-consuming mesh generation. In contrast, rectangular grids lack this pre-processing step at the expense of poor resolution of topographic features not aligned with the grid. However, one can improve cartesian grids by incroporating additional techniques such as cut-cells [25, 113]. Still, the striking advantage of cartesian grids is their suitability for straightforward parallelization on GPUs due to their simplicity [39, 114, 115]. The advent of GPU computing has reduced computation times by a factor of up to 100 compared to conventional models [116, 114, 115]. This speed-up in computation time does not only allow to increase the number of cells, it also decreases model uncertainties, as ensemble simulation becomes a viable option. Fast shallow water schemes can be proficiently applied to interactively study the effects of protection measures to prevent flood damages [46].

Three state-of-the-art shallow water schemes, the scheme of Kurganov and Petrova (KP) [18], the scheme of Horváth et al. (HWP) [1], and the scheme of Chen and Noelle (CN) [117] are compared in this paper. The HWP scheme is a successor of the KP scheme, improving it at wet/dry fronts. The CN scheme is based on the hydrostatic reconstruction (HR) method introduced in [14]. An extension of the CN scheme to two dimensions is presented here. The advantages and disadvantages of the schemes are shown by simulating a comprehensive list of scenarios. We validate two analytic cases, a parabolic basin and a parabolic bump, and three laboratory cases, a U-shaped flume, a sine-generated flume, and a triangular hump. The first two laboratory cases mimic me-

andering rivers. The real-world validation cases used here have different terrain features, ranging from relatively flat lowlands to river valleys with highly varying bottom topography. We include the following real-world case studies: the Malpasset dam break in France from 1959, the 2011 Danube river flooding in the Lobau national park, and the 2013 Danube river flooding in the Wachau valley, both in Austria.

This paper goes beyond the existing literature by the following key points:

- Extension of the recent CN shallow-water scheme to two dimensions.
- Comparison of the KP, HWP and CN scheme regarding quality of simulation results and performance on real-world scenarios.
- Validation of the simulator on analytical and real-world (large-scale) cases including different boundary conditions.
- Practical implementation of flux boundaries in the case of walls, free outlets, and prescribed hydrographs, which do not need ghost cells in the numerical boundary treatment.

2 Related Work

Modern flood management tools require sufficiently accurate results while still being performant. We focus on schemes that combine accuracy, stability and efficiency at the same time. In order to satisfy these needs, the scheme should be able to handle dry and near dry states, and solve problems characterized by strong discontinuities, e. g., dam breaks. To simulate real-world floods over time periods of days, it is important for the solution to remain stable, e. g. the scheme should not produce spurious oscillations of the velocities. The scheme should be positivity-preserving, i. e., it should guarantee the non-negativity of the water depth. It should also be well-balanced, i. e., it should capture equilibria [16, 118, 119, 100, 61, 62, 63]. One simple steady state is the *lake-at-rest* equilibrium, which is a combination of still water and wet/dry boundaries. It is satisfied by most of the recently developed schemes [14, 120, 1, 18]. There exist also schemes that are even capable of preserving subsonic steady states [121] or achieving exact well-balancedness [122].

The basis for the KP and HWP schemes is a second-order scheme developed by Kurganov and Levy [17] on a regular grid. The scheme assumes a continuous bathymetry and uses a different reconstruction of the water surface in near dry areas than in the wet zones. The resulting scheme is not well-balanced and violates mass conservation. Furthermore, spurious waves may emerge in the shoal zones affecting the Courant-Friedrichs-Lewy (CFL) number [65] and finally restricting the time steps toward very small values. Kurganov and Petrova [18] improve the previous work by supporting a discontinuous bathymetry. They describe a reconstruction adjustment for the partially flooded cells, where values of the water depth become negative at the integration points. If the reconstructed water slope creates negative values at the integration points, they adjust the steepness of the slope so that the negative values become zero. Their correction solves the positivity problem by raising and lowering the water level at the left and right side of the cell according to the bathymetry function. This guarantees that all water heights are non-negative. However, at partially flooded cells, this can lead to large errors for small water heights, and the flow velocity will grow smoothly in these formerly dry areas. Another issue related to this modification is that the water climbs up on the shores at the dry/wet boundaries. Moreover, if a cell gets wet, it will almost never get completely dry again.

Bollermann [16] extends the Kurganov and Petrova [18] scheme and achieves well-balanced states in the partially flooded cells by constructing an alternative correction procedure, which is similar to the reconstruction used by Tai [70] to track fronts for granular avalanches. However, this modification is only introduced in one dimension and can lead to infinitely small time steps.

To overcome this time step limitation, Bollermann introduces a constraint on the time step, the so-called draining time technique [67]. This scheme is extended to two dimensions by Horváth et al. [1] and proved to be superior regarding performance than the original KP scheme [18]. This is achieved by removing the aforementioned defects at partially flooded cells.

The CN scheme [117], is based on the HR scheme of Audusse [14]. It uses a piecewise-constant bathymetry which leads to the known difficulty of discontinuous measures. The original HR scheme does not properly account for the acceleration of shallow water downhill flow [123]. Morales et al. [124] improve the existing HR scheme in the case of partially wet interfaces. In [117] a way to investigate and derive the two existing HR schemes by means of subcell reconstructions is presented. There exists also a second-order extension to the original HR scheme which is presented by Audusse and Bristeau [15].

3 Numerical Schemes

In this section we summarize the underlying numerical theory of the shallow water schemes that we compare and validate. The hyperbolic conservation law described by the two-dimensional shallow water equations of the Saint–Venant (SV) system can be written as:

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix} + \begin{bmatrix} 0 \\ -gu\sqrt{u^2+v^2}/C^2 \\ -gv\sqrt{u^2+v^2}/C^2 \end{bmatrix}, \tag{IV.1}$$

where h represents the water height, hu is the discharge along the x-axis, hv is the discharge along the y-axis (Figure IV.1a), u and v are the average flow velocities, g is the gravitational constant, B is the bathymetry (assumed to be time-independent), and C is the Chézy friction coefficient. Subscripts represent partial derivatives, i.e., $\mathbf{U}_t$ stands for $\frac{\partial \mathbf{U}}{\partial t}$.

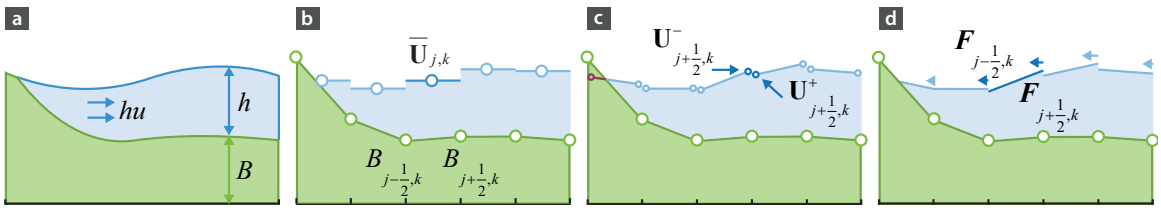


Figure IV.1: Schematic view of a shallow water flow, definition of the variables, and flux computation. a) Continuous variables. b) The conserved variables $\mathbf{U}$ are discretized as cell averages $\bar{\mathbf{U}}_{j,k}$. The bathymetry function B is reconstructed at cell interface midpoints. c) Left- and right-sided point values are computed at cell interface midpoints. The red circle indicates that a negative water height is computed. Since water heights cannot be negative, they are corrected before the flux computation. d) Fluxes are computed using the HLL flux function at the cell interfaces.

In vector form the system can be written as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U}, B)_x + \mathbf{G}(\mathbf{U}, B)_y = \mathbf{S}_B(\mathbf{U}, B) + \mathbf{S}_f(\mathbf{U}), \tag{IV.2}$$

where $\mathbf{U} = [h, hu, hv]$ is the vector of conserved variables, $\mathbf{F}$ and $\mathbf{G}$ are flux functions. $\mathbf{S}_B$ stands for the bed slope term and models the fluid's acceleration due to the gravitational forces. $\mathbf{S}_f$ represents the bed friction source terms. To provide realistic water flow, a friction term is introduced in the laboratory and real-world scenarios.

Following [117], there are two important steady-state equilibria. First, the *still-water* equilibrium, i. e.,

$$u, v = 0 \text{ and } w_x, w_y = 0, \tag{IV.3}$$

where w denotes the water level $w = h + B$. Second, the *lake at rest* equilibrium, which includes dry shores, i. e.,

$$hu, hv = 0 \text{ and } h\,\partial_x w, h\,\partial_y w = 0. \tag{IV.4}$$

If a numerical scheme is capable of balancing source and numerical flux terms for stationary solutions it is called *well-balanced*. All of the three presented schemes are well-balanced in the one-dimensional case, in the two-dimensional case only the HWP and the CN scheme are well-balanced.

3.1 Discretization

We choose a uniform grid $x_\alpha := \alpha\Delta x$ and $y_\beta := \beta\Delta y$, where Δx and Δy are the cell sizes. We denote by $C_{j,k}$ the cell $C_{j,k} := [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}]$. The SWE are discretized by the method of lines. For clarity of presentation we omit additional source terms besides the topography for now. The FVM is chosen for the spatial discretization on top of the uniform grid. An FVM discretizes the conserved variables $\bar{\mathbf{U}}$ as cell averages, e. g., $\bar{\mathbf{U}}_{j,k}$ for the finite volume $C_{j,k}$. This yields a system of ordinary differential equations for the cell averages

$$\frac{d}{dt}\bar{\mathbf{U}}_{j,k}(t) = -\frac{\mathbf{F}_{j+\frac{1}{2},k}(t) - \mathbf{F}_{j-\frac{1}{2},k}(t)}{\Delta x} - \frac{\mathbf{G}_{j,k+\frac{1}{2}}(t) - \mathbf{G}_{j,k-\frac{1}{2}}(t)}{\Delta y} + \mathbf{S}_{j,k}(t), \tag{IV.5}$$

where $\mathbf{F}_{j\mp\frac{1}{2},k}$ and $\mathbf{G}_{j,k\mp\frac{1}{2}}$ are the discretized fluxes and $\mathbf{S}_{j,k}$ is an appropriate source term discretization. The interface fluxes are computed by the Harten-Lax-van Leer (HLL) flux [76]. The following steps are executed in an FVM:

Step 1: Initialization/Update of the cell averages $\bar{\mathbf{U}}_{j,k}$ for all interior cells by an ODE solver, see Figure IV.1b

Step 2: Reconstruction of the interface values, see Figure IV.1c

- For partially wet cells a special reconstruction might be used

Step 3: Computation of the interface fluxes with the HLL flux, see Figure IV.1d

- At boundary interfaces the fluxes are computed according to the boundary type

Step 4: Computation of the source terms

Not all of the sub-steps are required in each scheme. Step 1 is the same for all schemes and is either given by the initial condition or by the common time integration, as explained in detail after the spatial discretizations in Section 3.5. Steps 2-4 will guide us through the upcoming sections, as they are organized in sub-sections describing bathymetry reconstruction, point value reconstruction, special point value reconstruction in the case of wet/dry fronts, numerical fluxes and source terms.

3.2 The KP and HWP Scheme

Bathymetry Reconstruction

For the second-order schemes KP and HWP the original bathymetry $\hat{B}$ is replaced with a piecewise bilinear approximation $B^{\mathrm{KP}}(x, y)$, which is the same for the KP and HWP scheme, i. e. $B^{\mathrm{KP}}(x, y) =$

$B^{\mathrm{HWP}}(x,y)$. The original bathymetry values $\hat{B}(x,y)$ are sampled at the cell vertices $j \pm \frac{1}{2}, k \pm \frac{1}{2}$ to set up the continuous, piecewise bilinear approximation $B^{\mathrm{KP}}(x,y)$. Thus, the vertex values $B^{\mathrm{KP}}_{j\pm\frac{1}{2},k\pm\frac{1}{2}} = \hat{B}_{j\pm\frac{1}{2},k\pm\frac{1}{2}}$ of the cell $C_{j,k}$ are then used to derive the reconstructed bathymetry values at the cell center:

$$\bar{B}^{\mathrm{KP}}_{j,k} = \frac{1}{4}\left(B^{\mathrm{KP}}_{j+\frac{1}{2},k} + B^{\mathrm{KP}}_{j-\frac{1}{2},k} + B^{\mathrm{KP}}_{j,k+\frac{1}{2}} + B^{\mathrm{KP}}_{j,k-\frac{1}{2}}\right), \tag{IV.6}$$

which is also equal to the average value of $B^{\mathrm{KP}}(x,y)$ in the cell $C_{j,k}$. The bathymetry $B^{\mathrm{KP}}(x,y)$ of each cell $C_{j,k}$ is given by a bilinear form. For details see [18].

Point Value Reconstruction

For second-order accuracy, left- and right-sided point values have to be computed at the cell interface midpoints through slopes taking into account the neighbouring values. The left- and right-sided point values are given by $\mathbf{U}^{\pm}_{j\mp\frac{1}{2},k} = \bar{\mathbf{U}}_{j,k} \mp \frac{\Delta x}{2}(\mathbf{U}_x)_{j,k}$ and $\mathbf{U}^{\pm}_{j,k\mp\frac{1}{2}} = \bar{\mathbf{U}}_{j,k} \mp \frac{\Delta y}{2}(\mathbf{U}_y)_{j,k}$. To suppress unphysical oscillations, a minmod-limiter is applied to the slopes [76]

$$\begin{aligned}
(\mathbf{U}_x)_{j,k} &= \mathrm{minmod}\left(\theta\frac{\bar{\mathbf{U}}_{j,k} - \bar{\mathbf{U}}_{j-1,k}}{\Delta x}, \frac{\bar{\mathbf{U}}_{j+1,k} - \bar{\mathbf{U}}_{j-1,k}}{2\Delta x}, \theta\frac{\bar{\mathbf{U}}_{j+1,k} - \bar{\mathbf{U}}_{j,k}}{\Delta x}\right), \\
(\mathbf{U}_y)_{j,k} &= \mathrm{minmod}\left(\theta\frac{\bar{\mathbf{U}}_{j,k} - \bar{\mathbf{U}}_{j,k-1}}{\Delta y}, \frac{\bar{\mathbf{U}}_{j,k+1} - \bar{\mathbf{U}}_{j,k-1}}{2\Delta y}, \theta\frac{\bar{\mathbf{U}}_{j,k+1} - \bar{\mathbf{U}}_{j,k}}{\Delta y}\right),
\end{aligned} \tag{IV.7}$$

where θ is a predefined parameter, set to 1.3 in our simulations.

Special Reconstruction at Wet/Dry Fronts

Special reconstruction methods are introduced to keep the scheme positivity-preserving and well-balanced in the case of wet/dry fronts in the KP and HWP scheme. The KP scheme violates the well-balancedness for partially wet cells. In the KP scheme, the velocities of "nearly dry" cells are reconstructed in a special way to avoid high velocities. Specifically, the velocities are desingularized according to

$$u = \frac{\sqrt{2}h(hu)}{\sqrt{h^4 + \max(h^4, \epsilon)}}, \quad v = \frac{\sqrt{2}h(hv)}{\sqrt{h^4 + \max(h^4, \epsilon)}}, \tag{IV.8}$$

where ϵ is a small apriori chosen positive number. This reconstruction has to be used cautiously and consistently in the code since the correction factor is below one for $h^4 < \epsilon$. Thus, discharges computed by a reconstruction of the desingularized velocities $u = u^{\epsilon}(h, hu)$ and subsequent multiplication by h do not recover the original discharges, that is, $h \cdot u \neq hu$. In our simulations we do not use the specified desingularization but a cut-off for the velocities in nearly dry regions.

Horváth et al. [1] propose a correction for partially wet cells to obtain a well-balanced reconstruction. In the HWP scheme, the slope is modified to avoid negative water heights at the almost dry cells, and a separation point is generated. The separation point is deduced from the intersection of intermediate horizontal water lines and the bathymetry. Another improvement of the HWP scheme is the positivity preserving time integration at partially flooded cells. This so-called draining time step technique is presented in [16] for the one-dimensional case and is extended to two dimensions in [1]. By considering the outgoing mass flux it is possible to determine a time step that keeps the cell dry after applying the time integration for this time step. To ensure well-balancedness, a splitting of the fluxes into gravitational and advective parts is necessary and a specialized time integration in the nearly dry cells is needed. Effectively, a slope hit by a wave can dry out if the source terms force the water back. In previous schemes, a small amount of water has to remain to avoid high velocities, which in turn may lead to slow simulations.

