A robust high-resolution hydrodynamic numerical model for surface water flow and transport processes within a flexible software framework

vorgelegt von

Dipl.-Ing. Franz Simons

an der Fakultät VI - Planen Bauen Umwelt
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften

Dr.-Ing.

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Matthias Barjenbruch
Gutachter:	Prof. Dr.-Ing. Reinhard Hinkelmann
	Prof. Qiuhua Liang
	apl. Prof. Dr.-Ing. Frank Molkenthin

Tag der wissenschaftlichen Aussprache: 25.10.2019

Berlin 2020

"All models are wrong, but some are useful."

George E. P. Box (1979)

To Frederik

Abstract

In water resources management computer simulations and numerical methods have become a more and more important tool for supporting decisions. They enable the understanding of complex relationships in the hydrological cycle and predictions about future development.

This thesis describes the development of a numerical modeling software which allows the integrated simulation of surface water flow and transport processes as well as their interactions. To make use of the improvements in the methods to survey high-resolution topography information and to capture small-scale processes, numerical methods were developed which allow a robust and highly detailed simulation of surface water flow and transport processes in urban and natural environments. A robust numerical scheme for the solution of the shallow water equations based on the finite volume method was implemented, which can handle complex flow conditions, e. g. small water depths, wetting/drying and varying flow conditions including sub- and supercritical flows, hydraulic jumps and sharp water level gradients. In addition, the shallow water equations were augmented by the transport of contaminants and sediments and an infiltration model based on the Green-Ampt equation. A numerical framework was developed which provides the fundamental infrastructure for explicit high-order finite volume schemes, robust numerical methods and a flexible codebase which allows simple extension by new processes and numerical schemes. By means of several verification tests and case studies involving channel flow, rainfall-runoff, tracer transport, infiltration and sediment transport, the suitability of the software framework and the developed numerical schemes was demonstrated.

Kurzfassung

In der Wasserwirtschaft sind Computersimulationen und numerische Methoden zu einem immer wichtigeren Werkzeug zur Entscheidungsunterstützung geworden. Sie ermöglichen es, komplexe Zusammenhänge im Wasserkreislauf zu verstehen und Vorhersagen über die zukünftige Entwicklung zu treffen.

Die vorliegende Arbeit beschreibt die Entwicklung einer numerischen Modellierungssoftware, die die integrierte Simulation von Strömungs- und Transportprozessen in Oberflächengewässern ermöglicht. Um auch kleinskalige Prozesse abzubilden und die Verbesserungen der Methoden zur Erfassung hochauflösender Geländemodelle nutzen zu können, wurden numerische Methoden entwickelt, die eine robuste und hochdetaillierte Simulation von Strömungs- und Transportprozessen in urbanen und natürlichen Gebieten ermöglichen. Es wurde ein robustes numerisches Verfahren zur Lösung der Flachwassergleichungen auf Basis der Finite-Volumen-Methode implementiert, das komplexe Strömungsbedingungen (z. B. kleine Wassertiefen, Benetzen und Trockenfallen, schießenden und strömenden Abfluss, Wechselsprünge und steile Wasserstandsgradienten) beherrscht. Zusätzlich wurden die Flachwassergleichungen um den Transport von Schadstoffen und Sedimenten und ein Infiltrationsmodell auf Basis der Green-Ampt-Gleichung ergänzt. Es wurde ein numerisches Software-Framework entwickelt, das den grundlegenden Rahmen für explizite Finite-Volumen-Verfahren höherer Ordnung und robuste numerische Methoden zur Verfügung stellt. Das Softwarekonzept erlaubt eine einfache Erweiterung um weitere Prozesse und numerische Verfahren. Anhand mehrerer Validierungsrechnungen und Fallstudien, die sich mit Gerinneströmung, Niederschlags-Abfluss-Modellierung, Tracertransport, Infiltration und Sedimenttransport befassen, konnte die Eignung des Software-Frameworks und der entwickelten numerischen Verfahren nachgewiesen werden.

Acknowledgments

First of all I want to express my gratitude to my doctorate supervisor Professor Reinhard Hinkelmann for giving me the opportunity to work for six year as a scientific assistant at the Chair of Water Resources Management and Modeling of Hydrosystems in the Department of Civil Engineering at Technische Universität Berlin. Thank you for your guidance and always supporting my work! Gratitude is also expressed to my second advisor apl. Professor Frank Molkenthin for his willingness to supervise my thesis and for his suggestions and support in all topics related to software development.

Next, I want to thank Tobias Busse, whose work is base and reason of this thesis. I joined his research topic when writing my diploma thesis and learned a lot on software development under his supervision.

I want to thank Jingming Hou, Vikram Notay, Ilhan Özgen and Leopold Stadler for their support, encouragement and ongoing friendship. Leo, thank you for pushing me across the finish line! My gratitude is also expressed to Ralf Duda and all other former colleagues who made my work at the chair to a time I gladly remember.

Finally, I want to sincerely thank my wife Victoria and my sister Evalotte and my parents Elisabeth and Martin for their encouragement, support and love.

All of those people and many more contributed to the success of this thesis. Thank you!

Karlsruhe, January 2020 – Franz Simons

Contents

1 Introduction

1.1 Motivation

Water has always been very important in the history of mankind. It is the source of drinking water, food and energy, and is a way of transportation. The absence of fresh water due to draughts, salinization and heavy use in urban areas is a dangerous threat and causes ongoing water conflicts. Water also causes severe hazards such as floods, tsunamis and landslides. Seen on a much larger time scale water gains even more importance: it causes steady and ongoing changes to the earth's surface and has formed some of the most spectacular landscapes we can experience today.

Unsurprisingly, a lot of scientific research is concentrated on water. Major topics of interest are the quantity and quality of water and the study of water-induced hazards and possible protection measures. Nowadays, special emphasis is placed on the human influence on the natural processes. We have become aware of the fact that fresh water is a limited resource and that quantity, quality, and potential hazards of water are strongly influenced by climate change and human activity.

One key component of the hydrologic cycle is *surface water flow* which describes the entirety of all water flowing on the earth's surface driven by gravitational forces. It is made up of *surface runoff* caused by precipitation and the flow in creeks and rivers, the so-called *stream* or *channel flow* (Emmett, 1978; Hillel, 1998). Furthermore, surface runoff can be subdivided into the thin sheetlike *overland flow*, which is the primary type in runoff from small natural areas and the flow in small rills and gullies (Hillel, 1998). Surface water flow and its impacts are strongly influenced by land use. Soil sealing due to urbanization intensifies surface runoff and raises the risk of flash floods during and after heavy rainfall events. Deforestation and agricultural usage of land can lead to strong erosion resulting in the loss of fertile soil and the silting up of reservoirs etc. Further problems related to land-use change are the transport of pollutants due to dissolution in runoff or adsorption by soil particles. Examples for these non-point sources are the emission of fertilizers and pesticides

from agricultural areas or pollutants from streets and urban areas due to rainfall (e. g. Massoudieh and Kayhanian, 2009; Shen and Phanikumar, 2010; He et al., 2013).

In the year 2000 the European Union (EU) brought the Water Framework Directive (WFD) into force. Its aims are a good ecological and chemical status for all surface water bodies and a good chemical and quantitative status for all groundwater bodies. In the context of reaching these aims, there are two important aspects. First, the directive enjoins a water management based on river basins, i. e. the domain of interest is not anymore defined by administrative and political boundaries, but by the catchment area of the river which is defined by the topography. This involves collaboration across administrative districts and even country's frontiers (WFD art. 3, European Union, 2000).

The second aspect is the introduction of a combined approach in water pollution control (WFD art. 10, European Union, 2000). On one hand measures have to be taken to reduce the emission of pollutants at the source as much as possible with state of art technology. On the other hand the good status in the water bodies has to be achieved and observed. This approach avoids problems due to the accumulation of pollutants, diffusive sources, or insufficient knowledge of ecotoxicological relationships.

As a further element of an integrated river basin management, in the year 2007 the EU Floods Directive was brought into force (European Union, 2007). This directive deals with the assessment and management of flood risks. Advantage should be taken of common synergies and benefits which arise from the consideration of both directives, the WFD and the Floods Directive (art. 9, European Union, 2007).

At least since the publication of the Intergovernmental Panel on Climate Change (IPCC) Third Assessment Report "Climate Change 2001" it has become clear, that there is an ongoing change of climatic conditions in the world due to human activities. This implies that water resources management has to deal with new, rapidly varying conditions. It is predicted that the probability of extreme weather conditions, such as heavy rainfall events or droughts will increase and hereby the risk of floods or harvest losses. In the Fifth Assessment Report finished in the year 2014 the IPCC confirmed these findings.

The expected changes in the hydrologic water cycle due to climate change and the demands of the EU directives put more complex requirements on water resources management. To evaluate measures and impacts, new methods which allow an integrated consideration of the all participating processes are needed. Especially

the spatially and temporally variability of these processes and their response on changing climatic conditions is now of deeper interest.

As computer simulations are an important tool for supporting decisions, there is a need for software which allows the integrated simulation of surface water flow and transport processes as well as their interactions with each other. The underlying mathematical and numerical model concepts should allow a detailed investigation of the spatially and temporally varying flow field. An essential precondition is thereby provided by the recent improvements in methods to survey high-resolution topography information. Remote sensing techniques such as LIDAR and photogrammetry allow the creation of detailed digital elevation models which form the basis for highly detailed results.

The motivation is therefore to develop a software framework, which supports the integrated simulation of surface water flow and transport processes for a wide range of applications.

1.2 Existing mathematical and numerical model concepts for simulating surface water flow and transport processes

Modeling surface water flow and transport processes is of interest in hydrology as well as hydraulic engineering. In the following the historical development and state of the art are outlined.

From the hydrologist's point of view the mathematical modeling of rainfall-runoff relations has always been of interest. There is a wide range of mathematical model concepts and implementations. A historical perspective and a comprehensive overview on popular hydrologic models is given by Singh and Woolhiser (2002), Kampf and Burges (2007) and Borah (2011).

Hydrologic models can be classified by their spatial representation. *Lumped* hydrologic models treat the catchment as single unit and there is no explicit description of the processes and their variability in space. If in some way the spatial variability of the internal processes and the flow of water through the catchment is represented, a model is called a *distributed* model.

Another criteria for classification of hydrologic models is their basic mathematical foundation. The simplest approach are *empirical* or so-called "black-box" models.

They are based on mathematical input-output relationships, where effective precipitation is the input and direct surface runoff is the output of the model. There is not any consideration of physical processes in empirical models. An example is the concept of unit hydrograph introduced by Sherman (1932). In contrast, *conceptual* models describe separate processes. The input-output relationship of each process is expressed by simplified physical descriptions. An example is the linear reservoir method for expressing the retention of a catchment (e. g. Duggal and Soni, 1996). To describe the flow of water in the catchment in a distributed model the conceptual elements can be cascaded. Famous, widely used examples are the distributed conceptual models HEC-HMS (U.S. Army Corps of Engineers, 2000) and TOPMODEL (Beven and Kirkby, 1979; Beven et al., 1995).

The combined approach of the EU Water Framework Directive requires investigating not only the total discharge into a surface water body, but also the origin and the spatial and temporal distribution of the water or the transported matter. Conceptual and empirical models cannot provide the necessary information. They are suitable for investigating the total discharge of a catchment as an response to a precipitation event, allowing long term simulations and investigations of the hydrologic balance, but even distributed conceptual models can not provide details on the processes inside the catchment. Both empirical and conceptual models are based on parameters which cannot be measured in nature and have to be calibrated from measured precipitation and outflow time series.

The third, most complex class of hydrological models are *physically based* models which are derived from the partial differential equations expressing continuity of mass, momentum and energy. In the year 1969, Freeze and Harlan presented their blueprint for a distributed physical-based model and pointed out that there is a huge demand on data and computational power for the application of such models.

A physically based description of surface water flow is given by the shallow water equations. However, many distributed models only use simplified approximations to the shallow water equations (Kampf and Burges, 2007). An example is the popular distributed physically based model MIKE SHE (Bathurst and O'Connell, 1992; Refsgaard and Storm, 1995) which is based on the one- and two-dimensional solution of the diffusive wave approximation for channel flow and surface runoff, respectively. Several authors showed that the diffusive or kinematic wave can be good approximations for overland flow (e. g. Ponce et al., 1978; Moramarco and Singh, 2002; Tsai, 2003) and they are often used to simulate surface runoff (Smith and Woolhiser, 1971; Gottardi and Venutelli, 1993; Di Giammarco et al., 1996;

VanderKwaak and Loague, 2001; He et al., 2008; Weill et al., 2011). However, in cases of mild slopes, backwater effects, critical flow conditions and the influence of micro topography, the approximation shows bad results (Tayfur et al., 1993; Costabile et al., 2009; Cea et al., 2010; Yeh et al., 2011; Costabile et al., 2012).