Numerical Fluxes

The discretized fluxes are obtained through an approximate Riemann solver, namely the HLL flux. The central-upwind flux function is a direct generalization of the HLL flux [1]. The interface point values are then used as Riemann states for the numerical fluxes

$$\mathbf{F}_{j+\frac{1}{2},k} = \frac{a^+_{j+\frac{1}{2},k}\mathbf{F}\left(\mathbf{U}^-_{j+\frac{1}{2},k}, B_{j+\frac{1}{2},k}\right) - a^-_{j+\frac{1}{2},k}\mathbf{F}\left(\mathbf{U}^+_{j+\frac{1}{2},k}, B_{j+\frac{1}{2},k}\right)}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} \tag{IV.9}$$

$$+ \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}\left[\mathbf{U}^+_{j+\frac{1}{2},k} - \mathbf{U}^-_{j+\frac{1}{2},k}\right], \tag{IV.10}$$

$$\mathbf{G}_{j,k+\frac{1}{2}} = \frac{b^+_{j,k+\frac{1}{2}}\mathbf{G}\left(\mathbf{U}^-_{j,k+\frac{1}{2}}, B_{j,k+\frac{1}{2}}\right) - b^-_{j,k+\frac{1}{2}}\mathbf{G}\left(\mathbf{U}^+_{j,k+\frac{1}{2}}, B_{j,k+\frac{1}{2}}\right)}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}} \tag{IV.11}$$

$$+ \frac{b^+_{j,k+\frac{1}{2}}b^-_{j,k+\frac{1}{2}}}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}}\left[\mathbf{U}^+_{j,k+\frac{1}{2}} - \mathbf{U}^-_{j,k+\frac{1}{2}}\right], \tag{IV.12}$$

at the interface $j+\frac{1}{2},k$. The speed values a and b are functions of the eigenvalues λ of the Jacobian, i. e.

$$\begin{aligned} a^+_{j+\frac{1}{2},k} &= \max\left\{\lambda^-_{j+\frac{1}{2},k}, \lambda^+_{j+\frac{1}{2},k}, 0\right\} \\ &= \max\left\{u^-_{j+\frac{1}{2},k} + \sqrt{gh^-_{j+\frac{1}{2},k}},\ u^+_{j+\frac{1}{2},k} + \sqrt{gh^+_{j+\frac{1}{2},k}},\ 0\right\}. \end{aligned}$$

$$a^-_{j+\frac{1}{2},k} = \min\left\{u^-_{j+\frac{1}{2},k} - \sqrt{gh^-_{j+\frac{1}{2},k}},\ u^+_{j+\frac{1}{2},k} - \sqrt{gh^+_{j+\frac{1}{2},k}},\ 0\right\}$$

and analogously for $b^\pm_{j+\frac{1}{2},k}$.

Source Terms

For the KP and the HWP schemes the discretized source terms are [17, 18]

$$\mathbf{S}^{\mathrm{KP},(1)}_{j,k} := 0, \tag{IV.13}$$

$$\mathbf{S}^{\mathrm{KP},(2)}_{j,k} := -g\bar{h}_{j,k}\frac{B^{\mathrm{KP}}_{j+\frac{1}{2},k} - B^{\mathrm{KP}}_{j-\frac{1}{2},k}}{\Delta x}, \tag{IV.14}$$

$$\mathbf{S}^{\mathrm{KP},(3)}_{j,k} := -g\bar{h}_{j,k}\frac{B^{\mathrm{KP}}_{j,k+\frac{1}{2}} - B^{\mathrm{KP}}_{j,k-\frac{1}{2}}}{\Delta y}, \tag{IV.15}$$

where we use the approximated bathymetry values at cell interface midpoints.

3.3 The CN Scheme

Bathymetry Reconstruction

In the CN scheme [117], the bathymetry is given at the cell center $B^{\mathrm{CN}}_{j,k} = \hat{B}(x_j, y_k)$. The interface bathymetry values are reconstructed from the bathymetry and the water levels of the adjacent

cells as

$$B^{\mathrm{CN}}_{j\pm1/2,k} = \min(\max(\bar{B}^{\mathrm{CN}}_{j\pm1,k}, \bar{B}^{\mathrm{CN}}_{j,k}), \min(\bar{w}_{j\pm1,k}, \bar{w}_{j,k})). \tag{IV.16}$$

This reconstruction is an extension of the original hydrostatic reconstruction, which featured only an upwind evaluation of the bottom [14]. Here we omit the time dependence of the water level w, as we do in the following equations. Only the first maximum in (IV.16) can be precomputed for the whole simulation, the bathymetry interface values have to be reconstructed at each time step. The reconstructed bathymetry values are the same at the left and the right side of an interface.

Point Value Reconstruction

To achieve well-balancedness it is necessary to introduce a special reconstruction for the Riemann states which are fed into the approximate Riemann solver. The CN scheme defines the interface heights as

$$\begin{aligned} h^{\mathrm{CN}}_{j+\frac{1}{2}-,k} &= \min(\bar{w}_{j,k} - B^{\mathrm{CN}}_{j+\frac{1}{2},k}, \bar{h}_{j,k}), \\ h^{\mathrm{CN}}_{j+\frac{1}{2}+,k} &= \min(\bar{w}_{j+1,k} - B^{\mathrm{CN}}_{j+\frac{1}{2},k}, \bar{h}_{j+1,k}), \end{aligned} \tag{IV.17}$$

in x-direction and analogously in y-direction.

Special Reconstruction at Wet/Dry Fronts

No extra treatment for partially wet cells is needed for the CN scheme, since it is already included in the definition of the interface depths and the source term, see (IV.17) and (IV.19)-(IV.20).

Numerical Fluxes

The same flux as for the KP and HWP scheme is used, namely the HLL flux, see (IV.9).

Source Terms

The source terms in the CN scheme are given as

$$\mathbf{S}^{\mathrm{CN},(1)}_{j,k} := 0, \tag{IV.18}$$

$$\begin{aligned} \mathbf{S}^{\mathrm{CN},(2)}_{j,k} &:= \mathbf{S}^{\mathrm{CN},(2)}_{j-\frac{1}{2}+,k} + \mathbf{S}^{\mathrm{CN},(2)}_{j+\frac{1}{2}-,k} \\ &:= -g\frac{h_{j,k} + h^{\mathrm{CN}}_{j-\frac{1}{2}+,k}}{2}\frac{B^{\mathrm{CN}}_{j,k} - B^{\mathrm{CN}}_{j-\frac{1}{2},k}}{\Delta x} - g\frac{h_{j,k} + h^{\mathrm{CN}}_{j+\frac{1}{2}-,k}}{2}\frac{B^{\mathrm{CN}}_{j+\frac{1}{2},k} - B^{\mathrm{CN}}_{j,k}}{\Delta x}, \end{aligned} \tag{IV.19}$$

$$\begin{aligned} \mathbf{S}^{\mathrm{CN},(3)}_{j,k} &:= \mathbf{S}^{\mathrm{CN},(3)}_{j,k-\frac{1}{2}+} + \mathbf{S}^{\mathrm{CN},(3)}_{j,k+\frac{1}{2}-} \\ &:= -g\frac{h_{j,k} + h^{\mathrm{CN}}_{j,k-\frac{1}{2}+}}{2}\frac{B^{\mathrm{CN}}_{j,k} - B^{\mathrm{CN}}_{j,k-\frac{1}{2}}}{\Delta y} - g\frac{h_{j,k} + h^{\mathrm{CN}}_{j,k+\frac{1}{2}-}}{2}\frac{B^{\mathrm{CN}}_{j,k+\frac{1}{2}} - B^{\mathrm{CN}}_{j,k}}{\Delta y}. \end{aligned} \tag{IV.20}$$

3.4 Differences Between the Three Schemes

We provide a quick summary about the differences regarding reconstruction, source terms, drying and order. We remark once more that there are essential differences in the three schemes regarding the reconstruction of the bathymetry and the interface values, and the source term treatment. The HWP and the KP scheme differ in the vicinity of nearly dry cells, as highlighted in the special reconstruction paragraph in Section 3.2. Equations (IV.16) to (IV.20) are the core of the differences between the CN and the KP and HWP schemes, besides the second-order accuracy.

The HWP scheme is capable of drying cells by limiting the time step of nearly dry cells. In the two other schemes, a thin water layer remains. The CN scheme has order 1, while the other two schemes have order 2. A second-order extension of the CN scheme would render the simulation results more comparable, but this is beyond the scope of this paper.

3.5 Time Integration

For all three schemes, an explicit Euler time integrator is used

$$\bar{\mathbf{U}}_{j,k}^{n+1} = \bar{\mathbf{U}}_{j,k}^{n} - \Delta t \frac{\mathbf{F}_{j+\frac{1}{2},k}^{n} - \mathbf{F}_{j-\frac{1}{2},k}^{n}}{\Delta x} - \Delta t \frac{\mathbf{G}_{j,k+\frac{1}{2}}^{n} - \mathbf{G}_{j,k-\frac{1}{2}}^{n}}{\Delta y} + \Delta t\, \mathbf{S}_{j,k}^{n},$$

where quantities denoted by a superscript n depend on the last state $\bar{\mathbf{U}}^n$. The CFL condition restricts the time step $\Delta t = t_{n+1} - t_n$ and is given by

$$\Delta t \leq \mathbf{CFL} \cdot \min\left\{ \frac{\Delta x}{a}, \frac{\Delta y}{b} \right\}, \tag{IV.21}$$

where a and b represent the wave speeds at the interfaces parallel to the x- and y-axis. To ensure stability of a second-order (first-order) finite volume (FV) scheme, the **CFL** constant is not allowed to be greater than 0.25 (0.5) in the two-dimensional case.

3.6 Additional Source Terms

The friction term is included via an additional source term

$$S_f(U) = -\frac{g}{C^2} \begin{bmatrix} 0 \\ u\sqrt{u^2+v^2} \\ v\sqrt{u^2+v^2} \end{bmatrix}. \tag{IV.22}$$

For the Chézy friction C, we use the relationship $C = \frac{1}{n} h^{1/6}$, where n is Manning's roughness coefficient. It is evaluated in a semi-implicit manner described in [39]

$$S_f(\bar{U}_{j,k}^{n+1}) \approx \bar{U}_{j,k}^{n+1} \tilde{S}_f(\bar{U}_{j,k}^{n}), \quad \tilde{S}_f(U) = -\frac{g}{C^2} \begin{bmatrix} 0 \\ 1/h\sqrt{u^2+v^2} \\ 1/h\sqrt{u^2+v^2} \end{bmatrix}, \tag{IV.23}$$

when integrating from time t_n to t_{n+1}.

3.7 Boundary Treatment

For simulating floods, hydrograph data have to be prescribed at the inlets and outlets. Typically, hydrograph data consists of a time series of water levels and discharges, which are then prescribed

as discharge boundary conditions (BCs) and water level BCs. In urban regions, buildings and other structures can completely block water flow. Wall BCs are used to incorporate these fluid-solid interfaces. At the boundary of the computational domain, where the water flow has to be truncated artificially, free outflow BCs are prescribed. Thus, there are four important types of BCs needed in a simulation environment:

- Discharge
- Water level
- Walls
- Free outflow

Usually, hydrograph data are only available on one-dimensional lines, so there is a scalar value for the total discharge Q and/or a water level w for a cross-section. Thus, these lines have to be mapped onto the nearest connected path of grid interfaces. For discharge hydrographs some additional pre-processing is needed to compute the discharges at the rasterized interfaces of a specified cross-sectional hydrograph line. In this case, we assume a velocity vector $\tilde{\mathbf{u}} = (\bar{u}, \bar{v})^T$, that is normal to the cross-section and constant throughout this line. Then the normal interface discharges q_I are given by $q_I = Q\, p_I$ where p_I is the percentage of the total discharge per interface. The percentage p_I depends on the cross-sectional area covered by water at this interface. In the simple case of a hydrograph line parallel to the y-axis

$$p_I = A_I / A := h_I \Delta y / A, \tag{IV.24}$$

where A is the total cross-sectional area of water flow, since

$$Q = \sum_I \bar{u} A_I = \bar{u} \sum_I h_I \Delta y =: \bar{u} A.$$

If a hydrograph contains both discharge and water level, the water level is used to improve the percentages in the discharge hydrograph. The derived interface discharge q_I is then prescribed locally at each hydrograph interface.

In order to incorporate interface boundary data into the finite volume framework, we specify the flux at boundary interfaces, as required by (IV.5). Typically, the flux across an interface is computed by using the reconstructed point values from the left and right sides of the cell interface. Usually, this is done by extending the interior solution to derive conserved variables for the exterior cells of the domain according to the type of the BC.

This approach is termed the ghost cell method, since these cells only contribute via their reconstructed point values to the boundary flux, but are otherwise excluded from the regular solution update. If the ghost cell values are stored as conserved variables, an implementational issue arises if the cell is surrounded by regular cells. In this case, one has to store four different ghost states per cell accounting for each interface, therefore one is obliged to store these values in a special way. In our implementation, boundary data, such as affected interfaces, prescribed BC type and given values, is stored in a special datastructure.

Most BCs have a physical meaning regarding the boundary fluxes, thus in the FVM it seems natural to apply the BCs directly on the fluxes. Ghidaglia and Pascal [125] develop flux BCs to directly compute the normal flux at boundaries. In our simulations, the more flexible flux-based approach is employed. Summarizing, flux BCs are weakly prescribed BCs acting on the fluxes of boundary interfaces.

Care has to be taken to ensure well-posedness for the SWE boundary value problem. This can be understood through a linearized version of the SWE on a flat bottom, see [126, 125]. We are only allowed to set as many physical boundary conditions as there are incoming waves. A wave is

called incoming if the associated eigenvalue of the Jacobian is smaller than zero. If the flow is in a subcritical state, two boundary conditions have to be applied at the upstream boundary and one at the downstream boundary. In this case, we prescribe the discharge in normal and tangential direction at the upstream and the water level at the downstream boundary.

For inflow boundaries with prescribed discharges we follow the approach of Pankratz et al. [69]. For brevity we only present derivations for an eastern interface with outer unit normal $n = (1,0)^T$, thus $\mathbf{\Phi} := \mathbf{F}(\mathbf{U_I})n_x + \mathbf{G}(\mathbf{U_I})n_y = \mathbf{F}(\mathbf{U_I})$. Fluxes for other interfaces can be derived analogously. To guarantee that the mass influx equals the prescribed discharge, the inflow boundary flux should satisfy $\mathbf{F}^{(1)} = q_g$. The approach is based on the solution of a linearized Riemann problem with prescribed interface discharges $(hu)_I = q_g$ and $(hv)_I = (hv)$. Since the tangential discharge is extended continuously and not set to zero, this type of BC is called free-slip. The linearization is chosen around the interior state. In this case, the boundary flux $\mathbf{\Phi}$ is given by

$$\mathbf{\Phi} = \mathbf{F}(\mathbf{U_I}) = \begin{pmatrix} q_g \\ \frac{1}{2}gh_I^2 + \frac{q_g^2}{h_I} \\ q_g v_I \end{pmatrix},$$

where the interface state $\mathbf{U_I} = (h_I, hu_I, hv_I)$ is given by

$$\begin{aligned} h_I &= h + \frac{q_g - (hu)}{u - \sqrt{gh}}, \\ (hu)_I &= q_g, \\ (hv)_I &= (hv). \end{aligned}$$

We use the procedure of Ghidaglia and Pascal [125] to derive the boundary fluxes for specified water levels and walls and we explicitly state the fluxes applied at the boundary. The essential idea is to find a boundary flux $\mathbf{\Phi} := \mathbf{F}(\mathbf{U_I})$ whose interface state $\mathbf{U_I}$ satisfies certain BCs based on numerical and physical grounds.

At a wall boundary, there is no discharge through the boundary interface, so the normal velocity vanishes at the interface, $hu_I = 0$. Thus, the boundary flux is given by

$$\mathbf{\Phi} = \begin{bmatrix} 0 \\ chu + \frac{g}{2}h^2 \\ 0 \end{bmatrix}.$$

This way mass conservation is ensured.

In a similar way, a boundary flux is deduced for a prescribed water depth $h_I = h_g$. In this case the boundary flux is given by [126]

$$\mathbf{\Phi} = \begin{bmatrix} h_g u_I \\ h_g u_I^2 + \frac{g}{2}h_g^2 \\ h_g u_I v \end{bmatrix}, \tag{IV.25}$$

where

$$h_g u_I = hu + \frac{g}{2c}(h^2 - h_g^2).$$

Here, a free-slip condition is imposed on the tangential velocity $v_I = v$ coming either from a physical condition if $u > 0$ or from a numerical condition if $u \leq 0$.

The above BCs are derived for subcritical inflow. If an inflow is supercritical, all waves are incoming and we set $\mathbf{\Phi} = \mathbf{F}(\mathbf{U_I}) = \mathbf{F}(\mathbf{U_g})$, where $\mathbf{U_g}$ is a given state vector. If an outflow is supercritical, we set $\mathbf{\Phi} = \mathbf{F}(\mathbf{U_I}) = \mathbf{F}(\mathbf{U})$, where $\mathbf{U}$ are the conserved variables from the interior cell.

At the outflows where usually no data are available, so we use an extension by continuity. The boundary flux is $\mathbf{\Phi} = \mathbf{F}(\mathbf{U_I}) = \mathbf{F}(\mathbf{U})$, where $\mathbf{U_I}$ is the zero-order extension of the interior state $\mathbf{U}$. This yields the same BC as if the outflow was supercritical. These BCs are also called open BCs, since these boundaries do not generate any incoming waves.

4 Validation

In this section, we present multiple analytical, laboratory and real-world validation cases. For all three numerical schemes, the simulated results are compared against analytic or measured data. In the following plots, we abbreviate the analytical solution as AN and measured data as ME.

4.1 Analytical Test Cases

First, we discuss the two analytical cases for which no friction is assumed.

Parabolic Basin

The parabolic basin is a classical test case for validation. There is an analytical solution of Thacker [83]. It describes time-dependent oscillations of a planar water surface in a parabolic basin.

It is widely used for comparing different numerical schemes [112, 85, 86, 87]. Recently, Sampson et al. [87] extended the solution of Thacker to support bed friction. However, their solution is limited to one dimension. In this two-dimensional case, we use the same setup as Holdahl et al. [88], where the bathymetry is given by:

$$B(x,y) = D_0 \left(\frac{x^2 + y^2}{L^2} - 1 \right), \tag{IV.26}$$

where $L = 2500$ m, $D_0 = 1$ m. The water surface elevation and the velocities are given by:

$$w(x,y) = \frac{2AD_0}{L^2} \left(x \cos \omega t + y \sin \omega t + LB_0 \right), \tag{IV.27}$$

$$u(x,y) = -A\omega \sin \omega t, \tag{IV.28}$$

$$v(x,y) = A\omega \cos \omega t, \tag{IV.29}$$

$$\omega = \sqrt{\frac{2D_0}{L^2}}, \tag{IV.30}$$

where we choose $A = L/2, B_0 = -A/2L$, and the gravitational constant $g = 1$ m/s^2 for our simulations. Then we have:

$$w(x,y) = \frac{D_0}{L} \left(x \cos \omega t + y \sin \omega t + LB_0 \right) \tag{IV.31}$$

and $\omega^2 = 3.2 \times 10^{-6}$ s^{-2}.

We show simulation results for two resolutions and two snapshots in time for a one-dimensional slice in x-dimension in the middle of the computational domain at times $t = 2788$ s (Figure IV.2a, IV.3a) and $t = 5566$ s (Figure IV.2b, IV.3b). We plot water levels $\bar{w}_{j,k}$ and velocity magnitudes $|\bar{u}|$ at cell centers, which are the average values.

Both the KP and the HWP schemes capture well the water levels of the analytical solution. However, along the wet/dry boundaries velocity errors accumulate. This behaviour can be found

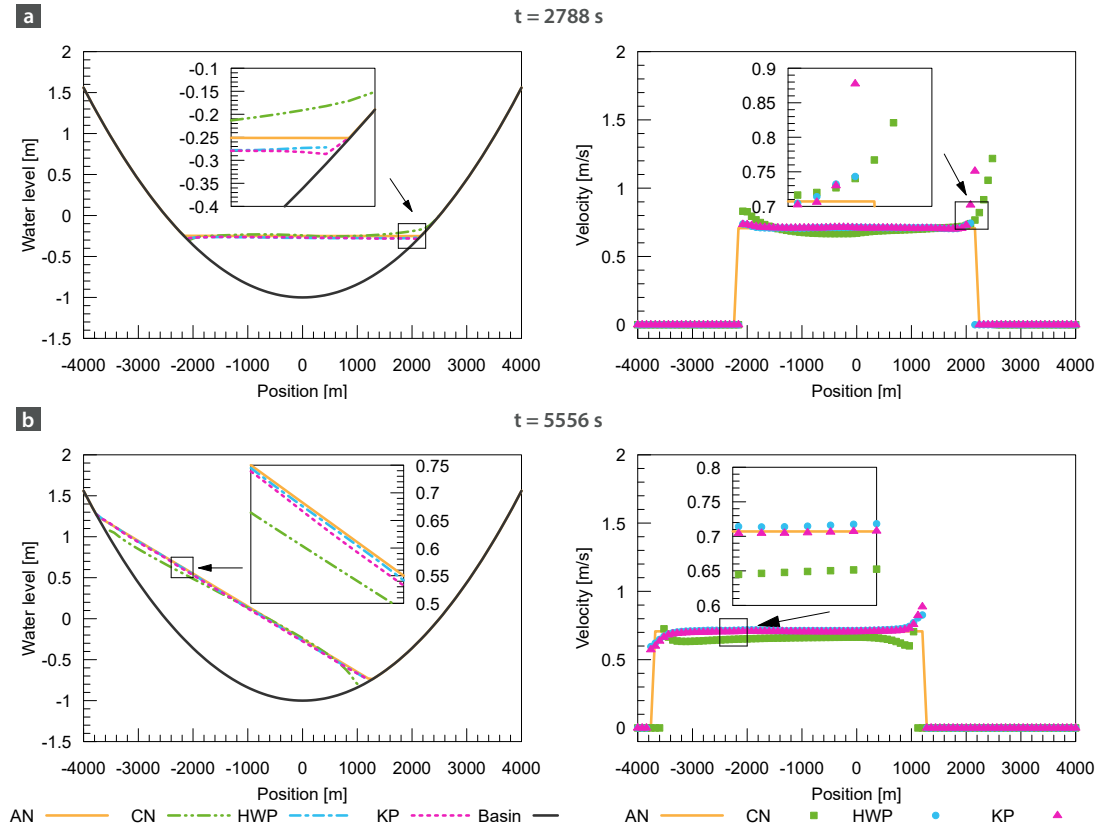


Figure IV.2: Water level and velocity profiles of the parabolic basin with a high resolution of 80 m at a) 2788 s and b) 5556 s. a) All three schemes accumulate errors along the wet/dry boundary. b) The CN scheme performs rather poorly at this later stage due to the diffusive nature of first-order schemes, as can be seen by the overall loss of kinetic energy in the velocity.

in many schemes [88, 18], and is difficult to avoid. The superiority of the second-order schemes KP and HWP at low resolutions (80 m) can be clearly seen in Figure IV.2. They capture both the water levels and the velocities accurately away from the wet/dry boundary. The CN scheme performs rather poorly for the low resolution at the later stage (Figure IV.2b), due to the diffusive nature of first-order schemes. However, at a finer resolution (20 m), the CN scheme improves significantly (Figure IV.3. At small cell sizes, the HWP scheme produces flickering water levels and velocities as can be seen in Figure IV.3b).

Parabolic Bump

This section is devoted to multiple steady state test cases with a parabolic bump to test different flow regimes and transitions between them in a quasi one-dimensional way. The scenario is set up analogously to Audusse et al. [15, 127] and is originally from Goutal et al. [128]. The analytical solutions for the steady states can be derived using the Bernoulli relation, see [129, 127]. The bathymetry is given by:

$$B(x,y) = \begin{cases} 0.2 - 0.05(x-10)^2 & \text{if } 8 < x < 12, \\ 0 & \text{else,} \end{cases}$$

for a domain of length L = 20.2 m and a width of 4 metres (Figure IV.4). Since in this setup all cells are always flooded, the HWP and the CN schemes effectively fall back to their predecessors, the KP and the HR schemes of Audusse, respectively. The cell size is set to 0.2 m resulting in 101 cells

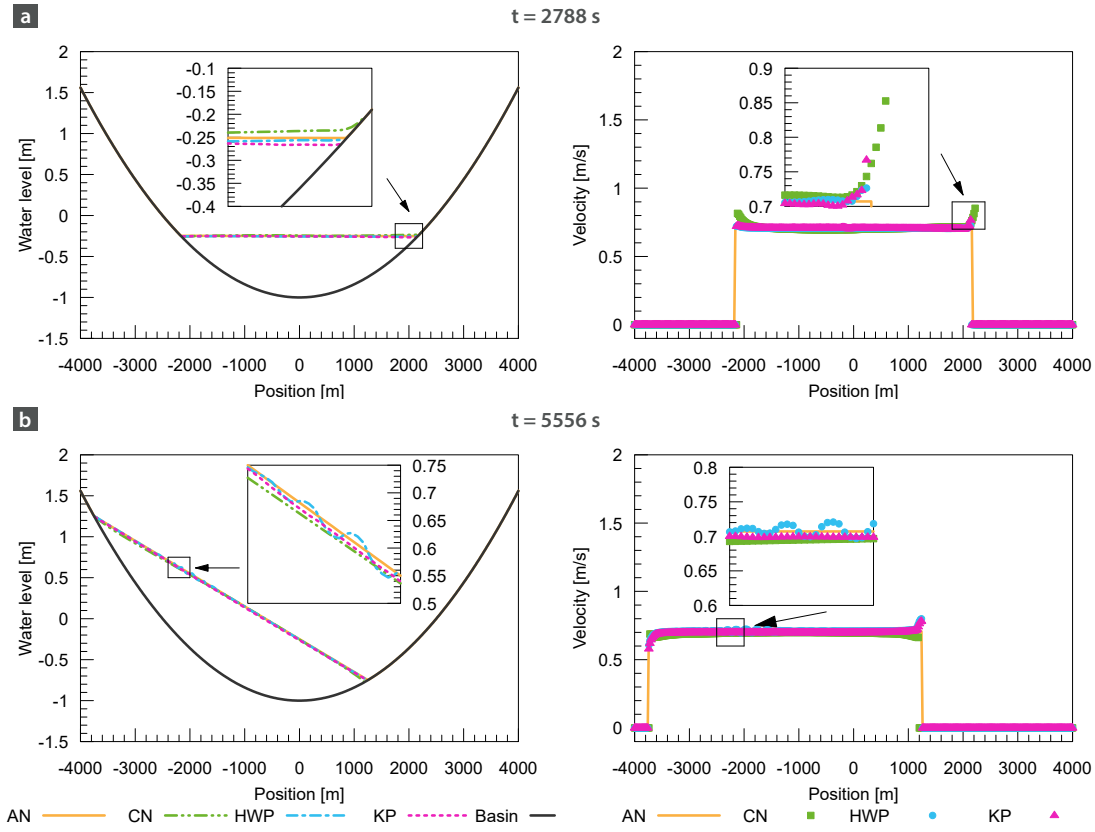


Figure IV.3: Water level and velocity profiles of the parabolic basin with a high resolution of 20 m at a) 2788 s and b) 5556 s. b) The zoom shows that the HWP scheme produces flickering water levels and velocities. The CN scheme improves notably for the smaller cell size.

in x-direction. We compare three flow regimes, subcritical flow, transcritical flow without shock, and transcritical flow with shock.

Figure IV.4: A schematic view of the parabolic bump shows its geometrical parameters.

In the case of subcritical flow, one boundary condition has to be specified at the inflow $x = -L/2$ and one at the outflow $x = L/2$. The water depth is given by:

$$h^3 + \left(B - \frac{q}{2gh_E} - h_E\right) h^2 + \frac{q^2}{2g} = 0, \quad \forall (x,y) \in [-L/2, L/2] \times [0,2],$$

where $h_E = h(L/2, 0) = w_0 = 2$ m, the water depth at the east outflow boundary. The discharge in this case is specified as $q_W = 4.42$ m^2/s at the inflow boundary. Figure IV.5a shows the results for this case.

In the case of a transcritical flow without shock, there is a transition from a subcritical flow regime to a supercritical flow regime. The water depth has to fulfill:

$$h^3 + \left(B - \frac{q}{2gh_c} - h_c - B_M\right) h^2 + \frac{q^2}{2g} = 0, \quad \forall (x,y) \in [-L/2, L/2] \times [0,2],$$

where $B_M = \max_{x \in [0,L]} B$ is the maximum bathymetry elevation and h_c is the corresponding water depth. The discharge is specified as $q_W = 1.53$ m^2/s. As long as the flow is subcritical we set w_E

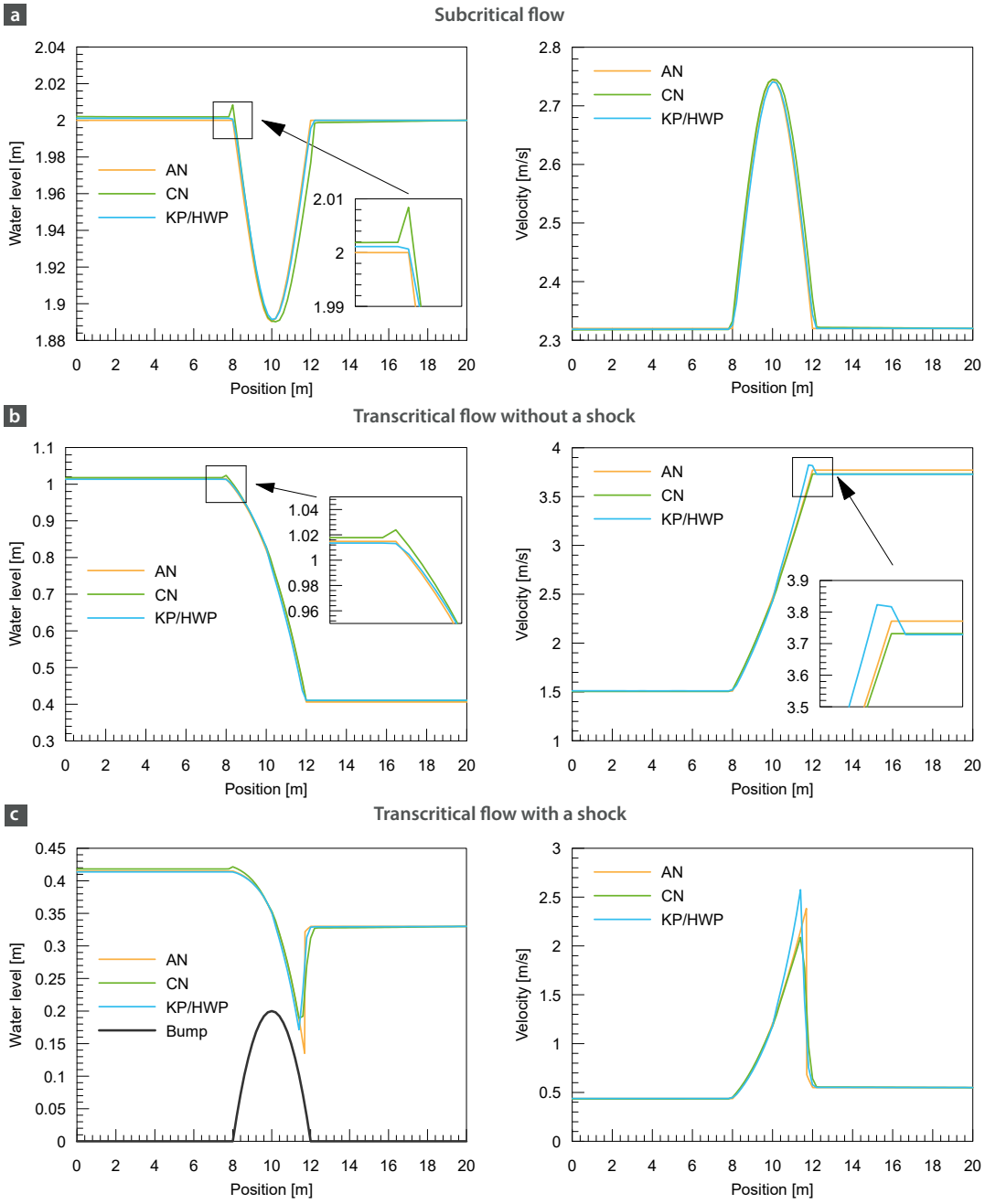


Figure IV.5: Water level and velocity profiles of the parabolic bump. a) Subcritical flow case. b) Transcritical flow. c) Transcritical flow with a jump case. All schemes can detect the correct jump location. The CN scheme overestimates the water level at the discharge inflow. The CN scheme produces a small water level overshoot at the beginning of the bump.

to the initial water level $w_0 = h_0 + B = 0.66$ m. The simulation is initialized with a hydrostatic equilibrium $q_0 = 0$ m^2/s. Figure IV.5b shows the results for this case.