Although, the diffusive or kinematic wave can be good approximations for overland flow under some circumstances, it becomes apparent, that using them would require a separate consideration of overland and channel flows where the approximations can lead to bad results. However, there is not always a clear transition between overland and channel flow. To simulate a wide range of surface water flow problems where flow conditions can switch between laminar and turbulent, the flow conditions can be either sub-, super- or transcritical, sharp water level gradients, small water depth and wetting/drying can occur, the full shallow water equations should be used. They also allow to take into account microtopography, which is crucial for the spatial variation of water depth and flow velocity (Zhang and Cundy, 1989; Esteves et al., 2000; Fiedler and Ramirez, 2000). Tayfur et al. (1993) state that although the consideration of the microtopography is not essential for simulating a discharge hydrograph, it is crucial for simulating transport and erosion processes which depend on the local water depth and flow velocities. "Models using the full-dynamic wave equations to route water are considered to be the most physically based models and the flows simulated by such models are considered to yield the most accurate predictions" (Borah, 2011).

In the field of hydraulic engineering, numerical models based on the full shallow water equations are widely used in engineering applications. Typical examples are the evaluation of river training measures, river bed morphology and flood water risks. Popular computer programs are TELEMAC-2D (Galland et al., 1991; Hervouet, 2007), HYDRO_AS-2D (Nujic, 1998; Hydrotec, 2018) and MIKE 21C (DHI, 2018). In addition, there is an ongoing development of robust numerical methods used to solve the shallow water equations (Hinkelmann et al., 2015). The term *robust* is used in literature to describe a numerical solution method which can handle arbitrary complex applications and stays stable, i.e. it provides a reasonable solution. The focus is also on accelerating the numerical computations using modern graphics processing units, GPUs (e. g. Lacasta et al., 2015; Liang et al., 2015), and on upscaling shallow water models (e. g. Özgen et al., 2015; Özgen, Liang and Hinkelmann, 2016).

In particular, Godunov-type schemes based on the finite volume method and approximate Riemann solvers have gained attention because they are able to handle discontinuities in the flow field (Toro and Garcia-Navarro, 2007). The so-called *C*-

property denotes the property of a numerical scheme to preserve a motionless water surface on an irregular bed at steady-state conditions (e. g. Liang and Marche, 2009; Kesserwani, 2013). A numerical scheme fulfilling the C-property is *well-balanced*. Very robust, well-balanced, high-resolution schemes that provide accurate results for cases with varying topography and wetting/drying have been presented e. g. by Audusse et al. (2004), Audusse and Bristeau (2005), Liang and Marche (2009), Song et al. (2011) and Hou et al. (2014). Here, the ability of a numerical method to resolve discontinuities is called the *resolution* of the method. A numerical scheme which smears the discontinuity less than another without introducing numerical oscillations has a *higher resolution*. The robust numerical shallow water models are also coupled with sediment transport to account for morphodynamic processes (e. g. Simpson and Castelltort, 2006; Heng et al., 2009; Liang, 2011). A comparison of different approaches is given by Zhao et al. (2017). Numerical models for shallow water transport of pollutants are for example presented by Benkhaldoun et al. (2007), Pavan et al. (2015) and Vanzo et al. (2016).

1.3 Hydroinformatics Modeling System

In software development, object-oriented design is a widely used technique. The key concepts of encapsulation, inheritance and abstraction allow the development of reusable, easily maintainable and extendible code. Several authors have already shown the advantages of an object-oriented design for building numerical models for environmental problems (e. g. Wang et al., 2005; Elshorbagy and Ormsbee, 2006; Kutija and Murray, 2007; Kolditz et al., 2008).

In the context of water and environmental related problems, the interaction of different processes is of importance and has to be taken into account. In addition, different physical and numerical model concepts for each process exist. The development of holistic models which incorporate several of these processes can be significantly improved by a flexible and extendible modeling framework which allows fast prototyping and testing of different physical and numerical methods and algorithms.

The *Hydroinformatics Modeling System* (`hms`) implements the software concepts and numerical methods presented in this thesis. Since 2007, `hms` is developed and continually improved at the Chair of Water Resources Management and Modeling of Hydrosystems, Technische Universität Berlin, Germany (TU Berlin, 2017). Its

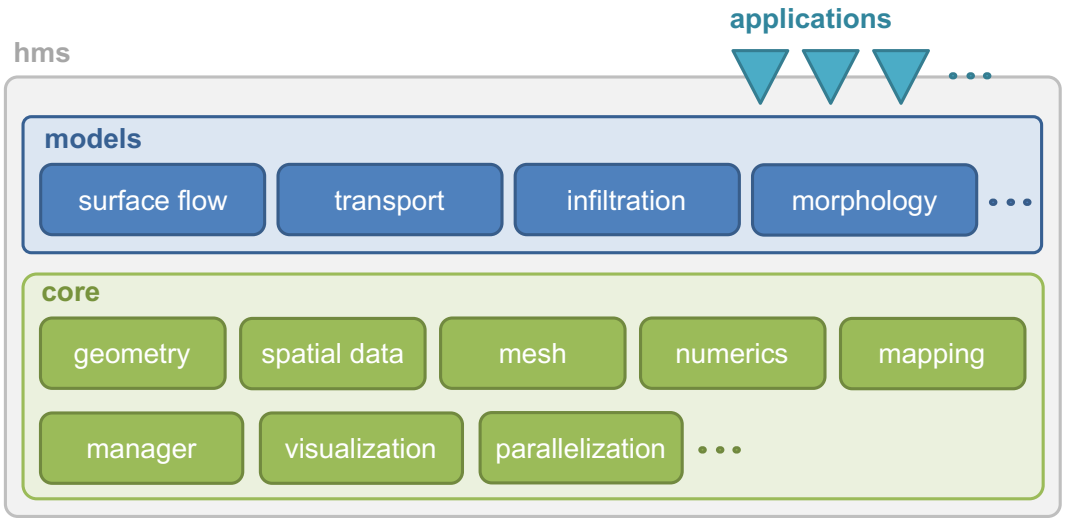


Figure 1.1: Pictorial representation of the hms software framework

origin was laid in the work of Busse et al. (2007), Busse et al. (2012) and Busse (2017). "hms is a framework and runtime environment, which enables the implementation and simulation of single and multiple processes in different spatial and temporal resolutions, as well as their interactions" (Busse, 2017). Thereby, the focus is on the simulation of water and environment related problems which extend to a part of the earth's surface and can be described in two dimensions. In a diploma thesis a finite volume method was implemented for solving the full shallow water equations using a first-order accurate upwind scheme on adaptive quadtree meshes (Simons, 2008). Mieth (2008) demonstrated coupling hms with other numerical software using standardized interfaces. Özgen (2011*a*) contributed to the development of the high-order accurate solver presented in Chapter 3. Matta et al. (2014) implemented the wind influence in the presented numerical model and presented an application on a reservoir in Brazil. Based on the software architecture presented in Chapter 4 Özgen et al. implemented coarse grid approaches for the numerical shallow water model, which consider the influence of small-scale topography on coarse grids (Özgen et al., 2015; Özgen, Liang and Hinkelmann, 2016; Özgen, Zhao, Liang and Hinkelmann, 2016; Özgen, 2017). Lately, the robust shallow water model presented in this thesis is applied to the simulation of flash floods in wadi systems (Tügel et al., 2018).

hms has a component-based architecture, i. e. the spectrum of tasks is separated into components which on one hand are used to build larger more complex components and on the other hand can be used alone or as building blocks in other contexts.

Therefore, all components are designed as independent from each other as possible. Figure 1.1 shows a pictorial representation of the framework, a detailed description is given in Section 4.1. Based on the core components, numerical models can be implemented. The actual applications are created by combining one or multiple numerical models.

1.4 Objectives

Figure 1.2 illustrates the objectives of this thesis:

The focus is on the development of a numerical modeling software, which allows to simulate surface flow processes in natural and urban environments. It should be possible to simulate both, rainfall-runoff and channel flow, without artificial separation of the processes. This means, the numerical model has to be able to handle a wide range of complex hydraulic conditions e. g. small water depths, wetting/drying and varying flow conditions including sub- and supercritical flows, hydraulic jumps and sharp water level gradients. The EU directives require an integrated view on environmental processes, e. g. the elution and transport of contaminants. In the developed software a multi-process approach will be followed. Further processes as infiltration of water and the transport of contaminants and sediment as well as morphological evolution will be considered. The numerical model has to consider the interactions between surface flow and transport processes.

Surface water flow and transport processes are strongly influenced by local details of the surface structure and land use. Therefore, the numerical model should take advantage from the improvements in the surveying methods to gather high-resolution topography information and should allow incorporating this data in the simulation.

All this requires numerical methods which allow a robust and detailed simulation of the processes. Therefore a robust high-resolution solver for the depth-averaged shallow water equations is implemented to allow highly detailed simulations. A numerical scheme based on an approximate Riemann solver is presented. The resolution of the scheme is increased by high-order reconstruction using total variation diminishing methods to guarantee the monotonicity of the solution. This solver is enhanced by the possibility to add further processes as infiltration and transport of contaminants or sediment. The Green-Ampt infiltration equation is implemented to consider infiltration into the vadose zone which allows determining the effective

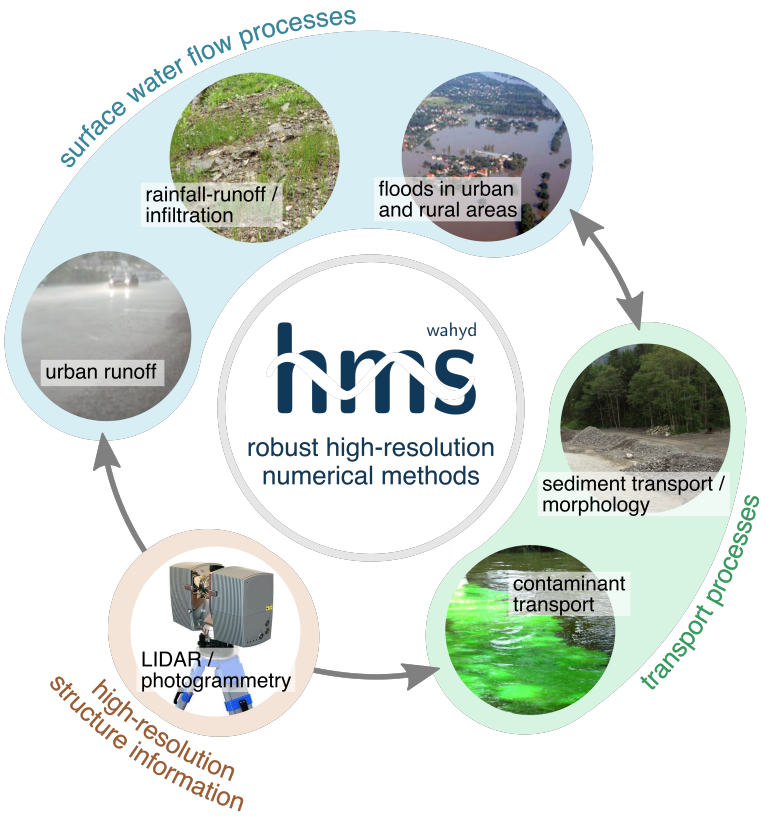


Figure 1.2: Illustration of the objectives of this thesis

rainfall and therefore an event-based simulation of rainfall-runoff scenarios. For the transport of contaminants or sediment the shallow water equations are augmented by additional transport equations and will be considered in the solution of the Riemann problem.

The numerical methods are implemented in the framework of `hms`. By design, `hms` already allows to build multi-process applications. The existing framework is extended by a flexible numerical framework which allows prototyping of numerical algorithms and direct performant coupling of numerical models within the initial design and structure of the `hms` framework.

The developed numerical model targets multi-process applications on the spatial scale of small catchments with an area of up to 1 km^2. Although the focus is not on high-performance simulation, the numerical algorithms are parallelized to allow an

efficient computation on modern multi-core desktop computers and small clusters.

The implementation of the developed numerical model is verified by several analytical test cases and the application is demonstrated in three case studies: flash flood in a simplified urban district, simulation of rainfall-runoff and tracer transport on plot scale and rainfall-runoff simulation for a small alpine catchment. The applications are evaluated by the quality and plausibility of the results and their practicability.

1.5 Outline

The remainder of this thesis is organized as follows:

Chapter 2 covers the mathematical model concepts adopted in this thesis. It explains the governing equations and their mathematical properties.

Chapter 3 deals with the numerical discretization of the governing equations and a robust numerical solution is presented. The used methods are explained in detail.

Chapter 4 starts with an overview on the software concepts of `hms`. Then, the design of the new numerical framework and its prototype implementation are presented. Finally, the usage of the software is explained briefly.