In the case of a transcritical flow with shock, commonly termed hydraulic jump in hydrodynamics, there is an abrupt rise in water levels, which occurs at the transition from a supercritical flow regime to a subcritical flow regime. We have to divide the domain into two domains where the solution is again regular. The water depths at the left and at the right of the jump are given by $h_1 = h(x_S-)$ and $h_2 = h(x_S+)$. The jump location x_S can then be determined with the help of the

Rankine-Hugoniot condition [78]:

$$q^2 \left(\frac{1}{h_1} - \frac{1}{h_2} \right) + \frac{g}{2} \left(h_1^2 - h_2^2 \right) = 0.$$

The water depth is given by the two equations:

$$h^3 + \left(B - \frac{q}{2gh_c} - h_c - B_M \right) h^2 + \frac{q^2}{2g} = 0, \quad \forall x \in [0, x_S],$$
$$h^3 + \left(B - \frac{q}{2gh_E} - h_E \right) h^2 + \frac{q^2}{2g} = 0, \quad \forall x \in [x_S, L],$$

where $B_M = \max_{x \in [0,L]} B$. These equations were solved numerically and plotted alongside the solutions of the schemes in Figure IV.5c. Boundary and initial conditions are $w_E = h_E = 0.33$m at the outflow, $q = q_W = 0.18$ m^2/s at the inflow and the initial state is $q_0 = 0$ m^2/s and $w_0 = 0.33$m, respectively. The schemes are able to detect the correct jump location and can resolve the regime change. The water level at the inflow is slightly off in the case of the first-order CN scheme, as already reported for the original HR scheme [15].

4.2 Laboratory Test Cases

In this section, we discuss three laboratory cases (U-shaped flume, sine-generated flume, dam break over a triangular hump) and compare the simulation results against data measured in controlled environments.

U-Shaped Flume

In order to test the model's capability to simulate the flow in meandering channels, a verification was conducted first for a U-shaped 180° channel. This test case is based on the laboratory experiment of [130]. The layout of the physical model is shown in Figure IV.6. Experimental data are taken from NCCHE [131], who in turn digitized the tables of de Vriend [130]. Measurements were made at several cross sections (C3-C24) along the flume.

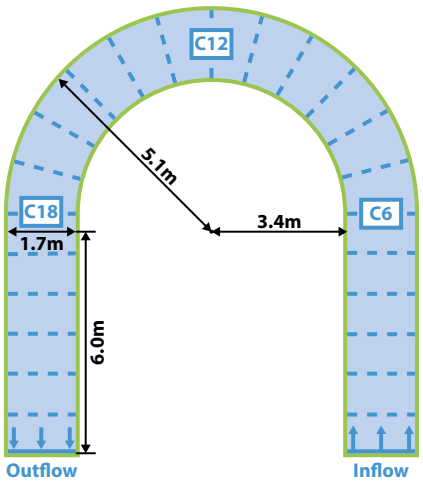


Figure IV.6: A schematic view of the U-shaped flume with its geometrical parameters and the selected cross sections C6, C12, C18.

A discharge of 0.18 m^3/s is prescribed at the inflow boundary and a constant water elevation of 0.1875 m at the outflow. Wall boundary conditions are applied along the rasterized flume

walls. Manning's roughness is uniformly set to 0.0001 $m^{1/3}/s$. The simulations are stopped and visualized after 300 s, where the CN scheme reaches a steady state (Figure IV.7).

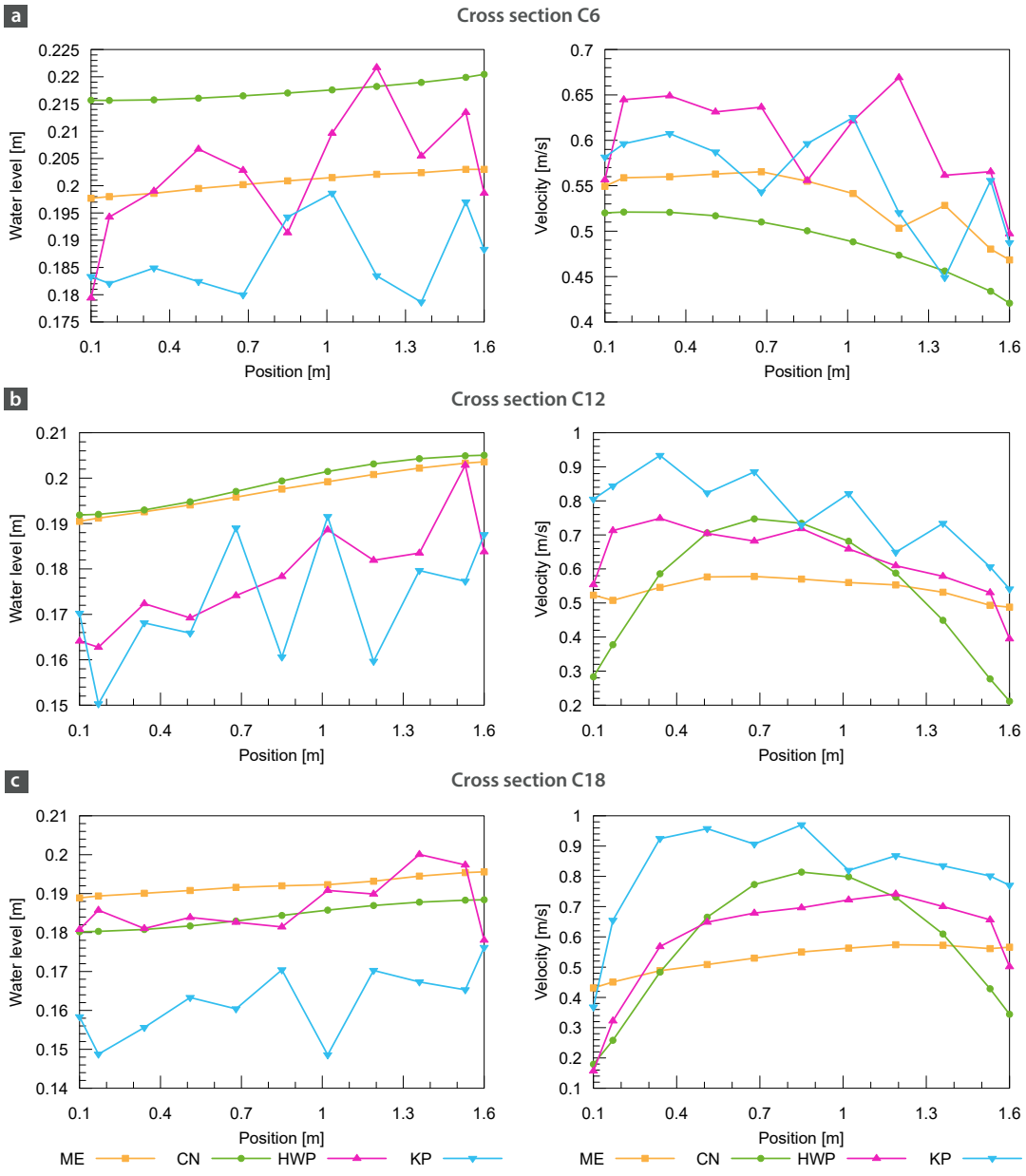


Figure IV.7: Water level and velocity profiles for the U-shaped flume at different cross sections. The x-axis corresponds to the cross-sectional position from the inner bank to the outer bank. The second-order schemes KP and HWP are oscillating and are not reaching a steady state, in contrast to the CN scheme.

In curved channels, the water surface elevation at the outer bank side is higher than the one along the inner bank due to the centrifugal force. The second-order schemes KP and HWP do not reach a steady state due to their oscillating nature. A comparison of computed and measured free surface elevations and depth-averaged velocities for the selected cross sections C6, C12 and C18 are shown in Figure IV.7a, IV.7b and IV.7c, respectively. There are some sources of discrepancies between the measured and the simulated data. First, we use a regular grid, where the cells are not aligned to follow the curvature of the flume and its walls. Second, in curved channels, there are distinctive characteristics that are completely three dimensional such as secondary currents, which cannot be covered by a depth-averaged model.

Sine-Generated Flume

Another idealized form of meandering channels are sine-generated flumes. This experiment was conducted at the Laboratório Nacional de Engenharia Civil (LNEC), Lisbon. The data are taken from [132]. For the 30° channel there is a conserved discharge of $Q = 0.0021$ m^3/s. The discharge is set as an inflow BC at $x = 0.1$ m and a constant water level of 0.3 m is maintained at the outflow at $x = 6.2$ m. The flume is 40 cm wide, and the river line of one meander is 2.964 m long (Figure IV.8). The simulation domain size is chosen to 6.4 m $\times$ 1.2 m with a uniform cell length of 0.0025 m. Manning's roughness is uniformly set to 0.0044 m$^{1/3}$/s. We compare the results after 100 s for the cross sections 5_1, 5_2, 5_3 in Figure IV.9.

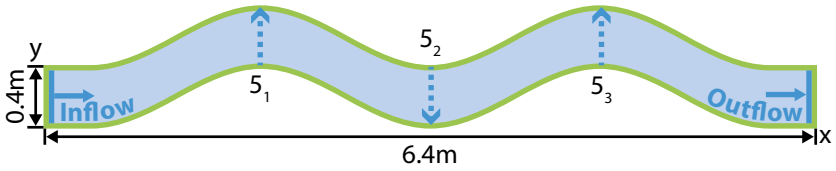


Figure IV.8: A schematic view of the sine-generated flume and selected cross sections: 5_1, 5_2, 5_3.

The discrepancies found for the U-shaped flume also hold in this case. Similarly, the HWP and KP schemes do not converge to a steady state as there are unphysical oscillations. Setting the minmod parameter to a lower value has a positive effect, however, even a minmod parameter of 1 does not remove the oscillations completely. All schemes capture the higher velocities at the inner banks, but the second-order schemes resolve the boundary layer more accurately (right plots in Figure IV.9). The inner banks correspond to the origins in the plots in Figure IV.9. As in the previous cases, the first-order scheme overestimates the water levels at the discharge inflow, causing the water levels to decrease along the flume, see cross sections 5_1 and 5_3 in Figure IV.9a, Figure IV.9c.

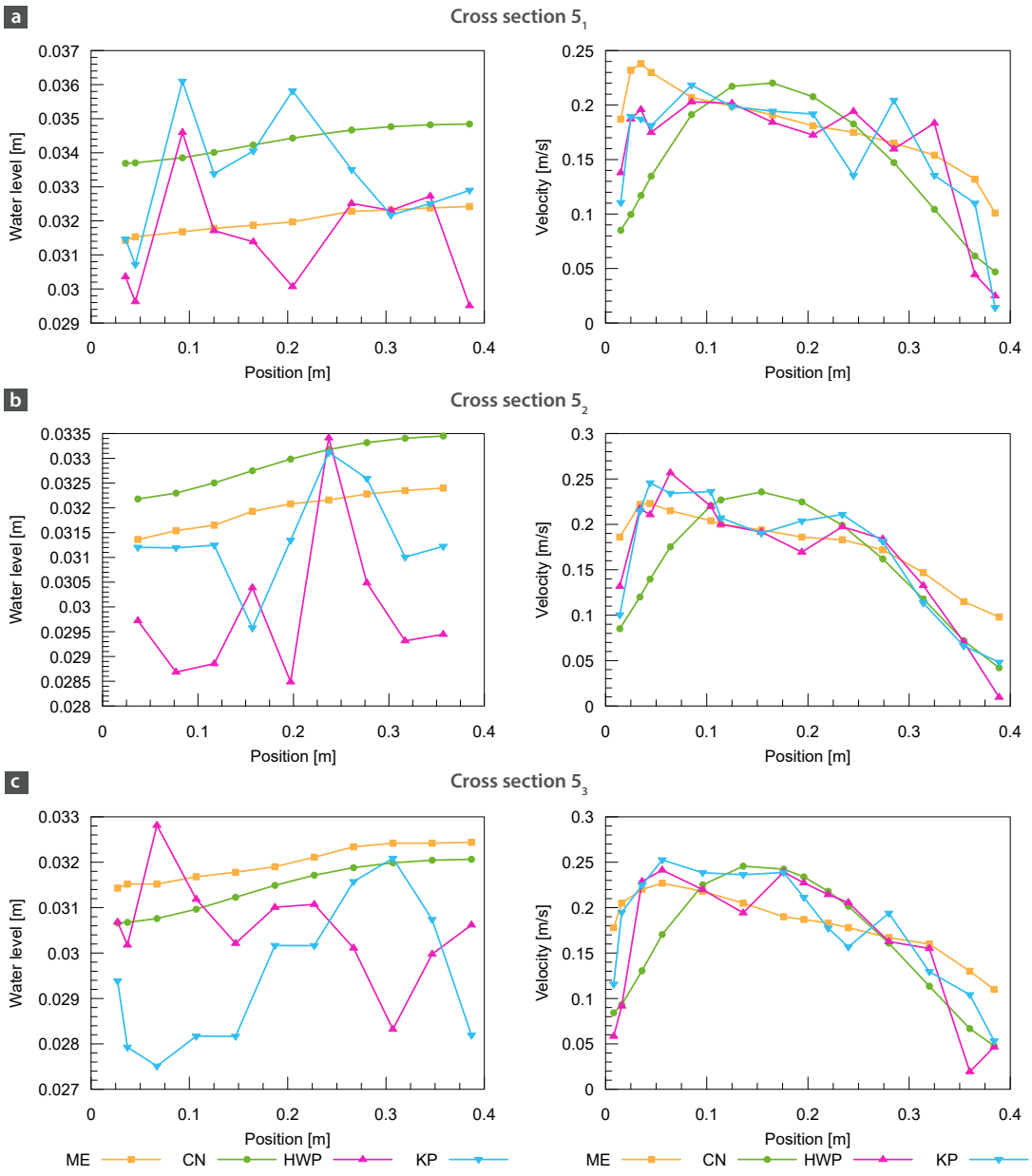


Figure IV.9: Water level and velocity profiles for the sine-generated flume at cross sections $5_1, 5_2, 5_3$. The x-axis corresponds to the cross-sectional position from the inner bank to the outer bank as marked in the schematic view. The second-order schemes KP and HWP resolve the boundary layer better than the CN scheme.

Dam-Break over a Triangular Hump

A dam-break wave generated in a laboratory experiment conducted at the Hydraulic Research Laboratory, Châtelet, is simulated. The schematic layout of the experiment is shown in Figure IV.10. The experimental data used for validation are from Liang [86]. The water of a reservoir is suddenly unleashed by an instantaneous removal of the gate. This generates a dam-break wave, which then hits a triangular hump. The water levels are collected at seven gauge points in the downstream region.

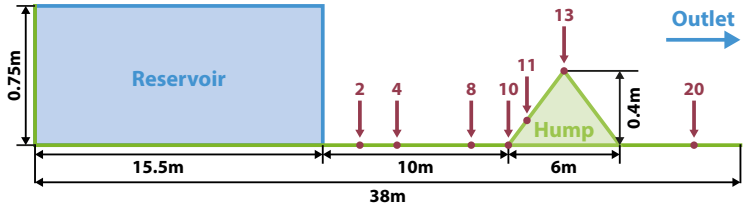


Figure IV.10: A schematic view of the triangular hump.

A uniform Manning coefficient is used, i. e., $n = 0.0125\ \mathrm{m}^{1/3}/\mathrm{s}$ [86]. The lateral boundaries are slippy solid walls, and the down-stream boundary condition is set to free outflow. The cell length is 0.01 m. The left plots of Figure IV.11 show the water level time series at four gauging points: G4, G10, G13 and G20. Overall, the three schemes produce very similar results, all of them capturing the measured data well, in particular the wave arrival times at the gauging locations. However, the water depths are overestimated at the downstream side of the obstacle [133, 134, 135]. The simulation results are in good agreement with the results of Singh et al. [135].

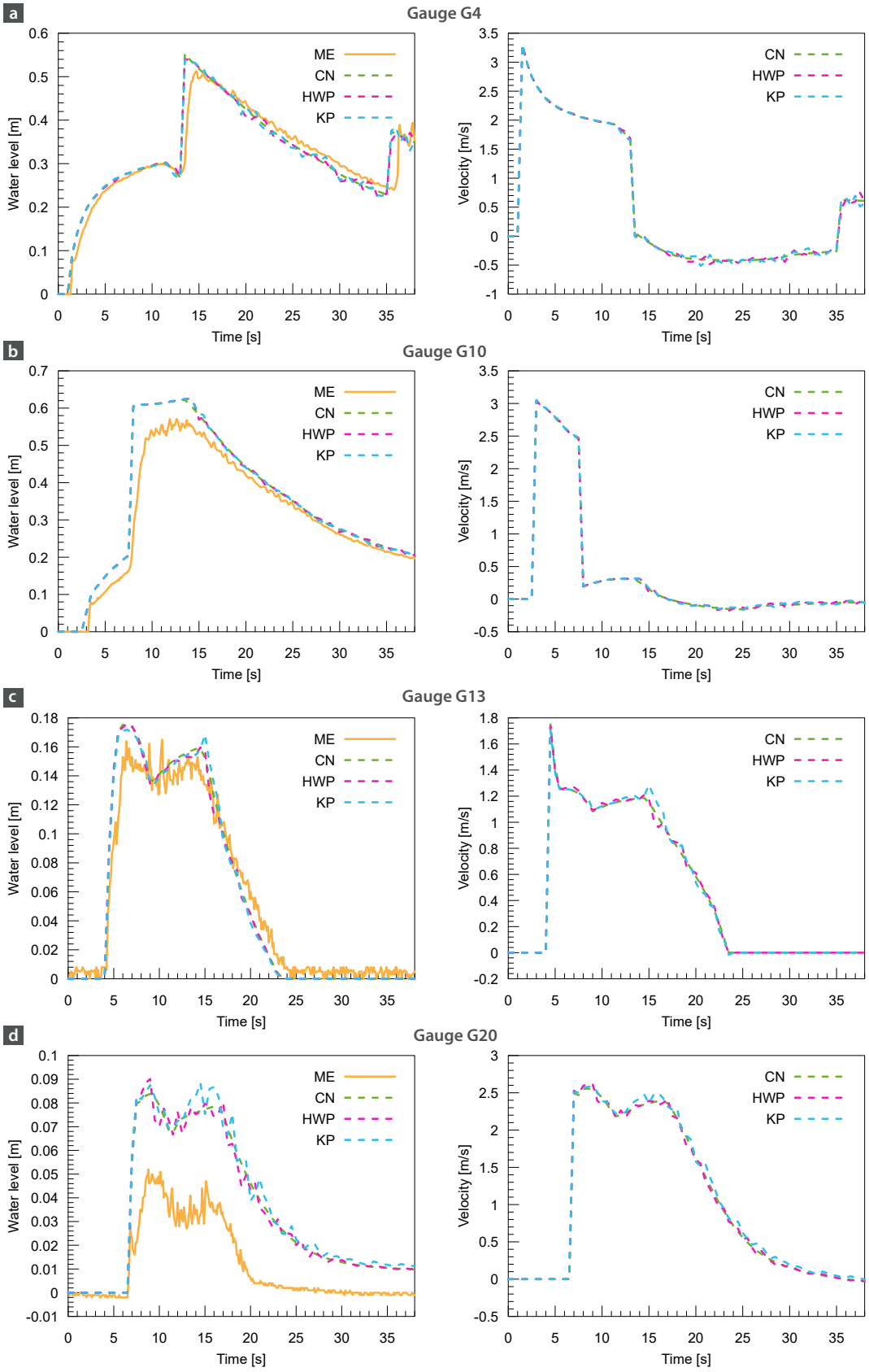


Figure IV.11: Water level (left) and velocity (right) time series for the triangular hump at gauging locations a) G4, b) G10, c) G13, and d) G20. All three schemes capture the data well, except for gauging point G20.

4.3 Real-World Test Cases

All real-world simulations were performed on GPUs. The implementation of our simulation system on the GPU uses the CUDA programming model on top of a NVIDIA Kepler architecture [114].

Malpasset

The Malpasset dam break event of 1959 was largely investigated in the past years [39, 108, 135]. The dam failed explosively and gave rise to a 40 meters high flooding wave. Our validation is based on the dataset available from the TELEMAC samples [105]. The original dataset consists of 104 000 unstructured points. Our simulation grid contains 1149×612 cells, each cell of the size of 15×15 m^2. We use a uniform Manning coefficient $n = 0.033$ m$^{1/3}$/s corresponding to weedy, stony earth channels and floodplains with pasture and farmland. We simulate the first 4000 seconds of the dam break, and compare the results with laboratory experiments on a 1:400 scale model [106, 107]. In these experiments, researchers have recorded wave front arrival times [106] and maximum water elevations [107] at 14 gauge locations (S1-S14). Our verification uses only locations S6-S14, since no data are available for the other gauge locations. The surveyed arrival times are determined based on the shut down of the voltage transformers at the historical flood event (Figure IV.12). The exact arrival times of the flood wave front are unknown, and the measurements are affected by uncertainties, e. g., the rupture time of the dam.

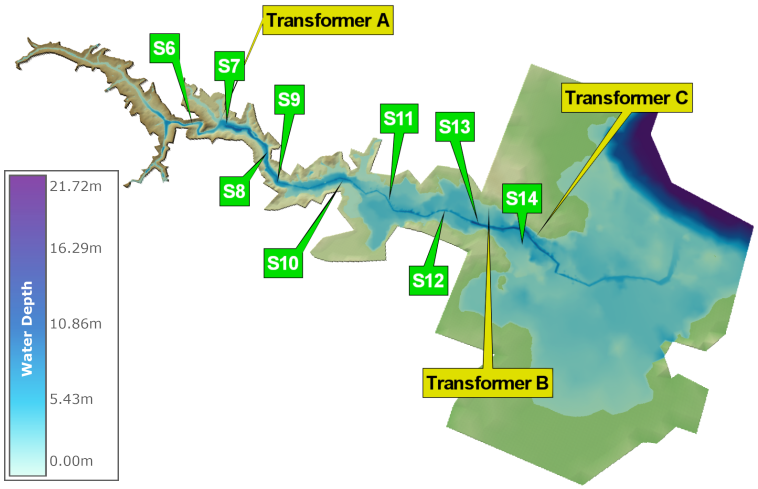


Figure IV.12: Malpasset Dam Break, France. Water extent 4000 seconds after the dam break. Labels show the nine gauge locations (S6-S14) of the laboratory experiments and the three voltage transformers (A-C) in the real world.

Figures IV.13 show our simulation results compared to the physical data acquired by the laboratory model. Overall, there is good agreement with the measurements. Small discrepancies between the scale model and the numerical results were also reported in other studies [39, 108, 119], and our results are consistent with these. In case of the CN scheme, the wave arrivals at the 9 gauge locations are more delayed compared to the KP and HWP schemes (Figure IV.13b). The reason for this is the first order accuracy of the CN scheme which dissipates the wave energy faster than the second order schemes. On the other hand, the CN scheme provides better results for the voltage transformer wave arrival times (Figure IV.13c). The KP and HWP schemes produce quite similar results regarding arrival times and maximum water levels.

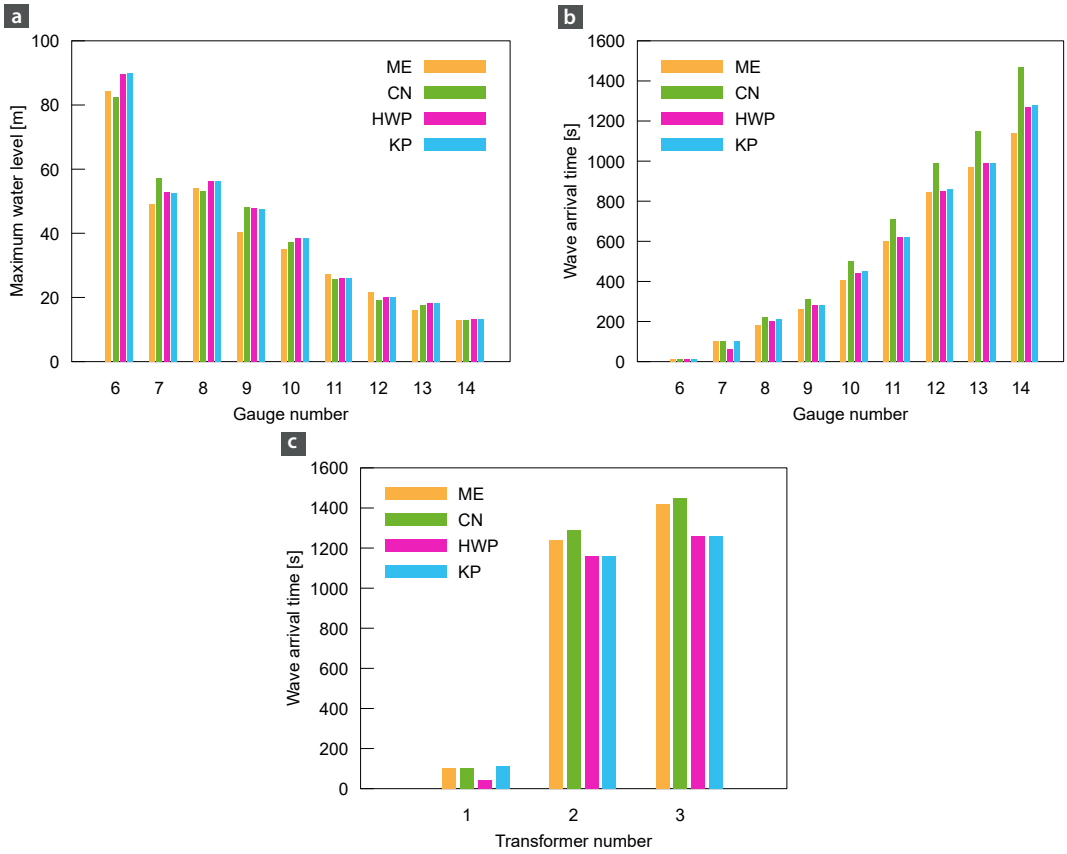


Figure IV.13: Malpasset Dam Break, France. a) Maximum water elevations at the gauge locations (S6-S14). b) Wave arrival times at the gauge locations (S6-S14). c) Wave front arrival times at the three voltage transformers (1-3).

Lobau

This case study involves the Lobau area, which is the alluvial backwater and floodplain of the Donau-Auen National Park in Austria. It extends on the left bank of the river Danube from river kilometer (rkm) 1918 to rkm 1908 downstream of the city of Vienna. If the water level in the Danube rises, water flows from the river into the floodplain, causing regular flooding events. The size of the area is 1474 ha and it consists of floodplain forests and surface water bodies. Even though this is a rural area and does not contain any buildings, it is challenging to perform simulations for this region, since it has a very complex bathymetry (lots of small channels and steep slopes). We reconstructed the flooding of January 2011, and simulated the first for 4 days. The Lobau can only be flooded through a small weir, the Schönauer Schlitz. The inflow and gauging locations are shown in Figure IV.14 along with initial state, the simulated water extent at peak discharge and at the final time.

Water level and discharge values for the Schönauer Schlitz are visualized in Figure IV.15a, which are used to prescribe the inflow conditions from the Danube into the Lobau. The size of the simulation domain is approximately 7.5×5 km^2 and the cell size was set to 4×4 m^2. Water level time series at three gauging locations are compared against the measured values in Figure IV.15b-d, where day zero on the x-axis corresponds to 13 January 2011, 12 am.

We can see that all three schemes have difficulties predicting the correct arrival times at the two inner gauging locations PD.LP16 and PD.LP18. A non-uniform distribution of the Manning roughness coefficient is used based on the land use. These roughness values are not calibrated and the exact initial state before the flooding is unknown, which may explain the deviations from the observations. The water levels of the HWP and KP scheme are very similar, this might be due

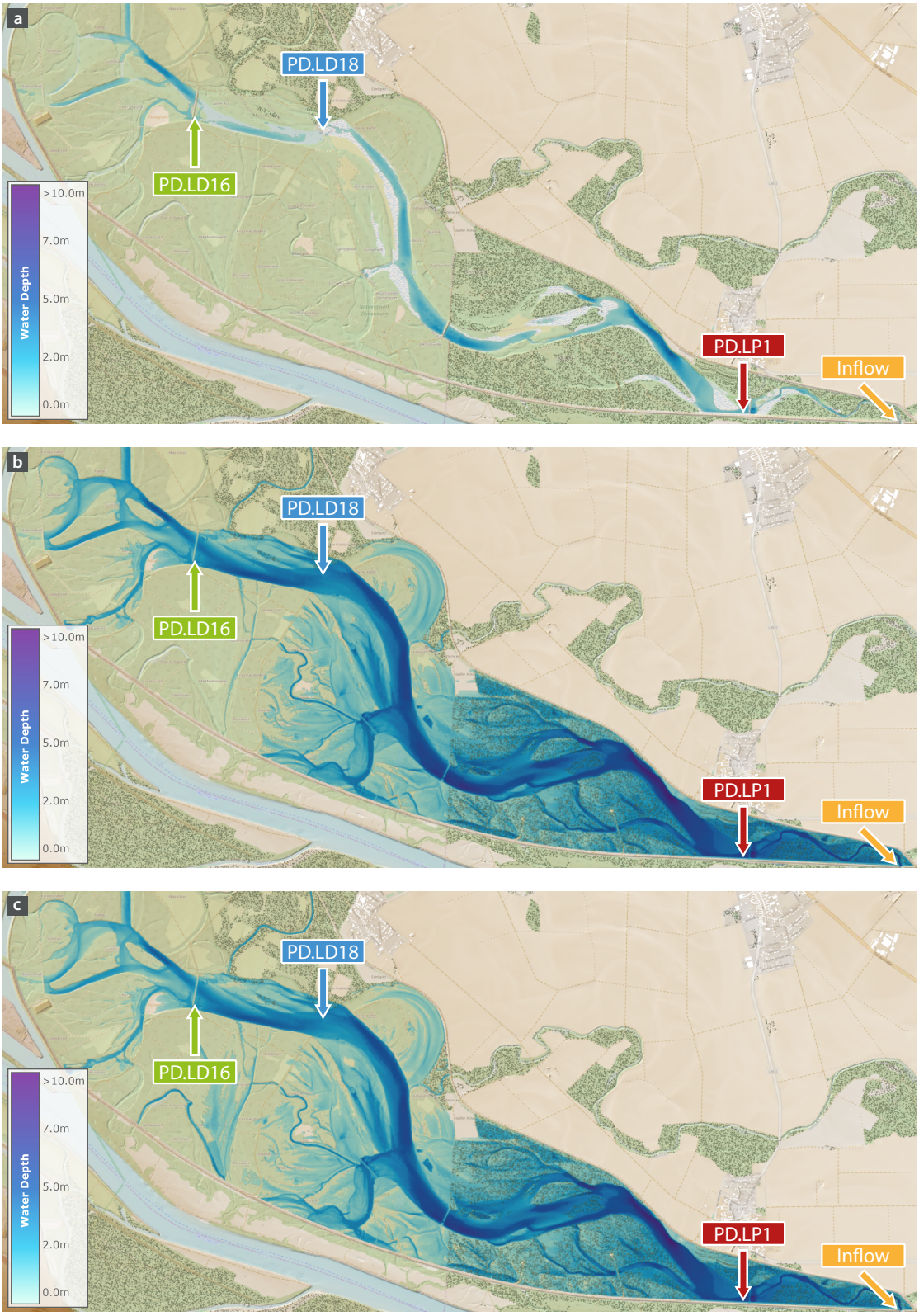


Figure IV.14: Lobau, Donau-Auen National Park, Austria. The colored labels show the location of the inflow at the Schönauer Schlitz (lower right) and the three gauging locations PD.LP1, PD.LP16 and PD.LP18. a) Initial state. b) Water extent at the peak discharge after 2 days simulated by the CN scheme. c) Simulated water extent after 4 days simulated by the CN scheme.