Chapter 5 shows the verification of the numerics and the software concept. Several test cases are carried out to prove the accuracy of the numerical schemes and the implemented software.

Chapter 6 presents three case studies to demonstrate the capabilities and limitations of the developed numerical model concept.

Chapter 7 summarizes the thesis and evaluates the findings and results based on the objectives mentioned before. Development potentials are shown and suggestions for future research are given.

1.6 Electronic supplemental material

The full source code of `hms` is available at `https://gitlab.tu-berlin.de/hms`. Access to the GIT repository is granted by the Chair of Water Resources Management and Modeling of Hydrosystems, Technische Universität Berlin, Germany.

In the electronic supplemental material for this thesis, the numerical model setups for all test cases and applications presented in the Chapters 5 and 6 are supplied.

1.7 Notation

In general, the following notation is used in this thesis. The following mathematical symbols are used in the Chapters 2 and 3: Bold printed letters denote vectors. Here, $\mathbf{q}$ is always the vector of conserved state variables and $\mathbf{f}$ and $\mathbf{g}$ denote numerical fluxes in x and y direction. Regular printed letters denote scalar values. An upright subscript describes a specific type of value, e. g. ν_t is the turbulent viscosity. In general, an italic subscript denote the spatial location or component of a variable. For example, $\mathbf{q}_{i+1}$ is the vector of state variables of the cell $i+1$ and n_x is the x component of the normal vector $\mathbf{n}$. Italic superscripts denote the time level, i. e. c_i^{n+1} is the concentration of cell i at the new time level $n+1$. In general, greek letters denote constant values as for example density ρ or the Von Kármán constant κ.

All symbols are explained when they are used the first time. In addition, in the nomenclature on page 190 a list of all repeatedly used symbols and abbreviations is given.

In the Chapters 4 and 4.3 the Unified Modeling Language (UML) is used to represent relationships between classes and to give details on class implementations (cf. Wikipedia, 2017*b*). In the text, class and method names will be printed in monospaced font (e. g. `HNumericsEngine`).

2 Mathematical model concepts

In this chapter the mathematical model concepts for all physical processes considered in this thesis will be given. All equations will be given in the form of a general conservation law which will be described in Section 2.1. This general form is the basis for the software design of the generic finite volume solver described in Chapter 4.

2.1 General form of the conservation law

The general form of the conservation law for an infinitesimal two-dimensional fixed Eulerian control volume can be expressed as:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} = \mathbf{s} \tag{2.1}$$

where $\mathbf{q}$ is the vector of conserved state variables, $\mathbf{f}$ and $\mathbf{g}$ denote the vectors of advective and diffusive fluxes in x and y direction and $\mathbf{s}$ is the vector of sources and sinks. Thus, a temporal change in the conserved variables can be caused only by a net flux over the surface of the control volume or sources and sinks inside the control volume.

In the following sections, flow and transport processes and runoff generation can be regarded in the general form by defining a specific set of the vectors $\mathbf{q}$, $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{s}$.

2.2 Flow processes in surface water

2.2.1 Shallow water equations

The Reynolds-averaged Navier-Stokes (RANS) equations describe the time-averaged turbulent flow of a compressible Newtonian fluid in three-dimensional space. As

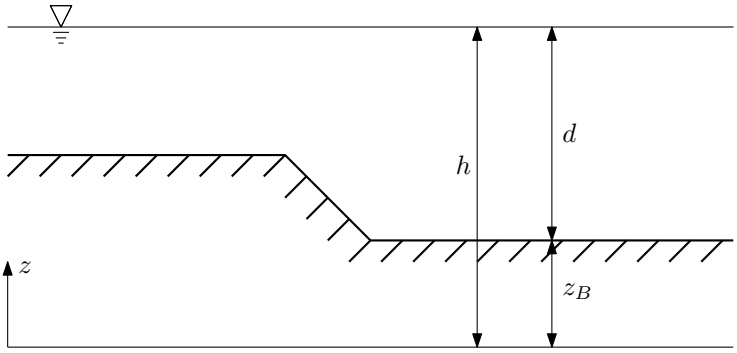


Figure 2.1: Definition of water depth d, bottom elevation z_B and water elevation h

water can be assumed as incompressible, the density is constant and can be neglected. In the case of shallow surface water flow, i. e. the wave length is much larger than the water depth, a hydrostatic pressure distribution can be assumed. In addition, the vertical flow component is often small compared to the horizontal component and can be neglected. This allows integrating the equations over the water depth. Applying these simplifications to the RANS equations leads to the two-dimensional depth-averaged shallow water equations (SWE). They can be written in the general form (Equation 2.1) by defining the vectors:

$$\mathbf{q} = \begin{bmatrix} d \\ ud \\ vd \end{bmatrix}, \ \mathbf{f} = \begin{bmatrix} ud \\ uud + \frac{1}{2}gd^2 - \nu\frac{\partial ud}{\partial x} \\ uvd - \nu\frac{\partial vd}{\partial x} \end{bmatrix},$$
$$\mathbf{g} = \begin{bmatrix} vd \\ vud - \nu\frac{\partial ud}{\partial y} \\ vvd + \frac{1}{2}gd^2 - \nu\frac{\partial vd}{\partial y} \end{bmatrix}, \ \mathbf{s} = \begin{bmatrix} m_\mathrm{w} \\ \frac{\tau_{\mathrm{B}x}}{\rho} + gd\frac{\partial z_\mathrm{B}}{\partial x} + f_x \\ \frac{\tau_{\mathrm{B}y}}{\rho} + gd\frac{\partial z_\mathrm{B}}{\partial y} + f_y \end{bmatrix} \tag{2.2}$$

In the following these vectors will be explained in detail. The first vector components describe the mass balance equation. In here d is the water depth and u and v are the depth-averaged flow velocities in x and y direction. The terms ud and vd denote the mass fluxes in x and y direction and m_w is a mass source/sink term which can account e. g. incoming water due to precipitation or outgoing water due to infiltration. This is explained more in detail in Section 2.3. Figure 2.1 shows the definition of water depth d, bottom elevation z_B and water elevation $h = z_\mathrm{B} + d$.

The remaining two components of the vectors (Equation 2.2) are the momentum balance equations in x and y direction. In here ud and vd are the specific discharges in x and y direction. The first terms in the flux vectors (uud, vud and uvd, vvd) are the advective fluxes of x and y momentum, respectively. The term $\frac{1}{2}gd^2$ is momentum transport due to pressure differences and the viscous fluxes in x and y direction are denoted by $\nu\partial ud/\partial x$, $\nu\partial ud/\partial y$ and $\nu\partial vd/\partial x$, $\nu\partial vd/\partial y$, respectively. Here, ν is the kinematic viscosity, which is explained in Section 2.2.3. In the source vector $\mathbf{s}$, the first term in the second and third component describes bed friction. Here, $\boldsymbol{\tau}_\mathrm{B}$ is the bed shear stress and is explained in detail in Section 2.2.2. The bed friction term is followed by the bottom slope source term and f_x and f_y which could account for other external forces (e.g. Coriolis force or wind drag).

2.2.2 Bed friction

The influence of friction plays an important role in shallow water simulation. Especially for very small water depth, a correct representation of friction is difficult. The simplest approach to describe bed friction is given by the Chézy[1, 2] friction law:

$$\frac{\boldsymbol{\tau}_\mathrm{B}}{\rho} = \frac{g}{C^2}\mathbf{v}|\mathbf{v}| \tag{2.3}$$

where $\boldsymbol{\tau}_\mathrm{B}$ is the bed shear stress and ρ is the density of water.

In the original Chézy law, the coefficient C is seen as a constant value. This allows only an insufficient description of friction as the bed shear stress will be independent of the water depth. However, other well-known friction laws can be seen as extension to the original Chézy law by defining C as function.

Using the friction law of Nikuradse (1933) the coefficient C will be given by:

$$C = 18\lg\left(\frac{12d}{k_\mathrm{s}}\right) \tag{2.4}$$

where k_s is the equivalent sand grain roughness.

Using the approach of Gauckler-Manning-Strickler[3], the Chézy coefficient can be

[1] In the remainder of this thesis, footnotes are used to identify well-known scientists for whom no specific contributions are cited.
[2] A. Chézy (1718 – 1798)
[3] R. Manning (1816 – 1897), P. G. Gauckler (1826 – 1905) and A. Strickler (1887 – 1963)

written as:

$$C = \frac{d^{1/6}}{n} \tag{2.5}$$

In here n is the Manning roughness coefficient, which is related to the Strickler roughness coefficient k_{Str} by:

$$k_{\mathrm{Str}} = \frac{1}{n} \tag{2.6}$$

The Darcy-Weißbach approach was originally developed for pipe flow, but can be used as an approximation in open channel flow, too. Writing as an extension of the Chézy coefficient, it states:

$$C = \sqrt{\frac{8g}{\lambda}} \tag{2.7}$$

In here the roughness coefficient λ by Colebrook and White is a function of bed roughness and the Reynolds number (Colebrook, 1939).

To represent the effect of vegetation Jain et al. (2004) proposed a water-depth-dependent friction coefficient. Using this approach, the Chézy coefficient is expressed as:

$$C = \frac{d^{1/6}}{n_0} \left(\frac{d}{d_0} \right)^{\varepsilon} \tag{2.8}$$

where n_0 is the minimum land-surface-dependent Manning roughness corresponding to water depth d_0, beyond which n is assumed to be constant, and ε is a coefficient related to vegetation drag (Jain et al., 2004). In Figure 2.2 the classical approach of Gauckler-Manning-Strickler is compared with the water-depth-dependent approach for different values of ε. Below water depth d_0 the Chézy coefficient computed by Equation 2.8 decreases more rapidly than in the classical approach, i. e. the friction increases stronger with decreasing water depth. The coefficient ε influences the decreasing slope.

2.2.3 Turbulence

Turbulence is a three-dimensional anisotropic process. In a mathematical model only a simplified description is possible. By time-averaging the Navier-Stokes equations

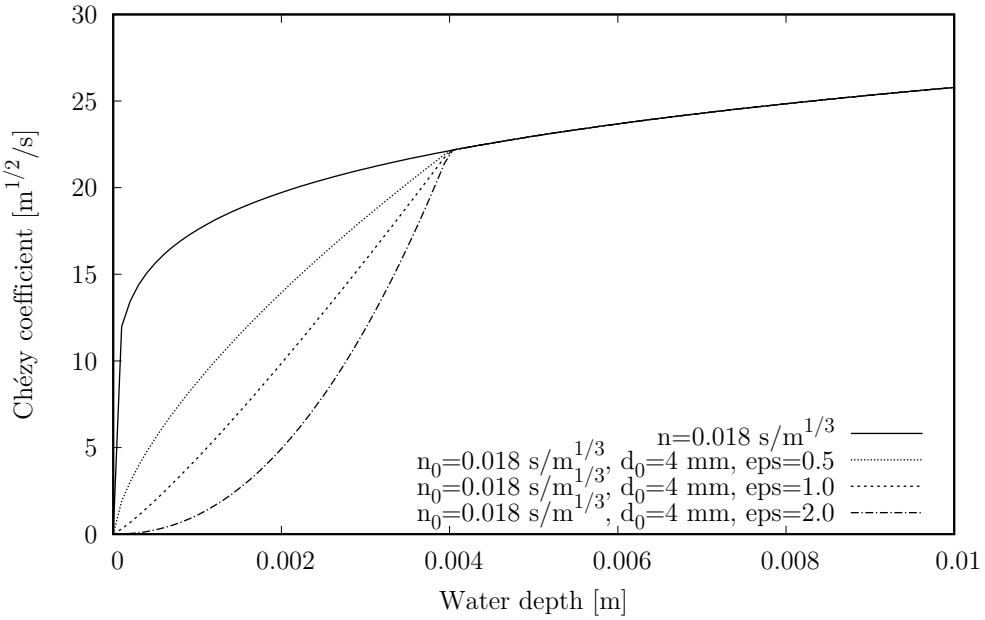


Figure 2.2: Comparison of classical friction law of Gauckler-Manning-Strickler and water-depth-dependent approach of Jain et al. (2004)

the RANS equations are obtained. The averaging process results in new terms, the so-called Reynolds stresses and a closure problem arises. To get a closed set of equations a turbulence model is necessary.