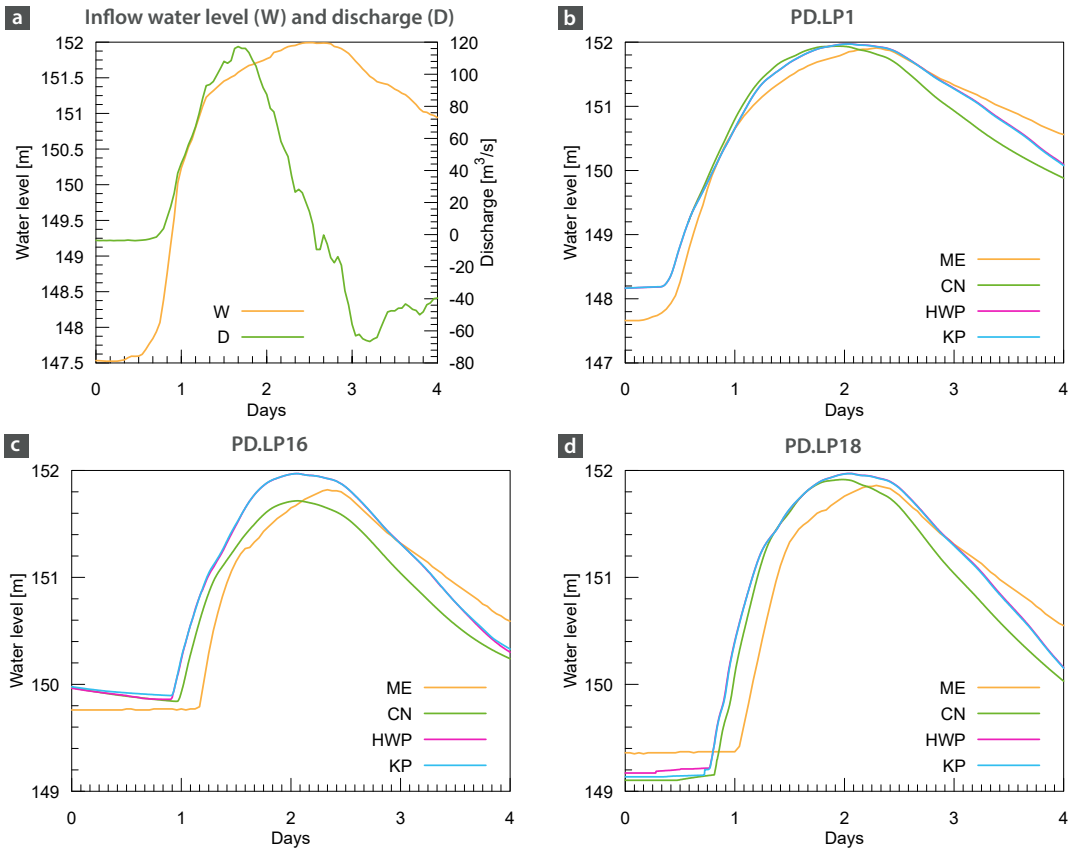


Figure IV.15: Lobau, Donau-Auen National Park, Austria. a) Prescribed inflow boundary conditions data at the Schönauer Schlitz. b-d) Water level time series for the Lobau at different gauging locations.

to the fact that the desingularization originally proposed for the KP scheme has been omitted. Both of them produce a higher maximum water level at PD.LP16 and PD.LP18 and the flood wave is faster than that of the CN scheme (Figure IV.15c,d).

Wachau

The Wachau valley is located in Lower Austria. It was carved out by the Danube over thousands of years. It features a riverine landscape with a settled flood plain bounded by steep slopes. This case study aims to reproduce the 100-year Danube river flood of 2013 [136]. The focus lies on the correct prediction of water levels along the river. This case also emphasizes the proper setting of in- and outflow conditions at the upstream and downstream boundaries of the simulated reach of the Danube. Numerical difficulties may arise due to the complex topography. Measured data are available for four gauging locations along the river, namely near Stein-Krems (rkm 2002,7), Loiben (rkm 2005,99), Dürnstein (rkm 2009,15) and Kienstock (rkm 2015,21) (Figure IV.16).

The dataset of Kienstock and Stein-Krems is used to prescribe the upstream and the downstream boundary conditions, respectively. At the upstream boundary we use a discharge hydrograph with water level information and at the downstream we use only water levels (Figure IV.17a). Roughness values were set according to land-use data. We simulated 14 days, starting on 30 May 2013 at 5 pm with a prefilled moving-water steady state. Simulations were performed using two different mesh sizes, a coarse grid with a cell length of 12 m and a finer grid with a cell length of 3 m. The simulation results for the coarse grid are presented in Figure IV.17b,c. The CN scheme overestimates the water levels by about 1m. The second-order schemes capture the water levels more accurately with the HWP performing better than the KP scheme. On the finer grid, the

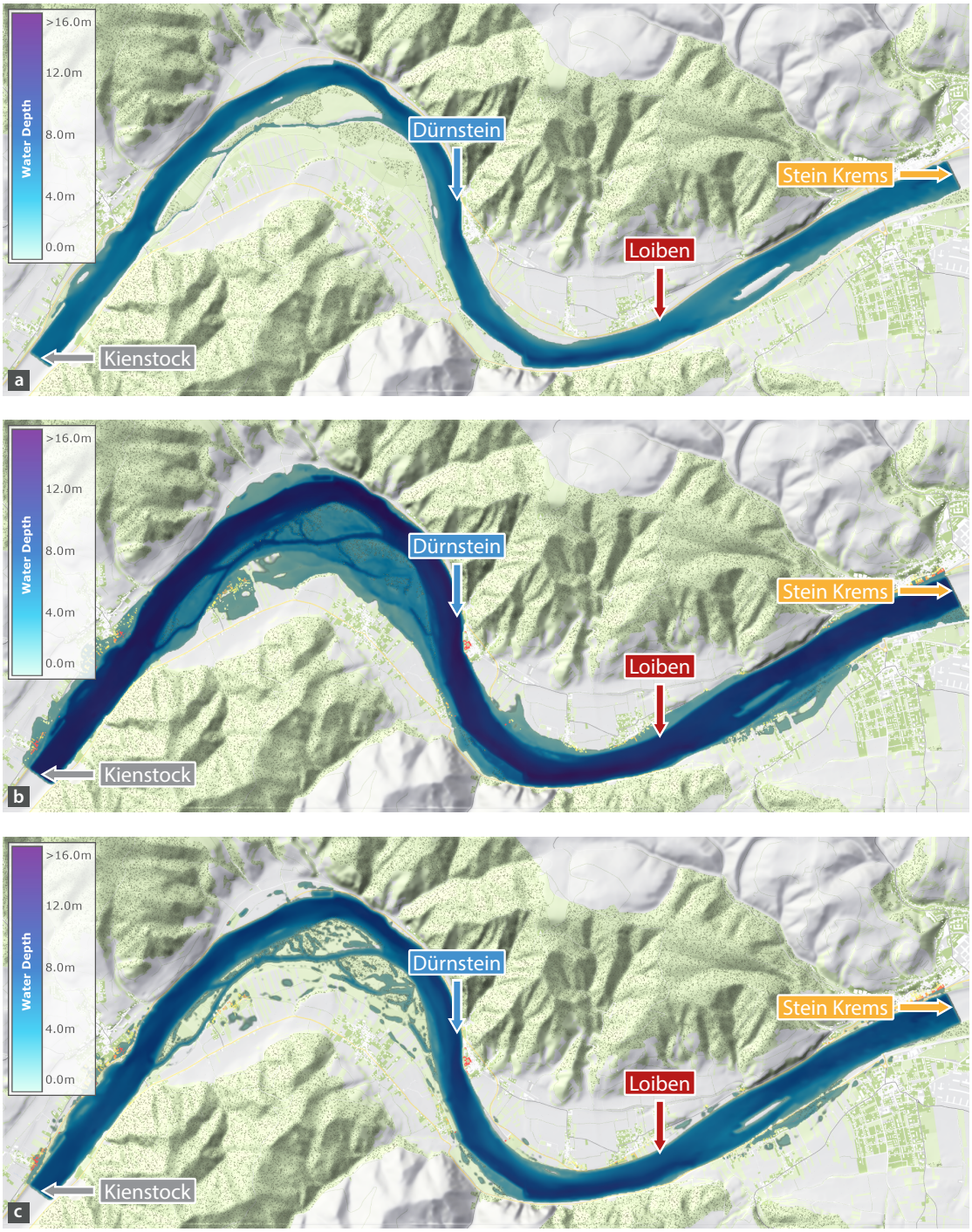


Figure IV.16: Wachau, Austria. The colored labels show the upstream (Kienstock/grey) and the downstream (Stein Krems/orange) boundary locations and the 2 gauging locations (Dürnstein/blue, Loiben/red). a) Initial state. b) Water extent at the peak discharge after 6.5 days simulated by the CN scheme. c) Water extent after 14 days simulated by the CN scheme.

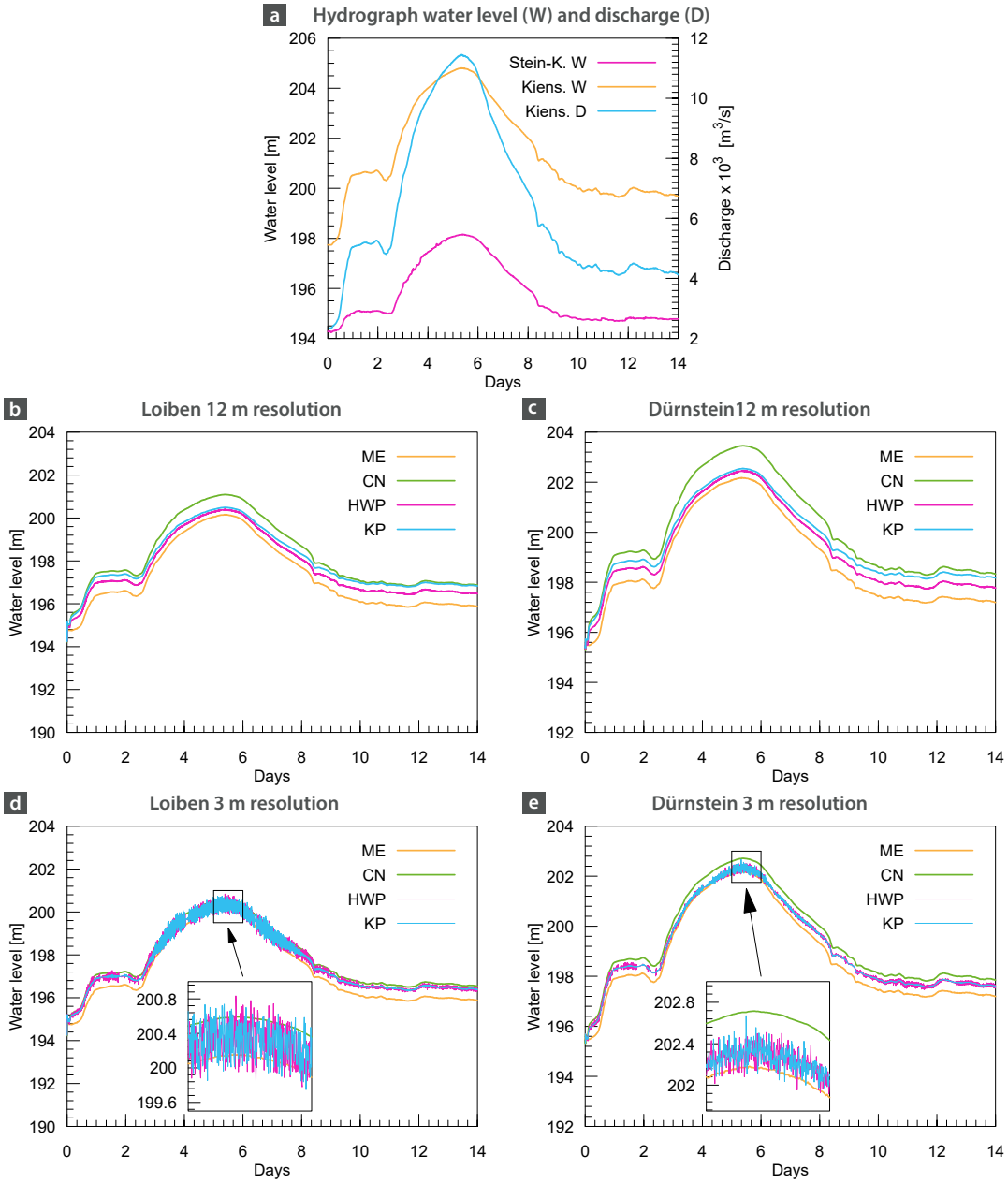


Figure IV.17: Wachau, Austria. a) Water level (W) and discharge (D) data for the upstream (Kienstock) and downstream (Stein Krems) boundary conditions. b,c) Measured and simulated water level time series at the gauging locations using 12×12 m^2 cells. The second-order schemes KP and HWP match the measured water level time series quite well, although all of the schemes overestimate the water levels, particularly the CN scheme. d,e) Measured and simulated water level time series at the gauging locations using 3×3 m^2 cells. The second-order schemes KP and HWP produce spurious oscillations at these small cell sizes. The CN scheme benefits from the small cell size and simulates water levels close to the measurements.

second-order schemes KP and HWP produce spurious oscillations, which slow down the simulations dramatically. In this case, we applied a velocity limitation to 15 m/s. The results are shown in Figure IV.17d,e. Of course, the accuracy of the schemes need to be considered in the context of the accuracy of the water level and discharge measurements used as boundary conditions [137].

5 Conclusion and Future Work

In this paper, we compare three numerical schemes for the shallow water equationss by extensive simulations for various study cases. The three schemes (KP, HWP, and CN) are well-balanced state-of-the-art schemes, chosen because of their suitability for fast flood simulations on GPUs. The comparison focuses on the prediction of flood wave arrival times and water level series, since they are especially relevant for decision makers. The simulation results revealed that each of the schemes has its strengths and weaknesses. For large cell sizes the validation case studies showed that the second-order schemes, the KP and HWP schemes, perform better in real-world scenarios, as the water levels are closer to the measurements. For smaller cell sizes, which capture the topography more accurately, non-physical oscillations occur in both of them. Oscillations hinder the schemes in reaching steady states in the U-shaped and sine-generated flume case studies, where high mesh resolutions are needed to resolve the curved wall boundaries. The oscillatory behaviour in the real-world cases are a real drawback of the second-order schemes. In contrast, the CN scheme provides smooth solutions without oscillations even in the case of a high mesh resolution. But the first-order CN scheme can not predict the water levels to the same accuracy as the second-order schemes. This is particularly true for large cell sizes. The CN scheme overestimates water levels in river scenarios, especially near the upstream discharge boundaries. Overall, there is good agreement between the simulated and measured values, suggesting that the simulator can be used for flood management purposes.

Future research should bring together the advantages of the CN scheme and the second-order schemes. Specifically, the stability, robustness, and speed of the CN scheme should be maintained while gaining second-order accuracy. A possible remedy in the case of curved walls might be so-called cut-cells, which allow the inclusion of curved and non-aligned lines into regular grids, or, improved wall boundary conditions. Furthermore, hydraulic structures, such as dikes, embankments, weirs, etc. should be integrated into the simulation framework. Another active research topic is spatio-temporal adaptivity, which facilitates fast and accurate simulations in the case of large domains with complex geometries.

Acknowledgements

This work was supported by grants from the Austrian Science Fund (FWF) project number W1219-N22 (Vienna Doctoral Programme on Water Resource Systems), the Austrian Research Promotion Agency (FFG) within the scope of the COMET K1 program (Project Nr. 854174), and the European Research Council (ERC) Advanced Grant FloodChange, project number 291152. We thank Steb Cologne, riocom, and Prof. Guoxian Chen, Wuhan University, China.