In the concept of eddy viscosity proposed by Boussinesq[4] it is assumed that the Reynolds stresses are proportional to the velocity gradient in the mean flow. Following this, the turbulent stresses are expressed analogous to the viscous stresses. The viscosity ν in Equation 2.2 is therefore the sum of eddy viscosity ν_t and molecular viscosity ν_{mol}:

$$\nu = \nu_t + \nu_{mol} \tag{2.9}$$

As the eddy viscosity is usually much higher than the molecular viscosity of the fluid, the later one is neglected. However, ν_t is not a property of the fluid, but strongly dependent on the local turbulence structure. Turbulence models based on the concept of eddy viscosity describe ν_t as a function of the flow field. They differ in their complexity, accuracy and the computational demand. Cea et al. (2007)

[4] J. V. Boussinesq (1842 – 1929)

compared several turbulence models for depth-averaged simulations. The authors showed that for overland flow with extremely small water depth, turbulence is mainly produced by bed friction and complex turbulence models give the same eddy viscosity as simple ones.

In the presented numerical model two approaches to describe turbulence are followed. In the first approach the eddy viscosity is assumed to be constant:

$$\nu_t = \text{const.} \tag{2.10}$$

This assumption strongly simplifies the process of turbulence.

The second approach belongs to the so-called algebraic turbulence models which relate the eddy viscosity to the mean velocity gradient of the flow by an algebraic expression. Here, Prandtl's mixing length model is used (Prandtl, 1925). The derivation of the depth-averaged turbulence model is given by Malcherek (2003). The depth-averaged eddy viscosity is expressed as:

$$\nu_t = \underbrace{\frac{1}{6}\kappa u_* d}_{\text{vertical}} + \underbrace{\frac{1}{40}\frac{\kappa^3 d^3}{u_*}\left[2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right]}_{\text{horizontal}} \tag{2.11}$$

where the first term expresses the vertical and the second term expresses the horizontal part. The von Kármán constant is given by $\kappa = 0.41$ and the friction velocity u_* is given as:

$$u_* = \sqrt{\frac{|\boldsymbol{\tau}_B|}{\rho}} = \sqrt{\frac{g}{C^2}|\mathbf{v}|^2} \tag{2.12}$$

Both presented approaches to consider the influence of turbulence on the flow field are strong simplifications of the real processes. To study the flow field in detail, e. g. in the near-field of hydraulic structures, a more sophisticated turbulence model would be necessary. However, as a depth-averaged numerical shallow water model already simplifies the three-dimensional structure of flow, the presented approaches allow an efficient consideration of turbulence. This applies especially to large-scale applications.

2.2.4 Further sources/sinks

Wind can act as a source or sink on the flow field. For large water bodies as lakes and reservoirs it has to be taken into account. Several approaches exist, to take the effect of wind into account. As the focus of this thesis is mostly on surface runoff, no details will be given. The implementation of the wind influence in the presented numerical model and its application on a reservoir in Brazil is described by Matta et al. (2014).

2.2.5 Simplifications of the shallow water equations

In the following two important simplifications of the shallow water equations should be mentioned. As the mass balance equation will stay the same as in Equation 2.2, these simplifications will be shown by means of the momentum balance in x direction:

$$\underbrace{\frac{\partial ud}{\partial t} + \frac{\partial uud}{\partial x} + \frac{\partial vud}{\partial y} + \underbrace{\frac{\partial}{\partial x}\left(\frac{1}{2}gd^2\right) + \underbrace{\frac{g}{C^2}u|\mathbf{v}| + gd\frac{\partial z_B}{\partial x}}_{\text{kinematic wave}} = 0}_{\text{diffusive wave}}}_{\text{dynamic wave}} \tag{2.13}$$

Using the full momentum equation is referred to as dynamic wave equation (viscous fluxes and external forces are commonly neglected in this consideration). By neglecting the temporal change of momentum and the advective flux terms, the diffusive wave equation is obtained. If further the pressure gradient is neglected we get the kinematic wave equation. In this case, the momentum equation reduces to the balance of bottom slope source term and bed friction which refers to uniform flow conditions.

2.2.6 Mathematical properties

The shallow water equations (Equations 2.1 and 2.2) are a system of nonlinear partial differential equations. To further analyze the mathematical properties of the equations, the eigenvalues can be determined. Therefore source terms and diffusive

fluxes will be neglected. Transforming Equation 2.1 in a quasi-linear form yields:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial x} + \frac{\partial \mathbf{g}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial y} = 0 \tag{2.14}$$

where the Jacobian matrices are defined as (Leveque, 2002):

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}} = \begin{bmatrix} 0 & 1 & 0 \\ -u^2 + gd & 2u & 0 \\ -uv & v & u \end{bmatrix}, \quad \frac{\partial \mathbf{g}}{\partial \mathbf{q}} = \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + gd & 0 & 2v \end{bmatrix} \tag{2.15}$$

Determining the eigenvalues and -vectors of the Jacobian matrices allows classifying the system of partial differential equations as elliptic, parabolic or hyperbolic. As shown by Leveque (2002) we get three real eigenvalues λ and three linearly independent right eigenvectors $\mathbf{r}$ for each 3×3 Jacobian matrix and therefore the system of partial differential equations can be classified as hyperbolic:

$$\begin{aligned} \lambda^{x1} &= u - c_\mathrm{w}, \quad \lambda^{x2} = u, \quad \lambda^{x3} = u + c_\mathrm{w}, \\ \mathbf{r}^{x1} &= \begin{bmatrix} 1 \\ u - c_\mathrm{w} \\ v \end{bmatrix}, \quad \mathbf{r}^{x2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{r}^{x3} = \begin{bmatrix} 1 \\ u + c_\mathrm{w} \\ v \end{bmatrix} \end{aligned} \tag{2.16}$$

$$\begin{aligned} \lambda^{y1} &= v - c_\mathrm{w}, \quad \lambda^{y2} = v, \quad \lambda^{y3} = v + c_\mathrm{w}, \\ \mathbf{r}^{y1} &= \begin{bmatrix} 1 \\ u \\ v - c_\mathrm{w} \end{bmatrix}, \quad \mathbf{r}^{y2} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{r}^{y3} = \begin{bmatrix} 1 \\ u \\ v + c_\mathrm{w} \end{bmatrix} \end{aligned} \tag{2.17}$$

Here, $c_\mathrm{w} = \sqrt{gd}$ is the speed of gravity waves in shallow water.

Hyperbolic equations always have a wavelike solution (Leveque, 2002). From each point in the state space defined by $\mathbf{q}$, the solution to Equation 2.14 is made up by three waves which travel with wave speeds given by the eigenvalues. The direction of traveling is defined by the eigenvector. While the 1- and 3-wave eigenvector ($\mathbf{r}^{x1}$, $\mathbf{r}^{x3}$ and $\mathbf{r}^{y1}$, $\mathbf{r}^{y3}$) are nonlinear and represent the tangent at the start point, the 2-wave eigenvector ($\mathbf{r}^{x2}$ and $\mathbf{r}^{y2}$) is linearly degenerated and represents a line in

the state space. The latter is called a contact discontinuity or shear wave and will be illustrated in Section 2.4.2. In addition, the wave patterns will be explained in Chapter 3.

2.3 Runoff generation and infiltration

In a rainfall event not all water will be available for surface runoff. The amount of water which is available for runoff, the so-called rainfall excess or effective rainfall, depends on various factors (e. g. precipitation, soil, land use, topography etc.). In the following the runoff generation in natural and urban areas will be discussed.

2.3.1 Water balance in natural and urban areas

The water balance in natural and urban areas consists of several components. Figure 2.3 shows an overview of the components in natural and urban areas. Not all of them have the same importance in all areas. While in urban areas the most important components are precipitation, surface runoff and evaporation, in natural areas many other components contribute to the runoff generation.

There are storage components as interception, which is the amount of precipitation retained on the leaf surfaces, and the soil moisture, which is the amount of water stored in the soil. Further, water is stored in snow and ice and in the groundwater table. Water is also stored in small depressions on the surface, this is called depression storage.

Other components are processes which describe the flow of water within the domain. Precipitation is the source of water, it describes the movement of water from the clouds to the ground surface as rain, snow, hail etc. Evapotranspiration denotes the movement of water into the clouds by evaporation of water stored on the surface of vegetation and land and by transpiration, which describes the process of vapor passed to the environment by plants. The flow of water from the land surface into the upper soil layers, i. e. the vadose zone, is denoted as infiltration. The water percolates deeper into the saturated zone and contributes to the groundwater recharge. From there, water can exfiltrate into surface water bodies. If there are less porous soil layers parallel to the surface, the water flows in the upper soil layers following the topography of the domain. This flow is called interflow.

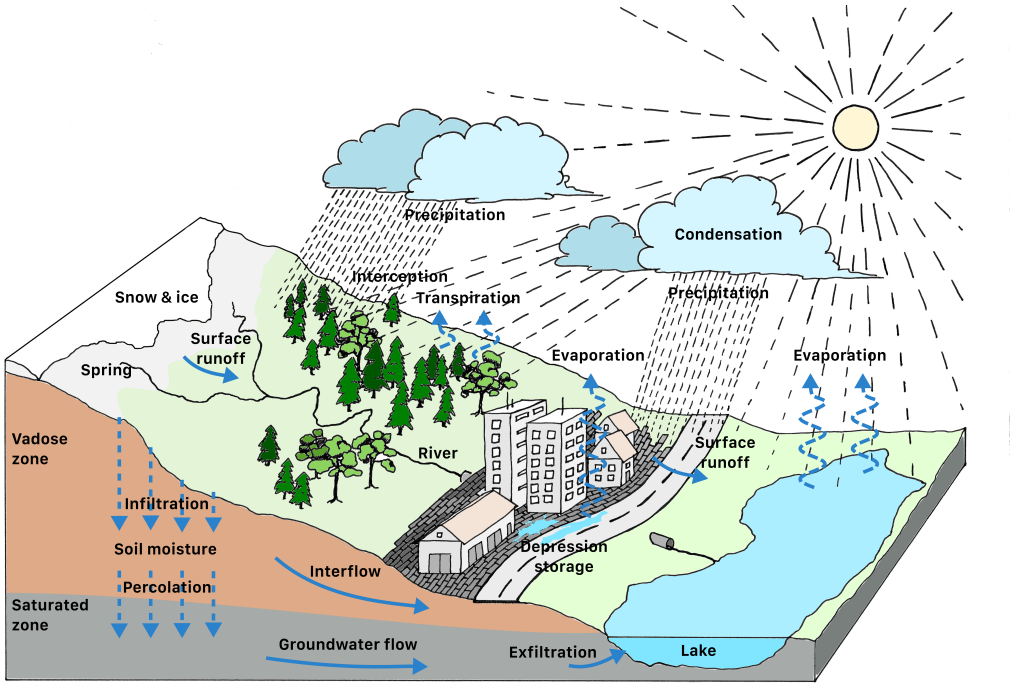


Figure 2.3: Hydrological cycle in natural and urban areas

If the rainfall intensity exceeds the infiltration rate of the soil, or if the soil is fully saturated and the rainfall event continues, the water will gather on the surface and fill furrows and other depressions and, finally, will start running off the surface. This is called infiltration excess or *Hortonian* runoff, when the rainfall intensity is higher than the potential infiltration rate of the soil, or saturation excess or *Dunne* runoff, when the soil is fully saturated due to the rising groundwater table and no more water can be stored (Fiedler and Ramirez, 2000; Weill et al., 2009).

In natural areas, surface runoff is observed only for very heavy or long-term rainfall events. Due to the high degree of sealed surfaces in urbanized areas infiltration is less significant and surface runoff already occurs for short-term rainfall events.

The amount of water held in the depressions and the upper soil layer is not only important for determining the amount of runoff, but is is also important for evaporation. Especially in urban areas the water held in depressions is crucial for the climate of the city (Gessner et al., 2014; Nehls et al., 2015).

It is obvious that at least a simple consideration of runoff generation is necessary to simulate surface runoff in natural areas. In the numerical model presented in this thesis only the processes are considered which are important in the simulation of short-term events. The most important one is the infiltration of water into the soil and the storage of water in the depression storage. On the other hand interception, snow, ice and evapotranspiration will be neglected. This is a good approximation for the temperate zone and natural areas, where the amount of evapotranspiration is very small compared to the other components in the considered time interval. In warm regions and especially in urban areas the evapotranspiration can be also important for short rainfall events and should be considered.

In a classical runoff generation model, the depression storage is often considered by constant parameters which depend on the surface properties and are difficult to determine. In a numerical model based on the shallow water equations which considers high-resolution surface topography, these losses are considered automatically as they are determined by the physics of surface water flow. That means, if the water depth and the momentum of the fluid is not high enough to overtop a depression, the water will stay in that place. Therefore, no separate consideration of the depression storage is necessary.

2.3.2 Mathematical model concept

A mathematical description of the effective rainfall is given by:

$$\begin{aligned}\text{Effective rainfall} = \text{Total rainfall} - \text{Evapotranspiration} \\ - \text{Changes in storage} - \text{Infiltration}\end{aligned} \tag{2.18}$$

After neglecting the evapotranspiration and the changes in storages as interception, snow and ice, the mass source/sink term in the shallow water equations (Equation 2.2) equals the residual from infiltration and precipitation and can be expressed as:

$$m_{\mathrm{w}} = i - r \tag{2.19}$$

Here, we are following the mathematical rule of negative fluxes entering the balance volume and positive fluxes are leaving the balance volume. The total rainfall is denoted as r and the infiltration rate is denoted as i. In addition, using the general conservation law (Equation 2.1), the conservation of the specific soil water volume V_{w} can be expressed as:

$$\mathbf{q} = [V_{\mathrm{w}}]\,,\ \mathbf{f} = [0]\,,\ \mathbf{g} = [0]\,,\ \mathbf{s} = [i] \tag{2.20}$$

It can be seen, that the shallow water equations with Equation 2.19 as source/sink term and the conservation equation of the specific soil water volume are coupled via the infiltration rate.

2.3.3 Infiltration in the vadose zone

To describe water flow in soils, a conservation law for an infinitesimal fixed Eulerian control volume in the vertical soil column can be written:

$$\frac{\partial \theta}{\partial t} + \frac{\partial f_w}{\partial z} = s \tag{2.21}$$

where θ is the volumetric water content, f_w the flux of water in the soil matrix and s a source/sink term which could e. g. denote exfiltration in deeper soil layers or root water uptake. The flux of water in the soil matrix can be described by the Buckingham-Darcy equation (Radcliffe and Simunek, 2010), which is a modified

version of Darcy's law for unsaturated flow:

$$f_w = K(h)\left(\frac{\partial h}{\partial z} + 1\right) \tag{2.22}$$

In here, $K(h)$ is the unsaturated hydraulic conductivity as a function of negative pressure head h. Substituting Equation 2.22 in Equation 2.21 yields the one-dimensional Richards equation (Richards, 1931):

$$\frac{\partial \theta}{\partial t} + \frac{\partial}{\partial z}\left(K(h)\frac{\partial h}{\partial z}\right) + \frac{\partial K(h)}{\partial z} = s \tag{2.23}$$

In here, the second term on the left side accounts for the effect of capillarity on water movement and the third term on the left side accounts for the effect of gravity on water movement (Radcliffe and Simunek, 2010). With increasing saturation of the soil matrix, the vertical gradient of the pressure head vanishes and only the effect of gravity is left.

Equation 2.23 describes the water content in a vertical unsaturated soil column, but can easily be extended to two- or three-dimensional domains. Evaluating the Buckingham-Darcy equation (Equation 2.22) at the soil surface, $z = 0$ will give the infiltration into the soil surface (Radcliffe and Simunek, 2010):

$$i(t) = K(h)\left(\frac{\partial h(z,t)}{\partial z} + 1\right)\Big|_{z=0} \tag{2.24}$$

For surface runoff applications, the infiltration is the most important process and the actual water content in the soil matrix is of less importance. To avoid solving the Richards equation, several simplified infiltration equations have been developed (Green and Ampt, 1911; Horton, 1941; Philip, 1957; Parlange et al., 1985).

Green-Ampt equation

In this thesis the infiltration equation of Green and Ampt (1911) is implemented. In the derivation of the equation the wetting front is assumed as a sharp front, i. e. the transition between the initial and saturated water content occurs abruptly (Figure 2.4). Using this assumption, the Buckingham-Darcy infiltration equation (Equation

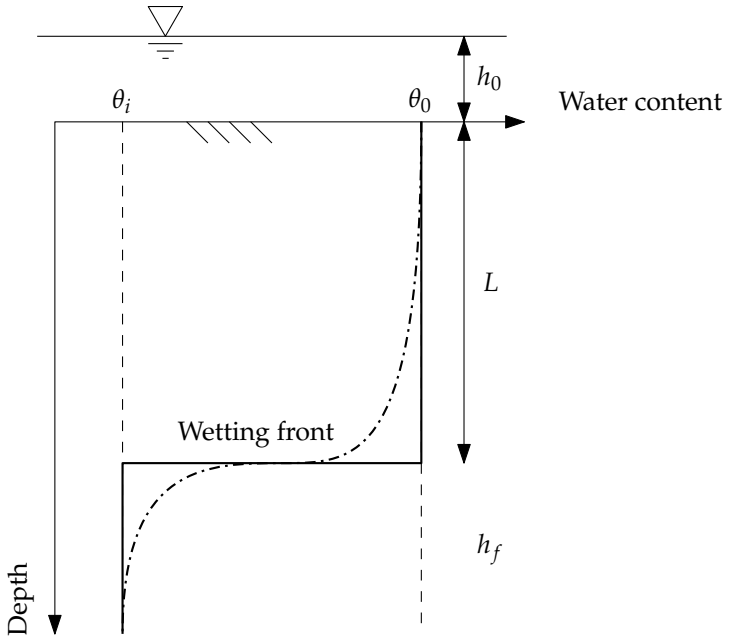


Figure 2.4: Actual (dash dotted line) and approximated wetting front (solid line)

2.24) can be written as:

$$i(t) = K(h_0)\left(\frac{h_0 - h_f}{L} + 1\right) \tag{2.25}$$

where h_0 and h_f are the pressure head at the soil surface and the wetting front, respectively. L is the distance from the soil surface to the wetting front (cf. Figure 2.4) which will increase over time . A second equation for the infiltration rate is introduced, which states, that the infiltration rate is equal to the change of the water volume behind the advancing wetting front:

$$i(t) = \frac{\mathrm{d}}{\mathrm{d}t}\left[(\theta_0 - \theta_i)\, L\right] \tag{2.26}$$

In here, θ_0 is the water content at the surface and and θ_i is the initial water content of the soil.

Rewriting and setting Equation 2.25 equal to Equation 2.26 yields:

$$K(h_0)\,\frac{h_0 - h_f + L}{L} = (\theta_0 - \theta_i)\,\frac{\mathrm{d}L}{\mathrm{d}t} \tag{2.27}$$

Separating variables and integrating the wetting front distance from 0 to L and time from 0 to t:

$$\int_0^L \frac{L}{h_0 - h_f + L}\,\mathrm{d}t = \int_0^t \frac{K\left(h_0\right)}{\theta_0 - \theta_i}\,\mathrm{d}t \tag{2.28}$$

yields (Radcliffe and Simunek, 2010):

$$\left(\theta_0 - \theta_i\right) L - \left(\theta_0 - \theta_i\right)\left(h_0 - h_f\right) \ln\left(1 + \frac{\left(\theta_0 - \theta_i\right) L}{\left(\theta_0 - \theta_i\right)\left(h_0 - h_f\right)}\right) = K\left(h_0\right) t \tag{2.29}$$

Using the definition for the cumulative infiltration $I\left(t\right) = \left(\theta_0 - \theta_i\right) L$, Equation 2.29 can be rewritten as:

$$I\left(t\right) = K\left(h_0\right) t + \left(\theta_0 - \theta_i\right)\left(h_0 - h_f\right) \ln\left(1 + \frac{I\left(t\right)}{\left(\theta_0 - \theta_i\right)\left(h_0 - h_f\right)}\right) \tag{2.30}$$

which is the Green-Ampt equation for cumulative infiltration.

As Equation 2.30 cannot be solved explicitly for $I\left(t\right)$, Newton's method is used in this thesis to determine for the cumulative infiltration. To obtain the infiltration rate, a simple first-order approximation for the temporal derivative is used:

$$i\left(t\right) = \frac{\mathrm{d}I\left(t\right)}{\mathrm{d}t} = \frac{I^{n+1} - I^n}{\Delta t} \tag{2.31}$$

where I^{n+1} and I^n are the cumulative infiltration from the current and previous time step, respectively.

Equation 2.31 determines the potential infiltration rate, i. e. the infiltration rate which would be possible, if there is an unlimited amount of water available for infiltration. Before updating the soil water volume using Equation 2.1 and 2.20, it is limited to the maximum water volume that is available for infiltration:

$$i\left(t\right) = \min\left(i\left(t\right), {}^{d}/_{\Delta t} + r\right) \tag{2.32}$$

where d is the depth of ponding surface water.

In addition, a residual rate is computed as a source for the shallow water equations using Equation 2.19. If $m_\mathrm{w} < 0$, the rainfall intensity exceeds the infiltration rate and water is ponding at the surface, and if $m_\mathrm{w} > 0$, the infiltration rate is higher than

the rainfall intensity. This way, the presented model describes so-called infiltration excess or Hortonian runoff. For the simulation of saturation excess or Dunne runoff, a description of groundwater flow and exfiltration is necessary. This is beyond the scope of this thesis.

2.4 Transport processes in surface water

2.4.1 Depth-averaged advection-diffusion equation

The transport of a single component is described by the depth-averaged advection-diffusion equation. Seen as an extension to Equation 2.1, the vectors $\mathbf{q}$, $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{s}$ are given as:

$$\mathbf{q} = [cd]\,,\ \mathbf{f} = \left[ucd - D\frac{\partial cd}{\partial x}\right],\ \mathbf{g} = \left[vcd - D\frac{\partial cd}{\partial y}\right],\ \mathbf{s} = [m_\mathrm{c}] \tag{2.33}$$

In here, c is the depth-averaged component concentration, i. e. the ratio of tracer volume to water volume. The first term in the flux vector $\mathbf{f}$ describes the advective flux and the second term describes the diffusive flux. D is the diffusion coefficient, which is analogous to the kinematic viscosity ν in Equation 2.2 the sum of the molecular and the turbulent diffusion coefficient. In case of turbulence, the turbulent diffusion is much higher than the molecular diffusion, and the molecular diffusion can be neglected. The ratio of turbulent viscosity and turbulent diffusion can be expressed by the dimensionless turbulent Schmidt number:

$$\mathrm{Sc_t} = \frac{\nu_\mathrm{t}}{D_\mathrm{t}}$$

Gualtieri et al. (2017) give an comprehensive review on the value of the turbulent Schmidt number at different flow scenarios. In the source vector m_c describes a general source or sink term. This could be for example a constant source or sink term or could be time-dependent, as e. g. a decay term. For multicomponent transport, a conservation law (Equation 2.33) for each component must be formulated and m_c will include more complex terms, e. g. chemical reaction terms which describe the interaction of the components with each other. An example is a numerical water quality model with a transport equation for each chemical component (e.g. nitrogen, phosphorus, dissolved oxygen etc.).

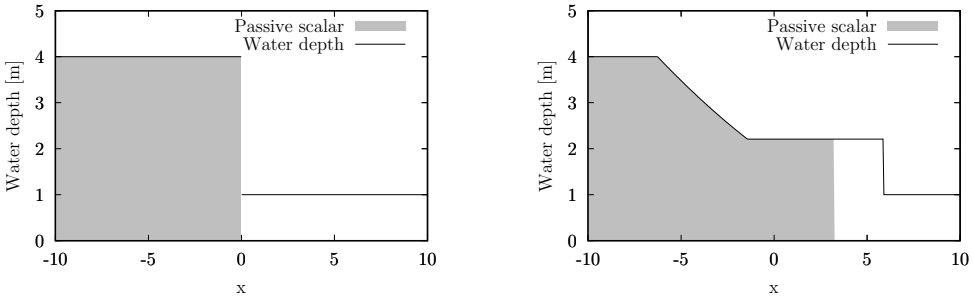
2.4.2 Mathematical properties

To compute transport processes in surface water, it is necessary to couple flow and transport. If the transported component does not influence the flow field at all, which is a good assumption for most chemical components or if the component is a passive tracer, Equation 2.33 just augments the system of shallow water equations. We still get six eigenvalues and six corresponding eigenvectors:

$$\begin{aligned} \lambda^{x1} &= u - c_{\mathrm{w}}, \quad \lambda^{x2} = u, \quad \lambda^{x3} = u + c_{\mathrm{w}}, \\ \mathbf{r}^{x1} &= \begin{bmatrix} 1 \\ u - c_{\mathrm{w}} \\ v \\ c \end{bmatrix}, \quad \mathbf{r}^{x2} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{r}^{x3} = \begin{bmatrix} 1 \\ u + c_{\mathrm{w}} \\ v \\ c \end{bmatrix} \end{aligned} \tag{2.34}$$

$$\begin{aligned} \lambda^{y1} &= v - c_{\mathrm{w}}, \quad \lambda^{y2} = v, \quad \lambda^{y3} = v + c_{\mathrm{w}}, \\ \mathbf{r}^{y1} &= \begin{bmatrix} 1 \\ u \\ v - c_{\mathrm{w}} \\ c \end{bmatrix}, \quad \mathbf{r}^{y2} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{r}^{y3} = \begin{bmatrix} 1 \\ u \\ v + c_{\mathrm{w}} \\ c \end{bmatrix} \end{aligned} \tag{2.35}$$

The tracer concentration c is continuous across the 1- and 3-waves. Across the 2-wave or contact discontinuity, the water depth and the momentum in flow direction are constant. However, there is a jump in the momentum perpendicular to the observed direction and in the tracer concentration which are both transported by the 2-wave. For the tracer transport, the contact discontinuity marks the boundary between water that was initially on the left and water that was initially on the right. Figure 2.5 illustrates this situation. Initially, there is a jump in the water depth at $x = 0$. To the left and right of the jump the water depth is constant and the initial flow velocity is zero. The water on the left site is colored, the water on the right site is clear. In the temporal evolution of this situation, there is a rarefaction wave going to the left and a shock wave going to right. Across both, the water color remains constant. A third wave traveling to right marks the boundary between the colored and the clear water. It is called contact discontinuity as the fluid is in contact with itself (i. e. water depth and momentum in flow direction remain constant), but there



(a) Initial conditions (b) After 1 s

Figure 2.5: Development of a contact discontinuity

is a discontinuity in the transported matter, here the color, which has no influence on the flow dynamics.