Chapter V

Conclusions and Outlook

1 Summary

The overall goal of this thesis is to develop and implement a robust shallow water equations (SWE) scheme capable to simulate large-scale real-world floods on high resolution meshes. In order to do so, the implementation is done using the latest technologies in parallel algorithms and graphics processing units (GPUs). Flood simulation systems are usually tailored to solve a specific problem, scenario or they are simply too slow. These properties make them unacceptable for real-world decision making in flood management, where large ensembles of flood scenarios have to be simulated to handle the prediction uncertainty. The simulation system needs to be capable of adapting to various scenarios, since real-world applications differ case-by-case. In this work we address these issues by delivering a solution which involves the development of a new two-dimensional SWE scheme, its parallel implementation on GPUs and validation on multiple cases.

In recent years, a lot of effort has been made towards efficient GPU-based solvers for shallow water equations. However, their application to real-world scenarios is often plagued with significant slow-downs. This happens due to numerical instabilities, like spurious high velocities generated at dry/wet boundaries. Such slow-downs are unacceptable for real-world decision making in flood management, where large ensembles of flood scenarios have to be simulated to handle the prediction uncertainty. Keeping this problem in mind, we developed the HWP scheme, which is presented in Chapter II. The new scheme improves the accuracy of the solution along the dry/wet boundaries which results in stabilized velocities and increased time step sizes. This is achieved by applying a new reconstruction for the free surface which not only ensures well-balanced state and positive water height, but tackles also the non-physical velocities that appear in the partially flooded cells. To guarantee mass conservation an additional step to the global time step (GTS) computation is needed. For each partially flooded cell's interface which is in the state of draining, a special time step is computed, called the draining time step (DTS). The DTS is used to determine the time needed to completely drain a cell. When advancing the solution in time the smallest of the GTS and DTS is taken, to make sure the mass remains conserved.

Since the new reconstruction technique requires a more complex algorithm and involves many additional computations, it represents an extra burden on the GPU. The new technique increases the usage of GPU resources such as shared memory and register space. Due to the extra work load, it limits the occupancy and utilization of the device. A possible way how to tackle this is presented in Chapter III. With the new shuffle instructions, first time introduced in GPUs based on the Kepler architecture, one can decrease the burden on the limited resources by exchanging data directly between threads without the need for shared memory. By using the new HWP scheme and the shuffle instructions along with a profiler-driven development and design, we successfully outperform the previous KP scheme in large-scale simulations. Moreover, the HWP scheme avoids the build-up of unrealistically high velocities close to the dry/wet boundaries, which actually end up contributing to the loss of efficiency for the KP scheme. The timings of the presented ensemble simulations suggest that even multiple scenarios can be explored prior to the predicted flooding event. The algorithms from Chapter III can be transferred to other parallel platforms and architectures that allow programmers to directly share data between threads using registers or shared (L2) memory, e. g., OpenCL has support for shuffle instructions similarly to the CUDA API.

Finally, in Chapter IV a comprehensive comparison and validation of multiple shallow water schemes is presented, on analytic, laboratory and real-world cases. We compare the HWP scheme with the KP scheme and with the first-order CN scheme. In the comparisons, we focus on the prediction of the wave arrival times and the water levels, since they are of key importance for decision makers. In Chapter IV, we discuss also all boundary conditions (BCs) used in the validation cases, such as, up- and downstream BCs, wall BCs used for buildings and flood protection walls. The presented validation cases reveal both strengths and weaknesses of the investigated schemes. The results of the real-world cases show that the second-order schemes, KP and HWP,

provide better estimations of the measured values. However, for smaller cell sizes non-physical oscillations occur in both of them. The same oscillations also happen in the U-shaped and sine-generated flume case studies, where high mesh resolutions are needed to resolve the curved wall boundaries, preventing the schemes to reach a steady state. The exact mechanism leading to these oscillations, as well as their dependency on the mesh size are still to be investigated. In contrast, the CN scheme provides smooth solutions without oscillations even in the case of a high mesh resolution. Nevertheless, the first-order CN scheme cannot predict the measured values with the same accuracy as the KP and HWP schemes. Taking all into account, the simulation results are in good agreement with the measured values. Hence they can provide decision makers with useful information for real-world flood management and for creating action plans.

2 Future Works

Future studies should focus on improving the accuracy, robustness and performance of the presented solutions. Possible future works involve second-order extension of the CN scheme while maintaining its stability and speed. In order to allow for proper modelling of rivers floods, modelling of hydraulic structures, like dikes, embankments, weirs, should be integrated into the simulations. In addition, for more robust modelling of urban floods, surface water simulation with coupled sewer simulations are needed. Another issue to be addressed is a more accurate handling of curved and non-aligned lines, where the cut-cells method [113, 26] could provide a remedy. Furthermore, spatio-temporal adaptivity is also an active research topic both from a numerical and a GPU implementation point of view [23, 24, 42]. Incorporating anisotropic porosity [138, 139] into the system might give an additional speed-up which would be beneficial in creation of fast estimates. Anisotropic porosity accounts for sub-grid scale effects and allows for faster simulations on coarser grids while maintaining an acceptable accuracy in comparison with higher resolution grids. This is a useful feature to deliver less accurate but fast estimations of flood scenarios.

In summary, we foresee an exciting and challenging future for flood simulations, combined with GPUs and hardware-adapted parallel algorithms for faster and more accurate decision making.

Bibliography

[1] Zsolt Horváth, Jürgen Waser, Rui AP Perdigão, Artem Konev, and Günter Blöschl. A two-dimensional numerical scheme of dry/wet fronts for the saint-venant system of shallow water equations. *International Journal for Numerical Methods in Fluids*, 77(3):159–182, 2015.

[2] J Hall, Berit Arheimer, M Borga, R Brázdil, Pierluigi Claps, A Kiss, TR Kjeldsen, J Kriauciuniene, ZW Kundzewicz, Michel Lang, et al. Understanding flood regime changes in europe: A state of the art assessment. *Hydrology and Earth System Sciences*, 18(7):2735–2772, 2014.

[3] Günter Blöschl, Ladislav Gaál, Julia Hall, Andrea Kiss, Jürgen Komma, Thomas Nester, Juraj Parajka, Rui AP Perdigão, Lenka Plavcová, Magdalena Rogger, et al. Increasing river floods: fiction or reality? *Wiley Interdisciplinary Reviews: Water*, 2(4):329–344, 2015.

[4] Brenden Jongman, Stefan Hochrainer-Stigler, Luc Feyen, Jeroen CJH Aerts, Reinhard Mechler, WJ Wouter Botzen, Laurens M Bouwer, Georg Pflug, Rodrigo Rojas, and Philip J Ward. Increasing stress on disaster-risk finance due to large floods. *Nature Climate Change*, 4(4):264–268, 2014.

[5] Josè I Barredo. Normalised flood losses in europe: 1970–2006. *Natural Hazards and Earth System Sciences*, 9(1):97–104, 2009.

[6] P Watkiss. The climate cost project. final report, volume 1: Europe. stockholm environmental institute, sweden. Technical report, ISBN 978-91-86125-35-6, 2011.

[7] Bernhard Lehner, Petra Döll, Joseph Alcamo, Thomas Henrichs, and Frank Kaspar. Estimating the impact of global change on flood and drought risks in europe: a continental, integrated analysis. *Climatic Change*, 75(3):273–299, 2006.

[8] Nicola Lugeri, Elisabetta Genovese, Carlo Lavalle, and Ad De Roo. Flood risk in europe: analysis of exposure in 13 countries. *European Commission Directorate-General Joint Research Centre, EUR22525 EN*, 2006.

[9] EU Civel Protection. Directive 2007/60/ec of the european parliament and of the council of 23 october 2007 on the assessment and management of flood risks. *Official Journal of the European Union*, L288:27–34, 2007.

[10] R Merz, G Blöschl, and G Humer. Hochwasserabflüsse in österreich–das hora-projekt. *Österreichische Wasser-und Abfallwirtschaft*, 60(9):129–138, 2008.

[11] Roger Temam. *Navier-stokes equations*, volume 2. North-Holland Amsterdam, 1984.

[12] AJC de Saint-Venant. 'theorie du mouvement non permanent des eaux, avec application aux crues des rivieres et a l'introduction de marees dans leurs lits.". *Comptes rendus des seances de l'Academie des Sciences*, 36:174–154, 1871.

[13] Robert Bridson. *Fluid simulation for computer graphics*. CRC Press, 2015.

[14] Emmanuel Audusse, François Bouchut, Marie-Odile Bristeau, Rupert Klein, and Benoît Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM Journal on Scientific Computing*, 25(6):2050–2065, 2004.

[15] Emmanuel Audusse and Marie-Odile Bristeau. A well-balanced positivity preserving "second-order" scheme for shallow water flows on unstructured meshes. *Journal of Computational Physics*, 206(1):311–333, 2005.

[16] Andreas Bollermann, Guoxian Chen, Alexander Kurganov, and Sebastian Noelle. A well-balanced reconstruction of wet/dry fronts for the shallow water equations. *J. Sci. Comput.*, 56(2):267–290, August 2013.

[17] Alexander Kurganov and Doron Levy. Central-upwind schemes for the saint-venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 36(3):397–425, 2002.

[18] Alexander Kurganov and Guergana Petrova. A second-order well-balanced positivity preserving central-upwind scheme for the saint-venant system. *Communications in Mathematical Sciences*, 5(1):133–160, 2007.

[19] Boris Diskin and James L Thomas. Comparison of node-centered and cell-centered unstructured finite-volume discretizations: inviscid fluxes. *AIAA journal*, 49(4):836–854, 2011.

[20] Richard Comblen, Sébastien Legrand, Eric Deleersnijder, and Vincent Legat. A finite element method for solving the shallow water equations on the sphere. *Ocean Modelling*, 28(1):12–23, 2009.

[21] Po-Wei Li and Chia-Ming Fan. Generalized finite difference method for two-dimensional shallow water equations. *Engineering Analysis with Boundary Elements*, 80:58–71, 2017.

[22] Hamidreza Shirkhani, Abdolmajid Mohammadian, Ousmane Seidou, and Alexander Kurganov. A well-balanced positivity-preserving central-upwind scheme for shallow water equations on unstructured quadrilateral grids. *Computers & Fluids*, 126:25–40, 2016.

[23] Georges Kesserwani and Qiuhua Liang. Dynamically adaptive grid based discontinuous galerkin shallow water model. *Advances in Water Resources*, 37:23–39, 2012.

[24] Qiuhua Liang. A simplified adaptive cartesian grid system for solving the 2d shallow water equations. *International Journal for Numerical Methods in Fluids*, 69(2):442–458, 2012.

[25] David M Ingram, Derek M Causon, and Clive G Mingham. Developments in cartesian cut cell methods. *Mathematics and Computers in Simulation*, 61(3):561–572, 2003.

[26] Lennart Schneiders, Daniel Hartmann, Matthias Meinke, and Wolfgang Schröder. An accurate moving boundary formulation in cut-cell methods. *Journal of Computational Physics*, 235:786–809, 2013.

[27] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[28] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[29] Qiuhua Liang, Xilin Xia, and Jingming Hou. Efficient urban flood simulation using a gpu-accelerated sph model. *Environmental Earth Sciences*, 74(11):7285–7294, 2015.

[30] Xilin Xia and Qiuhua Liang. A gpu-accelerated smoothed particle hydrodynamics (sph) model for the shallow water equations. *Environmental Modelling & Software*, 75:28–43, 2016.

[31] R Vacondio, BD Rogers, and PK Stansby. Smoothed particle hydrodynamics: Approximate zero-consistent 2-d boundary conditions and still shallow-water tests. *International Journal for Numerical Methods in Fluids*, 69(1):226–253, 2012.

[32] Xilin Xia, Qiuhua Liang, Manuel Pastor, Weilie Zou, and Yan-Feng Zhuang. Balancing the source terms in a sph model for solving the shallow water equations. *Advances in Water Resources*, 59:25–38, 2013.

[33] A Barreiro, AJC Crespo, JM Domínguez, and M Gómez-Gesteira. Smoothed particle hydrodynamics applied in fluid structure interactions. *Fluid Structure Interaction VII*, 129:75, 2013.

[34] Tsang-Jung Chang, Kao-Hua Chang, and Hong-Ming Kao. A new approach to model weakly nonhydrostatic shallow water flows in open channels with smoothed particle hydrodynamics. *Journal of Hydrology*, 519:1010–1019, 2014.

[35] Damien Violeau and Benedict D Rogers. Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016.

[36] Shane Cook. *CUDA programming: a developer's guide to parallel computing with GPUs*. Applications of GPU Computing Series. Elsevier Science, 2012.

[37] Nicholas Wilt. *The cuda handbook: A comprehensive guide to gpu programming*. Pearson Education, 2013.

[38] André R Brodtkorb, Trond R Hagen, Knut-Andreas Lie, and Jostein R Natvig. Simulation and visualization of the saint-venant system using gpus. *Computing and Visualization in Science*, 13(7):341–353, 2010.

[39] André R. Brodtkorb, Martin L. Sætra, and Mustafa Altinakar. Efficient shallow water simulations on gpus: Implementation, visualization, verification, and validation. *Computers & Fluids*, 55(0):1–12, 2012.

[40] André R Brodtkorb and Martin L Sætra. Explicit shallow water simulations on gpus: Guidelines and best practices. In *XIX International Conference on Water Resources, CMWR*, pages 17–22, 2012.

[41] Martin L Sætra. Shallow water simulation on gpus for sparse domains. In *Numerical Mathematics and Advanced Applications 2011*, pages 673–680. Springer, 2013.

[42] Martin L Sætra, André R Brodtkorb, and Knut-Andreas Lie. Efficient gpu-implementation of adaptive mesh refinement for the shallow-water equations. *Journal of Scientific Computing*, 63(1):23–48, 2015.

[43] John Cheng, Max Grossman, and Ty McKercher. *Professional Cuda C Programming*. John Wiley & Sons, 2014.

[44] David B Kirk and W Hwu Wen-Mei. *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2016.

[45] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. NVIDIA Corporation, 2017.

[46] Jürgen Waser, Artem Konev, Bernhard Sadransky, Zsolt Horváth, H Ribičić, Robert Carnecky, P Kluding, and Benjamin Schindler. Many plans: Multidimensional ensembles for visual decision support in flood management. In *Computer Graphics Forum*, volume 33, pages 281–290. Wiley Online Library, 2014.

[47] Visdom - an integrated visualization system. `http://visdom.at`, 2017. Last visited on 2017-04-26.

[48] M Acuña and Takayuki Aoki. Real-time tsunami simulation on multi-node gpu cluster. In *ACM/IEEE conference on supercomputing*, 2009.

[49] Giuliano Di Baldassarre, Alberto Montanari, Harry Lins, Demetris Koutsoyiannis, Luigia Brandimarte, and Günter Blöschl. Flood fatalities in africa: from diagnosis to mitigation. *Geophysical Research Letters*, 37(22), 2010.

[50] Rui AP Perdigao and Günter Blöschl. Spatiotemporal flood sensitivity to annual precipitation: Evidence for landscape-climate coevolution. *Water Resources Research*, 50(7):5492–5509, 2014.

[51] Trond Runar Hagen, Jon M Hjelmervik, K-A Lie, Jostein R Natvig, and M Ofstad Henriksen. Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory*, 13(8):716–726, 2005.

[52] Clint Dawson and Christopher M Mirabito. The shallow water equations. *Lecture slides from the Institute for Computational Engineering and Sciences, University of Texas at Austin, USA*, 2008.

[53] Marc de la Asunción, José Mantas, and Manuel Castro. Programming cuda-based gpus to simulate two-layer shallow water flows. *Euro-par 2010-parallel processing*, pages 353–364, 2010.

[54] Alexander Kurganov and Guergana Petrova. Central-upwind schemes for two-layer shallow water equations. *SIAM Journal on Scientific Computing*, 31(3):1742–1773, 2009.

[55] Jean A Cunge and M Wegner. Intégration numérique des équations d'écoulement de barré de saint-venant par un schéma implicite de différences finies. *La Houille Blanche*, 1:33–39, 1964.

[56] Jian G Zhou, Derek M Causon, Clive G Mingham, and David M Ingram. The surface gradient method for the treatment of source terms in the shallow-water equations. *Journal of Computational physics*, 168(1):1–25, 2001.

[57] Tobias Brandvik and Graham Pullan. Acceleration of a two-dimensional euler flow solver using commodity graphics hardware. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 221(12):1745–1748, 2007.