2.5 Sediment transport and morphological evolution

Using the general conservation law, many processes can be expressed. This could be for example transport of heat. As a proof of the software concept presented in Chapter 4, sediment transport and morphological evolution are implemented as additional processes (Section 5.6). In the following a brief description of the mathematical model concept for these processes is given. However, as this thesis only touches upon this topic, there is no detailed discussion.

Sediment transport is a crucial process associated with surface runoff. Sediments are silting up fresh water reservoirs and the erosion driven by the water current can cause scour and loss of fertile soil.

In a capacity or equilibrium model, equilibrium of erosion and deposition is assumed at each point in time (Wu, 2007). To compute the temporal evolution of the moving bed, a conservation equation for bed sediment, the so-called Exner equation (Exner, 1925), and a formula for the equilibrium bed-load transport rate are used. Several formulas have been determined by laboratory and field experiments, e. g. Meyer-Peter and Müller (1948). Especially in dynamic cases, where there is locally strong entrainment of bed material, the assumption of equilibrium is inappropriate (Cao et al., 2004).

Non-capacity models do not assume equilibrium of sediment entrainment and deposition. Both processes are described using empirical equations. To compute the transport of the eroded material, a transport equation for the suspended load is introduced. As it may be difficult to distinguish between bed load and suspended load transport, one possibility is to not separate both processes and then the mathematical model will reduce to a total-load transport equation and a bed-load mass-balance equation. Both are coupled via empirical formulas for erosion and deposition.

The governing equations are presented by Simpson and Castelltort (2006) and can be written in the general form (Equation 2.1) by defining the vectors:

$$\mathbf{q} = \begin{bmatrix} cd \\ z_\mathrm{B} \end{bmatrix}, \ \mathbf{f} = \begin{bmatrix} ucd \\ 0 \end{bmatrix}, \ \mathbf{g} = \begin{bmatrix} vcd \\ 0 \end{bmatrix}, \ \mathbf{s} = \begin{bmatrix} D - E \\ \frac{E-D}{1-\phi} \end{bmatrix} \tag{2.36}$$

In here, c is the depth-averaged sediment concentration, E and D are the sediment entrainment and deposition flux, respectively. ϕ is the bed sediment porosity.

In addition the shallow water equations (Equation 2.2) have to be complemented to account for the water-sediment mixture. The mass balance equation is extended by the mass exchange between flow and erodible bed:

$$m_\mathrm{w} = \frac{D - E}{1 - \phi} \tag{2.37}$$

The momentum balance equations are complemented by the following source/sink terms, accounting for the spatial variations in the sediment concentration and momentum transfer due to sediment exchange between the flow and the erodible bed (Cao et al., 2004; Simpson and Castelltort, 2006):

$$f_x = \frac{1}{2}\frac{\rho_\mathrm{s} - \rho_w}{\rho} g \cdot h^2 \frac{\partial c}{\partial x} + \frac{\rho_0 - \rho}{\rho}\frac{E - D}{1 - \phi} u \tag{2.38}$$

$$f_y = \frac{1}{2}\frac{\rho_\mathrm{s} - \rho_w}{\rho} g \cdot h^2 \frac{\partial c}{\partial y} + \frac{\rho_0 - \rho}{\rho}\frac{E - D}{1 - \phi} v \tag{2.39}$$

where ρ_w and ρ_s are the densities of water and sediment. The density of the water-sediment mixture ρ is expressed as:

$$\rho = \rho_w (1 - c) + \rho_\mathrm{s} \cdot c \tag{2.40}$$

and the density of the saturated bed ρ_0 is expressed by:

$$\rho_0 = \rho_w \cdot \phi + \rho_s (1 - \phi) \tag{2.41}$$

To close the governing equations the entrainment flux E and the deposition flux D are needed. Several empirical relations exist (Cao and Carling, 2002). The entrainment flux of *non-cohesive* sediment is computed by (Cao et al., 2004):

$$E = \frac{160}{R^{0.8}} \frac{1 - \phi}{\theta_c} \frac{(\theta - \theta_c)\, d_G \cdot U_\infty}{d} \tag{2.42}$$

where $R = d_G \sqrt{s \cdot g \cdot d_G}/\nu$, $s = \rho_s/\rho_w - 1$, d_G is the grain diameter, ν is the kinematic viscosity of water and U_∞ is the free surface velocity which can be approximated from the depth-averaged velocity using an exponential velocity profile:

$$U_\infty = \frac{7|\mathbf{v}|}{6} \tag{2.43}$$

The Shields[5] parameter θ is computed by:

$$\theta = \frac{u_*}{s \cdot g \cdot d_G} \tag{2.44}$$

where u_* is the friction velocity expressed by Equation 2.12. Sediment movement is initiated if the critical value θ_c is reached, i. e. on the other hand $E = 0$ for $\theta \leqslant \theta_c$.

The entrainment flux for *cohesive* material can be computed by (Izumi and Parker, 2000 cited in Simpson and Castelltort, 2006):

$$E = \beta \cdot \left(\frac{|\mathbf{v}|}{u_c - 1} \right)^\gamma \tag{2.45}$$

Here, β is an entrainment coefficient, γ is an exponent and u_c is a threshold velocity for the initiation of sediment movement, i. e. $E = 0$ for $|\mathbf{v}| \leqslant u_c$.

For the deposition flux the following relation can be used (Cao et al., 2004):

$$D = \omega \cdot (1 - c_a)^m c_a \tag{2.46}$$

where m is an exponent and c_a is the near-bed sediment concentration which is

[5] A. F. Shields (1908 – 1974)

presumed to be proportional to the depth-averaged concentration $c_a = \alpha \cdot c$ with $\alpha = \min\left[2, (1-\theta)/c\right]$. The settling velocity of a single particle in tranquil water ω is computed by (Cheng, 1997):

$$\omega = \frac{\nu}{d_G}\left[\left(25 + 1.2 \cdot d_*^2\right)^{0.5} - 5\right]^{1.5} \tag{2.47}$$

with:

$$d_* = d_G\left(\frac{\rho_s}{\rho_w - 1}\frac{g}{\nu^2}\right)^{1/3} \tag{2.48}$$

2.6 Conclusions

In this chapter the mathematical model concepts for all physical processes regarded in this thesis were given. All equations were given in the form of a general conservation law (Section 2.1). This generalization allows to abstract the mathematical models and is the basis for the software design of the generic finite volume solver described in Chapter 4. In addition the mathematical properties of the governing equations were given, which determine the numerical solution process presented in Chapter 3. In the description of the runoff generation and sediment transport model, the coupling of the models by source/sink terms was shown.

Using the general conservation law, additional processes can be expressed. This could be for example transport of heat.

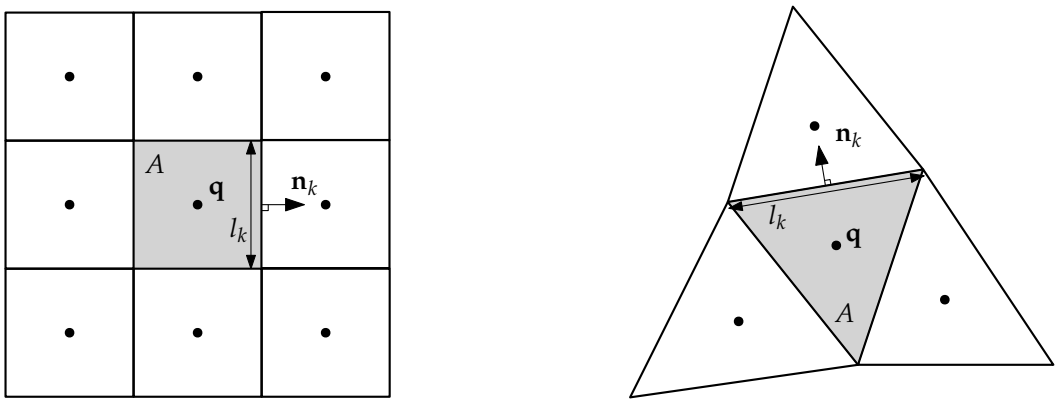
3 Robust numerical methods for hydrodynamic simulation

From numerical point of view, solving surface water flow and transport processes is a demanding task. An analytical solution of the governing equations is only possible for strongly simplified cases. As it will be described later in detail, small water depths, wetting and drying, flow transitions (e. g. hydraulic jumps), strongly varying bottom elevation and friction influence put strong requirements on a robust numerical method.

The term *robust* is used in literature to describe a numerical solution method which can handle arbitrary complex applications and stays stable, i. e. it provides a reasonable solution (e. g. Zhou et al., 2001; Jiwen and Ruxun, 2001; Krámer and Józsa, 2007; Nikolos and Delis, 2009; Song et al., 2011). In contrast, the term *accurate* describes a numerical scheme which allows accurate solution of a given problem, it does not imply the stability of the numerical scheme. However, obtaining an accurate solution for a complex application requires a robust numerical method.

Different criteria exist to evaluate the robustness of numerical methods. First of all the conservation of a conservative variable has to be satisfied. Second, a scheme should stay stable, even for complex applications. This means, the simulation should not stop due to numerical errors. A scheme which handles wetting and drying processes correctly, has to be *well-balanced* (e. g. Audusse and Bristeau, 2005; Canestrelli et al., 2009; Hou, Liang, Simons and Hinkelmann, 2013). That means, no artificial movement is allowed for water which which is ponding on an irregular surface (lake at rest). The so-called *C-property* denotes the ability of a numerical scheme to preserve a motionless water surface on an irregular bed at steady-state conditions (e. g. Liang and Marche, 2009; Kesserwani, 2013). A scheme fulfilling the C-property is well-balanced. This is achieved by a well-balanced formulation of the pressure flux and the slope source term.

In this chapter, the numerical solution of the partial differential equations presented in Chapter 2 is shown. First, the discretization will be explained. It will become



(a) Regular structured grid

(b) Unstructured triangular mesh

Figure 3.1: Spatial discretization and cell notation

apparent, that the flux calculation can be seen as solving a Riemann problem. The exact and approximated solution to the Riemann problem is shown. To improve the accuracy, the numerical method is extended to second-order accuracy in space and the approach for a robust solution is given.

3.1 Discretization method

The general conservation law (Equation 2.1) is given as a partial differential equation. For a numerical solution the derivatives have to be discretized. Hereby, it can be distinguished between the discretization of the temporal and the spatial derivatives. The most popular methods for spatial discretization are the finite difference method (FDM), the finite volume method (FVM) and the finite element method (FEM) (Hinkelmann, 2005).

3.1.1 Cell-centered finite volume method

Using the cell-centered finite volume method (CCFVM), the domain of interest is subdivided in control volumes (cells) of finite size. This can be for example done by a regular structured grid as seen in Figure 3.1a or an unstructured triangular mesh as seen in Figure 3.1b. All state variables $\mathbf{q}$ are computed at the midpoint of the cell and are representative for the whole cell. At the same time, all spatial

information which is smaller than the cell size gets lost. In the context of surface runoff, this means, that the flow in very small rills and furrows is not resolved spatially but approximated by a sheetlike flow. As a consequence, the cell size is crucial for the accuracy of a simulation and must be determined by analyzing the size of representative structures.

In comparison to the finite difference method, the CCFVM guarantees local and therefore global conservation of the equations (Hinkelmann, 2005).

In the first step of the derivation of the CCFVM, the general conservation law (Equation 2.1) is integrated over the domain of the control volume Ω:

$$\int_\Omega \frac{\partial \mathbf{q}}{\partial t} \, \mathrm{d}\Omega + \int_\Omega \left(\frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} \right) \mathrm{d}\Omega = \int_\Omega \mathbf{s} \, \mathrm{d}\Omega \tag{3.1}$$

Using the Green-Gauss Integral theorem, the volume integral over the flux term can be transformed into a surface integral. This can be explained by the fact, that a change in the flux inside the volume equals the difference of the fluxes over the surface of the volume Γ. Equation 3.1 can now be written as:

$$\int_\Omega \frac{\partial \mathbf{q}}{\partial t} \, \mathrm{d}\Omega + \oint_\Gamma \mathbf{F}\,\mathbf{n} \, \mathrm{d}\Gamma = \int_\Omega \mathbf{s} \, \mathrm{d}\Omega \tag{3.2}$$

For a two-dimensional cell, Ω equals the cell area and Γ equals the length of the edges of the cell. The vector $\mathbf{n} = \left(n_x, n_y\right)^{\mathrm{T}}$ is an outward pointing unit vector normal to the surface of the cell. The flux vector $\mathbf{F}$ normal to the surface is given as:

$$\mathbf{F}\,\mathbf{n} = \mathbf{f} n_x + \mathbf{g} n_y \tag{3.3}$$

Integrating the surface integral for an arbitrary cell in two-dimensional space with n_b edges gives:

$$\oint_\Gamma \mathbf{F}\,\mathbf{n} \, \mathrm{d}\Gamma = \sum_{k=1}^{n_b} \mathbf{F}_k \mathbf{n}_k l_k \tag{3.4}$$

in which k is the index of an edge of the considered cell and l_k is the edge length.

As the temporal change of the state variable $\frac{\partial \mathbf{q}}{\partial t}$ and the source $\mathbf{s}$ are assumed to be constant inside the volume, Equation 3.2 can now be written in a semi-discrete form

as:

$$\frac{\partial \mathbf{q}}{\partial t} A + \sum_{k=1}^{n_b} \mathbf{F}_k \mathbf{n}_k l_k = \mathbf{s} A \tag{3.5}$$

where A is the cell area.

3.1.2 Temporal discretization

The simplest approach to discretize the temporal term of Equation 3.5 is a first-order difference approximation, which can be derived by a Taylor series expansion and skipping the higher-order terms (Hinkelmann, 2005):

$$\frac{\partial \mathbf{q}}{\partial t} \approx \frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t} \tag{3.6}$$

In here, the indices $n+1$ and n denote the values of the new and the old time level, respectively. Substituting this approximation (Equation 3.6) in the discretized general conservation law (Equation 3.5) yields:

$$\mathbf{q}^{n+1} = \mathbf{q}^n - \frac{\Delta t}{A} \sum_{k=1}^{n_b} \mathbf{F}_k^n \mathbf{n}_k l_k - \Delta t\, \mathbf{s}^n \tag{3.7}$$

Equation 3.7 is the simplest explicit method, the so-called *forward Euler method*. For the flux and source terms, the values of the old time level are used and, as a result, the state variable on the new time level is depending only on variables of the old time level, which characterizes explicit methods. As there is no need for solving a system of equations, the computational effort per time step is low. However, explicit methods have a restriction on the time step size. The second approach is given by the *backward Euler method*:

$$\mathbf{q}^{n+1} = \mathbf{q}^n - \frac{\Delta t}{A} \sum_{k=1}^{n_b} \mathbf{F}_k^{n+1} \mathbf{n}_k l_k - \Delta t\, \mathbf{s}^{n+1} \tag{3.8}$$

In here, the flux and source terms are evaluated on the new time level. As a consequence, Equation 3.8 cannot be solved directly for a cell, but has to be solved as a system of equations in a fully implicit way. The computational effort per time step is therefore high. However, as the solution of a cell is always dependent on the solution of the other cells, this method shows an improved stability. In addition

there are no constraints on the time step size. An implicit time discretization of the shallow water equations is used e. g. by Stelling and Duinmeijer (2003). Both, the forward and backward Euler methods are first-order accurate.

The improved Euler method tries to combine the advantage of a simple solution of an explicit method and the stability of the implicit method. It is a second-order accurate approximation for the temporal term. However, as it is an explicit approach, there is a restriction in the time step size, too. In a first step, a value on the new time level is predicted analogous to Equation 3.7:

$$\mathbf{q}^{n+1,\,\text{predictor}} = \mathbf{q}^{n} - \frac{\Delta t}{A} \sum_{k=1}^{n_b} \mathbf{F}_k^n \mathbf{n}_k l_k - \Delta t\, \mathbf{s}^n \tag{3.9}$$

Using these predicted values, a corrector value is computed in an pseudo-implicit step analogous to Equation 3.8:

$$\mathbf{q}^{n+1,\,\text{corrector}} = \mathbf{q}^{n} - \frac{\Delta t}{A} \sum_{k=1}^{n_b} \mathbf{F}_k^{n+1,\,\text{predictor}} \mathbf{n}_k l_k - \Delta t\, \mathbf{s}^{n+1,\,\text{predictor}} \tag{3.10}$$

The final solution on the new time level is computed by the average of the predictor and corrector value:

$$\mathbf{q}^{n+1} = \frac{1}{2} \left(\mathbf{q}^{n+1,\,\text{predictor}} + \mathbf{q}^{n+1,\,\text{corrector}} \right) \tag{3.11}$$

Stability criterion

To guarantee the stability of explicit time stepping methods, special conditions on the time step size have to be fulfilled. As the value on the new time level is dependent on the value of the old time level of the considered cell and the direct neighbors only, the maximum propagation distance of a disturbance in one time step must be smaller or equal the cell distance. This condition can be expressed by the Courant–Friedrichs–Lewy (CFL) condition:

$$\text{Cr} = \frac{\text{max. propagation distance}}{\text{cell distance}} = \frac{v_{max} \cdot \Delta t}{\Delta s} \leq 1 \tag{3.12}$$

The dimensionless Courant number Cr must be smaller or equal one for a stable simulation. Here, Δs is the characteristic cell distance. The diameter of the incircle is

used, to compute Δs for square and triangular mesh cells:

$$\Delta s = \frac{4 \cdot A}{P} \tag{3.13}$$

where A and P are the cell area and perimeter, respectively. For the shallow water equations (Equation 2.2) the maximum propagation velocity is computed as:

$$v_{max} = |\mathbf{v}| + \sqrt{g \cdot d} \tag{3.14}$$

Due to the variation of water depth and flow velocity, it is difficult to determine the optimal time step size for a simulation. Therefore, for an improved efficiency, the time step size is determined adaptively while the simulation is running. For each computational cell a maximum time step size is computed by rearranging Equation 3.12:

$$\Delta t = \frac{\mathrm{Cr} \cdot \Delta s}{v_{max}} \tag{3.15}$$

The smallest computed time step size in the whole domain determines the time step size used in the discretized conservation law for all cells in the next time step. The CFL condition is a necessary prerequisite for a stable solution. However, it does not guarantee a stable solution, as the stability can be influenced by further factors in the numerical solution process.

The constraints on the time step size of the explicit time stepping methods seem to be a major drawback compared to an implicit method without such constraints. However, Hirsch (2007) states, that the choice of time stepping method depends on the physical time step that the simulation intends to capture: if the physical time step is of the same order as the time step computed from Equation 3.15, explicit methods are the most appropriate choice. Due to the computational overhead for the numerical solution of the system of equations, the implicit time stepping methods have no advantage to the explicit methods for small physical time steps. If the physical time step is significantly higher than the time step necessary for stability, implicit methods are a valid option (Hirsch, 2007). This will be the case for long-term or steady-state problems. In event-based rainfall-runoff applications the physical time step is limited by the accurate representation of the runoff patterns and therefore, an explicit time stepping method is a reasonable choice.

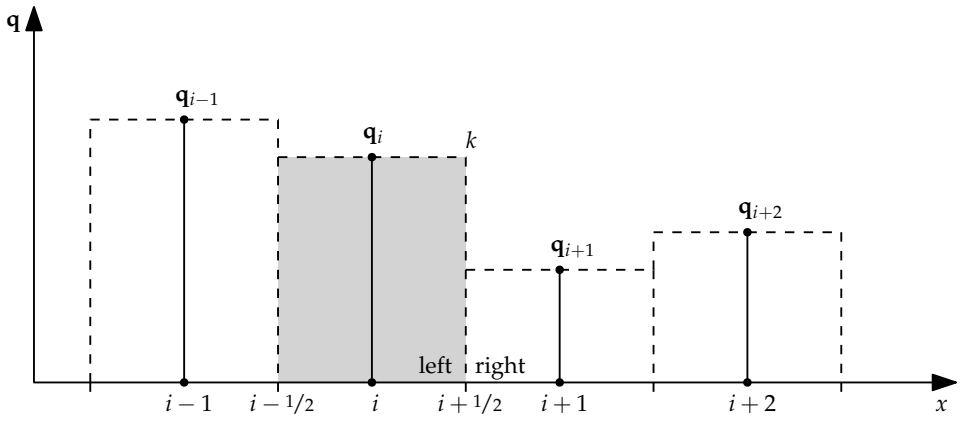


Figure 3.2: First-order reconstruction of face values in one dimension

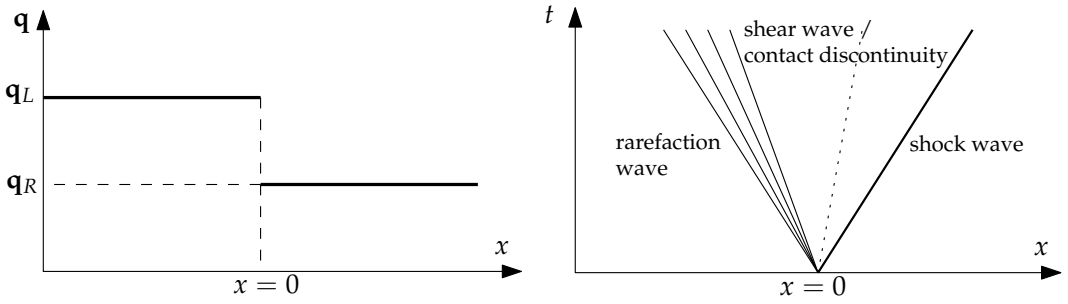
3.2 Godunov's method for advective fluxes

From Equation 3.5 we can see, that the numerical flux $\mathbf{F}_k$ has to be evaluated at the cell face k. In respect to the assumption of the CCFVM, that the midpoint cell value is representative for the whole cell, there is not one cell value assigned to a face, but always a right and left face value (Figure 3.2). Thus, the choice of the value to be used for the numerical flux calculation is not unique. As for hyperbolic problems information propagates as waves along characteristics, a method to overcome this ambiguity is looking in the direction from which the information should be coming (Leveque, 2002).

3.2.1 First-order upwind method

The easiest approach is given by the first-order upwind method (FOU). The advective flux is separated into a transporting and a transported variable, which are e. g. the flow velocity u and the water depth d in case of the flux in x direction for the mass balance equation in the SWE. The value of the transporting variable u at the cell face $k = i + 1/2$ is computed by linear averaging the values of the cell i and $i + 1$. This is:

$$u_{i+1/2} = \frac{u_{i+1} + u_i}{2} \tag{3.16}$$



(a) Initial values

(b) Solution structure in the x–t plane

Figure 3.3: Riemann problem

The value of the transported variable d at the cell face k is then determined by:

$$d_{i+1/2} = \begin{cases} d_i & \text{if } u_{i+1/2} \geq 0 \\ d_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \tag{3.17}$$

The FOU is a robust approach to evaluate the numerical flux at a cell face. However, it introduces strong numerical diffusion and smears sharp gradients in the solution over multiple cells.

Godunov (1959) proposed an algorithm for solving hyperbolic equations together with piecewise constant data. Originally it was developed in the framework of the finite difference method. For the finite volume method the algorithm can be seen as (modified after Leveque, 2002):

1. Reconstruct face values from the average cell values. In the simplest, first-order case, the face value will be the same as the midpoint value.
2. Evolve the hyperbolic equation exactly (or approximately) with this initial data to get the result a time Δt later. This is done by using the theory of Riemann problems for solving the numerical fluxes.
3. By updating the cell values with the sum of the numerical fluxes, new average cell values are obtained.