[58] Tobias Brandvik and Graham Pullan. Acceleration of a 3d euler solver using commodity graphics hardware. In *46th AIAA aerospace sciences meeting and exhibit*, page 607, 2008.

[59] Marc De La Asunción, José M Mantas, and Manuel J Castro. Simulation of one-layer shallow water systems on multicore and cuda architectures. *The Journal of Supercomputing*, 58(2):206–214, 2011.

[60] Miguel Lastra, José M Mantas, Carlos Ureña, Manuel J Castro, and José A García-Rodríguez. Simulation of shallow-water systems using graphics processing units. *Mathematics and Computers in Simulation*, 80(3):598–618, 2009.

[61] Sebastian Noelle, Normann Pankratz, Gabriella Puppo, and Jostein R Natvig. Well-balanced finite volume schemes of arbitrary order of accuracy for shallow water flows. *Journal of Computational Physics*, 213(2):474–499, 2006.

[62] Giovanni Russo. Central schemes for conservation laws with application to shallow water equations. In *Trends and Applications of Mathematics to Mechanics*, pages 225–246. Springer, 2005.

[63] Yulong Xing and Chi-Wang Shu. High order finite difference weno schemes with the exact conservation property for the shallow water equations. *Journal of Computational Physics*, 208(1):206–227, 2005.

[64] Benoıt Perthame and Chiara Simeoni. A kinetic scheme for the saint-venant system¶ with a source term. *Calcolo*, 38(4):201–231, 2001.

[65] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[66] David Gottlieb and Eitan Tadmor. The cfl condition for spectral approximations to hyperbolic initial-boundary value problems. *Mathematics of Computation*, 56(194):565–588, 1991.

[67] Andreas Bollermann, Sebastian Noelle, and Maria Lukáčová-Medvid'ová. Finite volume evolution galerkin methods for the shallow water equations with dry beds. *Communications in Computational Physics*, 10(02):371–404, 2011.

[68] C Hinterberger, J Fröhlich, and W Rodi. Three-dimensional and depth-averaged large-eddy simulations of some shallow water flows. *Journal of Hydraulic Engineering*, 133(8):857–872, 2007.

[69] Normann Pankratz, Jostein R Natvig, Bjørn Gjevik, and Sebastian Noelle. High-order well-balanced finite-volume schemes for barotropic flows: Development and numerical comparisons. *Ocean modelling*, 18(1):53–79, 2007.

[70] Yih-Chin Tai, S Noelle, JMNT Gray, and K Hutter. Shock-capturing and front-tracking methods for granular avalanches. *Journal of Computational Physics*, 175(1):269–301, 2002.

[71] Steve Bryson, Yekaterina Epshteyn, Alexander Kurganov, and Guergana Petrova. Well-balanced positivity preserving central-upwind scheme on triangular grids for the saint-venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 45(3):423–446, 2011.

[72] Alexander Kurganov, Sebastian Noelle, and Guergana Petrova. Semidiscrete central-upwind schemes for hyperbolic conservation laws and hamilton–jacobi equations. *SIAM Journal on Scientific Computing*, 23(3):707–740, 2001.

[73] Knut-Andreas Lie and Sebastian Noelle. On the artificial compression method for second-order nonoscillatory central difference schemes for systems of conservation laws. *SIAM Journal on Scientific Computing*, 24(4):1157–1174, 2003.

[74] Haim Nessyahu and Eitan Tadmor. Non-oscillatory central differencing for hyperbolic conservation laws. *Journal of computational physics*, 87(2):408–463, 1990.

[75] Peter K Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM journal on numerical analysis*, 21(5):995–1011, 1984.

[76] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[77] Dietmar Kröner. *Numerical schemes for conservation laws*, volume 22. Wiley Chichester, 1997.

[78] Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[79] Pierre-Arnaud Raviart and E Godlewski. Numerical approximation of hyperbolic systems of conservation laws. *Appl. Math. Sci*, 118, 1996.

[80] Bernd Einfeldt. On godunov-type methods for gas dynamics. *SIAM Journal on Numerical Analysis*, 25(2):294–318, 1988.

[81] Amiram Harten, Peter D Lax, and Bram Van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. In *Upwind and High-Resolution Schemes*, pages 53–79. Springer, 1997.

[82] Andreas Bollermann, Guoxian Chen, Alexander Kurganov, and Sebastian Noelle. A well-balanced reconstruction for wetting/drying fronts. *arXiv preprint arXiv:1412.3580*, 2014.

[83] William Carlisle Thacker. Some exact solutions to the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508, 1981.

[84] Marc de la Asunción, Manuel J Castro, ED Fernández-Nieto, José M Mantas, Sergio Ortega Acosta, and José Manuel González-Vida. Efficient gpu implementation of a two waves tvd-waf method for the two-dimensional one layer shallow water system on structured meshes. *Computers & Fluids*, 80:441–452, 2013.

[85] José M Gallardo, Carlos Parés, and Manuel Castro. On a well-balanced high-order finite volume scheme for shallow water equations with topography and dry areas. *Journal of Computational Physics*, 227(1):574–601, 2007.

[86] Q. Liang and F. Marche. Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in Water Resources*, 32:873–884, June 2009.

[87] Joe Sampson, Alan Easton, and Manmohan Singh. Moving boundary shallow water flow above parabolic bottom topography. *Anziam Journal*, 47:C373–C387, 2006.

[88] Runar Holdahl, Helge Holden, and Knut-Andreas Lie. Unconditionally stable splitting methods for the shallow water equations. *BIT Numerical Mathematics*, 39(3):451–472, 1999.

[89] Nathan Whitehead and Alex Fit-Florea. Precision & performance: Floating point and ieee 754 compliance for nvidia gpus. *rn (A+ B)*, 21:1–1874919424, 2011.

[90] Hrvoje Ribičić, Jürgen Waser, Raphael Fuchs, Günter Blöschl, and Eduard Gröller. Visual analysis and steering of flooding simulations. *IEEE transactions on visualization and computer graphics*, 19(6):1062–1075, 2013.

[91] Asier Lacasta, Mario Morales-Hernández, Javier Murillo, and Pilar García-Navarro. Gpu implementation of the 2d shallow water equations for the simulation of rainfall/runoff events. *Environmental Earth Sciences*, 74(11):7295–7305, 2015.

[92] Asier Lacasta, Mario Morales-Hernández, Javier Murillo, and Pilar García-Navarro. An optimized gpu implementation of a 2d free surface simulation model on unstructured meshes. *Advances in Engineering Software*, 78:1–15, 2014.

[93] Derek M Causon, David M Ingram, and Clive G Mingham. A cartesian cut cell method for shallow water flows with moving boundaries. *Advances in water Resources*, 24(8):899–911, 2001.

[94] Trond Runar Hagen, Knut-Andreas Lie, and Jostein R Natvig. Solving the euler equations on graphics processing units. In *International Conference on Computational Science*, pages 220–227. Springer, 2006.

[95] Andreas Klöckner, Tim Warburton, Jeff Bridge, and Jan S Hesthaven. Nodal discontinuous galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[96] Peng Wang, Tom Abel, and Ralf Kaehler. Adaptive mesh fluid simulations on gpu. *New Astronomy*, 15(7):581–589, 2010.

[97] Wen-Yew Liang, Tung-Ju Hsieh, Muhammad T Satria, Yang-Lang Chang, Jyh-Perng Fang, Chih-Chia Chen, and Chin-Chuan Han. A gpu-based simulation of tsunami propagation and inundation. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 593–603. Springer, 2009.

[98] R Vacondio, A Dal Palù, and P Mignosa. Gpu-enhanced finite volume shallow water solver for fast flood simulations. *Environmental Modelling & Software*, 57:60–75, 2014.

[99] Arnaud Duran, Q Liang, and Fabien Marche. On the well-balanced numerical discretization of shallow water equations on unstructured meshes. *Journal of Computational Physics*, 235:565–586, 2013.

[100] Gang Li, Jinmei Gao, and Qiuhua Liang. A well-balanced weighted essentially non-oscillatory scheme for pollutant transport in shallow water. *International Journal for Numerical Methods in Fluids*, 71(12):1566–1587, 2013.

[101] Jingming Hou, Qiuhua Liang, Franz Simons, and Reinhard Hinkelmann. A 2d well-balanced shallow flow model for unstructured grids with novel slope source term treatment. *Advances in Water Resources*, 52:107–131, 2013.

[102] Chi-Wang Shu. Total-variation-diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073–1084, 1988.

[103] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing runge-kutta schemes. *Mathematics of computation of the American Mathematical Society*, 67(221):73–85, 1998.

[104] Benjamin Schindler, Jürgen Waser, Hrvoje Ribičić, Raphael Fuchs, and Ronald Peikert. Multiverse data-flow control. *IEEE transactions on visualization and computer graphics*, 19(6):1005–1019, 2013.

[105] Open TELEMAC-MASCARET. http://www.opentelemac.org/, 2017. Last visited on 2015-06-24.

[106] S Soares Frazao, F Alcrudo, and N Goutal. Dam-break test cases summary. In *Proceedings of the Fourth CADAM Meeting*, pages 9–25, 1999.

[107] Jean-Michel Hervouet and Alain Petitjean. Malpasset dam-break revisited with two-dimensional computations. *Journal of Hydraulic Research*, 37(6):777–788, 1999.

[108] DL George. Adaptive finite volume methods with well-balanced riemann solvers for modeling floods in rugged terrain: Application to the malpasset dam-break flood (france, 1959). *International Journal for Numerical Methods in Fluids*, 66(8):1000–1018, 2011.

[109] Gary W Brunner. Hec-ras (river analysis system). In *North American Water and Environment Congress & Destructive Water:*, pages 3782–3787. ASCE, 2010.

[110] Jean-Charles Galland, Nicole Goutal, and Jean-Michel Hervouet. Telemac: A new numerical model for solving shallow water equations. *Advances in Water Resources*, 14(3):138 – 148, 1991.

[111] Daniel Cornel, Artem Konev, Berhard Sadransky, Zsolt Horvath, Meister Eduard Gröller, and Jügen Waser. Visualization of object-centered vulnerability to possible flood hazards. *Computer Graphic Forum*, 34(3):331–340, June 2015. 3rd Best Paper Award.

[112] Marc de la Asunción, Manuel J Castro, ED Fernández-Nieto, José M Mantas, Sergio Ortega Acosta, and José Manuel González-Vida. Efficient GPU implementation of a two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes. *Computers & Fluids*, 80:441–452, 2013.

[113] Matthias Meinke, Lennart Schneiders, Claudia Günther, and Wolfgang Schröder. A cut-cell method for sharp moving boundaries in cartesian grids. *Computers & Fluids*, 85:135–142, 2013.

[114] Zsolt Horváth, Rui AP Perdigão, Jürgen Waser, Daniel Cornel, Artem Konev, and Günter Blöschl. Kepler shuffle for real-world flood simulations on gpus. *The International Journal of High Performance Computing Applications*, 30(4):379–395, 2016.

[115] R Vacondio, A Ferrari, P Mignosa, F Aureli, and A Dal. Efficient non-uniform grid for gpu-parallel shallow water equations models. In *River Flow 2016*, pages 281–288. CRC Press, 2016.

[116] Vincenzo Casulli and Guus S. Stelling. A semi-implicit numerical model for urban drainage systems. *International Journal for Numerical Methods in Fluids*, 73(6):600–614, 2013.

[117] Guoxian Chen and Sebastian Noelle. A new hydrostatic reconstruction scheme based on subcell reconstructions. *SIAM Journal on Numerical Analysis*, 55(2):758–784, 2017.

[118] Dale R. Durran. *Numerical methods for fluid dynamics; with applications to geophysics*. Texts in applied mathematics ; 32. Springer, New York, NY [u.a.], 2. ed. edition, 2010.

[119] Jingming Hou, Qiuhua Liang, Hongbin Zhang, and Reinhard Hinkelmann. Multislope muscl method applied to solve shallow water equations. *Computers & Mathematics with Applications*, 68(12):2012–2027, 2014.

[120] Emmanuel Audusse, Christophe Chalons, and Philippe Ung. A simple well-balanced and positive numerical scheme for the shallow-water system. *Communications in Mathematical Sciences*, January 2015.

[121] François Bouchut and Tomas Morales De Luna. A subsonic-well-balanced reconstruction scheme for shallow water flows. *SIAM Journal on Numerical Analysis*, 48(5):1733–1758, 2010.

[122] MJ Castro Díaz, Juan A López-García, and Carlos Parés. High order exactly well-balanced numerical methods for shallow water systems. *Journal of Computational Physics*, 246:242–264, 2013.

[123] Olivier Delestre, Stéphane Cordier, Frédéric Darboux, and Francois James. A limitation of the hydrostatic reconstruction technique for shallow water equations. *Comptes Rendus Mathematique*, 350(13-14):677–681, 2012.

[124] Tomás Morales de Luna, Manuel J. Castro Díaz, and Carlos Parés. Reliability of first order numerical schemes for solving shallow water system over abrupt topography. *Applied Mathematics and Computation*, 219(17):9012–9032, 2013.

[125] Jean-Michel Ghidaglia and Frédéric Pascal. The normal flux method at the boundary for multidimensional finite volume approximations in cfd. *European Journal of Mechanics-B/Fluids*, 24(1):1–17, 2005.

[126] Denys Dutykh, Raphaël Poncet, and Frédéric Dias. The volna code for the numerical modeling of tsunami waves: Generation, propagation and inundation. *European Journal of Mechanics-B/Fluids*, 30(6):598–615, 2011.

[127] Olivier Delestre, Carine Lucas, Pierre-Antoine Ksinant, Frédéric Darboux, Christian Laguerre, T-N Vo, François James, Stéphane Cordier, et al. Swashes: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, 72(3):269–300, 2013.

[128] N. Goutal and F. Maurel. Note technique edf, he-43/97/016/b. In *Proceedings of the 2nd Workshop on Dam-Break Simulation*. Department Laboratoire National d'Hydraulique, Groupe Hydraulique Fluviale, 1997.

[129] Fraçnois Bouchut. Chapter 4 efficient numerical finite volume schemes for shallow water models. In V. Zeitlin, editor, *Nonlinear Dynamics of Rotating Shallow Water: Methods and Advances*, volume 2 of *Edited Series on Advances in Nonlinear Science and Complexity*, pages 189 – 256. Elsevier Science, 2007.

[130] H. J. De Vriend. Flow measurements in a curved rectangular channel. Technical report, Laboratory of Fluid Mechanics, Department of Civil Engineering, Delft University of Technology, 1979.

[131] NCCHE. http://www.ncche.olemiss.edu/publishing/Verification_and_Validation_Report/, 2017. Last visited on 2017-03-07.

[132] A.M.A.F. da Silva. *Turbulent Flow in Sine-generated Meandering Channels*. PhD thesis, Queen's University, 1995.

[133] P Brufau, ME Vázquez-Cendón, and P García-Navarro. A numerical model for the flooding and drying of irregular domains. *International Journal for Numerical Methods in Fluids*, 39(3):247–275, 2002.

[134] A. I. Delis and Th. Katsaounis. Relaxation schemes for the shallow water equations. *International Journal for Numerical Methods in Fluids*, 41(7):695–719, 2003.

[135] Jaswant Singh, Mustafa S. Altinakar, and Yan Ding. Two-dimensional numerical modeling of dam-break flows over natural terrain using a central explicit scheme. *Advances in Water Resources*, 34(10):1366 – 1375, 2011.

[136] G. Blöschl, T. Nester, J. Komma, J. Parajka, and R. A. P. Perdigão. The june 2013 flood in the upper danube basin, and comparisons with the 2002, 1954 and 1899 floods. *Hydrology and Earth System Sciences*, 17(12):5197–5212, 2013.

[137] G. Di Baldassarre and A. Montanari. Uncertainty in river discharge observations: a quantitative analysis. *Hydrology and Earth System Sciences*, 13(6):913–921, 2009.

[138] Ilhan Özgen, Dongfang Liang, and Reinhard Hinkelmann. Shallow water equations with depth-dependent anisotropic porosity for subgrid-scale topography. *Applied Mathematical Modelling*, 40(17):7447–7473, 2016.

[139] Ilhan Özgen, Jiaheng Zhao, Dongfang Liang, and Reinhard Hinkelmann. Urban flood modeling using shallow water equations with depth-dependent anisotropic porosity. *Journal of Hydrology*, 541:1165–1184, 2016.