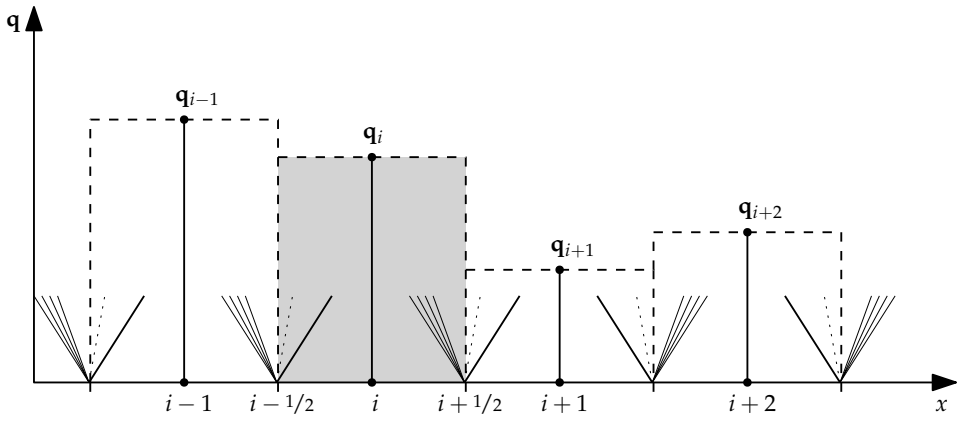


Figure 3.4: Riemann problems at each cell face in FVM for a 1D example

3.2.2 Theory of Riemann problems

A Riemann[1] problem consists of a hyperbolic conservation law and constant initial data with a single discontinuity given as (cf. Figure 3.3a):

$$\mathbf{q}(x, t_0) = \begin{cases} \mathbf{q}_L & \text{if} \quad x < 0, \\ \mathbf{q}_R & \text{if} \quad x > 0. \end{cases} \tag{3.18}$$

The solution of the Riemann problem results in a wave structure which describes the propagation of the discontinuity. As shown in Section 2.2.6, the solution of the SWE is made up by three waves. Depending on the flow conditions the left and right wave will be a shock or rarefaction wave (e. g. Figure 3.3b). The middle wave will be a contact discontinuity or shear wave, across which the water depth and the momentum in flow direction are constant and a jump in the momentum perpendicular to the observed direction occurs.

In the framework of the FVM a discontinuity and therefore a local Riemann problem exists at each cell face due to the difference in the left and right face values. Figure 3.2 shows four cells with a piecewise constant discretization. At the edge $k = i + 1/2$, there is a left and right cell value. The determination of the flux $\mathbf{F}_k$ over this edge can therefore be broken down to the solution of a local Riemann problem:

$$\mathbf{F}_{i+1/2} = f(\mathbf{q}_i, \mathbf{q}_{i+1}) \tag{3.19}$$

[1] G. F. B. Riemann (1826 – 1866)

The same is true for all other edges, as it can be seen in Figure 3.4. In the following, solution methods for the Riemann problem will be presented.

3.2.3 Exact Riemann solution for the SWE

In the following, the exact solution to the Riemann problem developed by Toro (1992) for the shallow water equations is given. The exact Riemann solver is implemented in `hms` to obtain reference solutions for analytical benchmarks (cf. Section 5.2 in Chapter 5).

As shown, three waves divide the solution structure into four regions. The most left and right region contain the unchanged initial states. Across the 1- and 3-wave (left and right wave) the water depth and normal flow velocity change, the tangential flow velocity and tracer concentration stay constant. In the region between the 1- and 3-wave, the so-called star region, a new constant water depth and normal flow velocity exist. It is divided by the 2-wave (middle wave), across which the tangential flow velocity and tracer concentration will change discontinuously.

To find the unknown water depth and flow velocity in the star region, two functions are derived which connect the state of the star region to the left and right state, respectively. The complete derivation is given by Toro (1992).

Three important relations are used in the derivation of the connecting functions:

Riemann invariants: Riemann invariants are functions whose values stay constant (invariant) across rarefaction and shear waves. They can be found by solving the ordinary differential equation:

$$\frac{\mathrm{d}q_1}{r_1^i} = \frac{\mathrm{d}q_2}{r_2^i} = \frac{\mathrm{d}q_3}{r_3^i} \tag{3.20}$$

where i is the number of the eigenvector (cf. Equations 2.16 and 2.17) and the suffixes 1 to 3 denote the vector component of the vector of conserved state variables $\mathbf{q}$ and the eigenvectors $\mathbf{r}$, respectively.

Two Riemann invariants can be found for each eigenvector. For the 1-, 2- and 3-wave in x direction these are:

1-wave:

$$u - 2\sqrt{gd} = constant, \qquad v = constant \tag{3.21}$$

2-wave:

$$u = constant, \qquad d = constant \tag{3.22}$$

3-wave:

$$u + 2\sqrt{gd} = constant, \qquad v = constant \tag{3.23}$$

The Riemann variants for the waves in y direction can be written analogously.

Rankine-Hugoniot jump condition: For a shock wave traveling with speed S, the Rankine-Hugoniot[2] jump condition states:

$$\mathbf{f}_{ahead} - \mathbf{f}_{behind} = S\left(\mathbf{q}_{ahead} - \mathbf{q}_{behind}\right) \tag{3.24}$$

$$\mathbf{g}_{ahead} - \mathbf{g}_{behind} = S\left(\mathbf{q}_{ahead} - \mathbf{q}_{behind}\right) \tag{3.25}$$

For example, for a left shock wave in x direction and after transformation we get:

$$d_*\left(u_* - S\right) = d_L\left(u_L - S\right) \tag{3.26}$$

$$u_* d_*\left(u_* - S\right) + \frac{1}{2} g d_*^2 = u_L d_L\left(u_L - S\right) + \frac{1}{2} g d_L^2 \tag{3.27}$$

$$d_* v_*\left(u_* - S\right) = d_L v_L\left(u_L - S\right) \tag{3.28}$$

From Equations 3.26 and 3.28 follows:

$$v_* = v_L \tag{3.29}$$

which means, the tangential flow velocity will stay constant across the shock wave. The same will be true for the shallow water equations augmented by a transport equation (Equation 2.33), i. e.:

$$c_* = c_L \tag{3.30}$$

Entropy condition: The solutions of the Riemann problem for a system of hyperbolic equations are weak solutions, i. e. they are not necessarily unique (Leveque, 2002). To guarantee a physically correct solution the entropy condition is used. The discontinuity wave traveling with speed S must fulfill:

[2] W. J. M. Rankine (1820 – 1872), P.-H. Hugoniot (1851 – 1887)

$$\lambda^i\left(\mathbf{q}_{behind}\right) > S > \lambda^i\left(\mathbf{q}_{ahead}\right) \tag{3.31}$$

where i is the number of the eigenvalue (cf. Equations 2.16 and 2.17). Equation 3.31 states, that the speed of the discontinuity must be in-between the speeds of the characteristics behind and ahead of the discontinuity computed by the eigenvalue based on the state behind and the state ahead the discontinuity, respectively.

As the left and right wave can be either a shock or rarefaction wave, the derivation of the connecting functions differs depending on the wave type. The middle wave is always a contact discontinuity or shear wave. The wave type of the left and right wave is determined by comparing the left and and right water depth with the water depth in the star region. The resulting functions are:

$$f_L = \begin{cases} 2\left(\sqrt{gd_*} - \sqrt{gd_L}\right) & \text{if } d_* \leq d_L \text{ (left wave is rarefaction wave)} \\ \left(d_* - d_L\right)\sqrt{\frac{1}{2}g\left(\frac{d_* + d_L}{d_* d_L}\right)} & \text{if } d_* > d_L \text{ (left wave is shock wave)} \end{cases} \tag{3.32}$$

$$f_R = \begin{cases} 2\left(\sqrt{gd_*} - \sqrt{gd_R}\right) & \text{if } d_* \leq d_R \text{ (right wave is rarefaction wave)} \\ \left(d_* - d_R\right)\sqrt{\frac{1}{2}g\left(\frac{d_* + d_R}{d_* d_R}\right)} & \text{if } d_* > d_R \text{ (right wave is shock wave)} \end{cases} \tag{3.33}$$

The function for the water depth in the star region combines the left and right connecting functions:

$$f\left(d_*\right) = f_L\left(d_*, d_L\right) + f_R\left(d_*, d_R\right) + u_R - u_L = 0 \tag{3.34}$$

Equation 3.34 can be solved iteratively for the unknown d_* using the Newton-Raphson method. The normal flow velocity is then computed by:

$$u_* = \frac{1}{2}\left(u_L + u_R\right) + \frac{1}{2}\left[f_R\left(d_*, d_R\right) - f_L\left(d_*, d_L\right)\right] \tag{3.35}$$

After determining the water depth and normal flow velocity in the star region, the solution can be sampled at any point in space-time. The solution is a similarity solution, i. e. the solution will be the same for any $x/t = constant$ (Leveque, 2002).

In the following, the solutions to five scenarios will be given:

- One shock and one rarefaction wave
- Two shock waves
- Two rarefaction waves
- Dry-bed rarefaction wave
- Supercritical flow

One shock and one rarefaction wave

In Figure 3.5a the similarity solution of a wet bed Riemann problem is given. The initial water depth on the left side is 4 m and on the right side 1 m. The initial flow velocity is $0\,\mathrm{m/s}$ on both sides. To demonstrate the shear or contact wave, a passive scalar is assumed on the left hand side. By solving Equations 3.34 and 3.35, the solution structure seen in Figure 3.5b can be drawn.

The wave speed of the right shock wave S_R can be computed by:

$$S_R = u_R + c_{\mathrm{w}R} \cdot \sqrt{\frac{1}{2}\frac{(d_* + d_R)\, d_*}{h_R^2}} \tag{3.36}$$

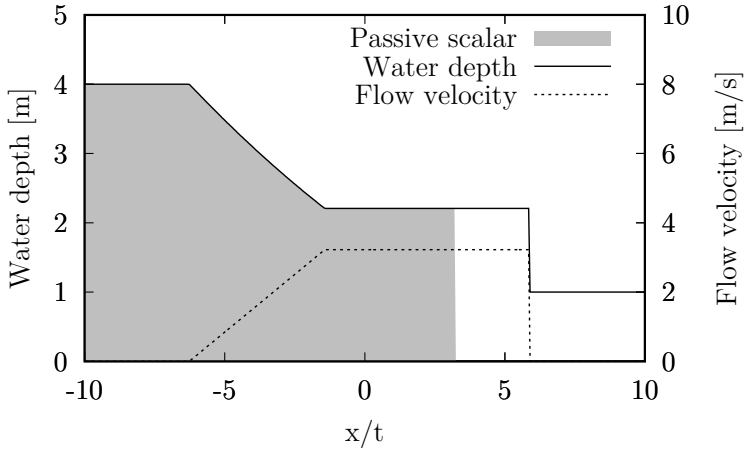
and the head and tail wave speeds of the left rarefaction wave (S_{HL} and S_{TL}) can be computed by:

$$S_{HL} = u_L - c_{\mathrm{w}L} \tag{3.37}$$

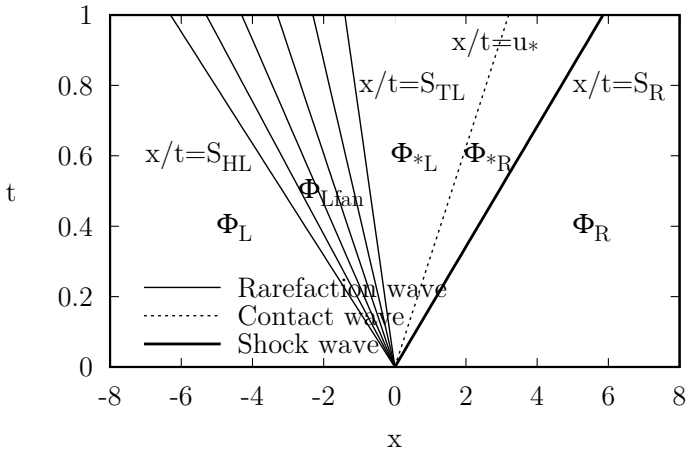
$$S_{TL} = u_* - c_{\mathrm{w}*} \tag{3.38}$$

So, if $x/t \leq S_{HL}$, the vector of primitive unknowns $\boldsymbol{\phi}$ is:

$$\boldsymbol{\phi} = \begin{bmatrix} d \\ u \\ v \\ c \end{bmatrix} = \boldsymbol{\phi}_L = \begin{bmatrix} d_L \\ u_L \\ v_L \\ c_L \end{bmatrix} \tag{3.39}$$



(a) Water depth, flow velocity and transport of passive scalar as a function of x/t



(b) Structure of the solution in the x–t plane

Figure 3.5: Similarity solution of the wet bed Riemann problem for the SWE with initially zero velocity and a high water depth on the left and a low water depth on the right.

If $S_{HL} \leq x/t \leq S_{TL}$, the sampling point lies within the rarefaction fan:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_{Lfan} = \begin{bmatrix} \frac{1}{g}\left[\frac{1}{3}\left(u_L + 2\,c_{\mathrm{w}L} - \frac{x}{t}\right)\right]^2 \\ \frac{1}{3}\left(u_L + 2\,c_{\mathrm{w}L} + 2\,\frac{x}{t}\right) \\ v_L \\ c_L \end{bmatrix} \tag{3.40}$$

The sampling point lies within the left side of the star region, when $S_{TL} \leq x/t \leq u_*$ and so:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_{*L} = \begin{bmatrix} d_* \\ u_* \\ v_L \\ c_L \end{bmatrix} \tag{3.41}$$

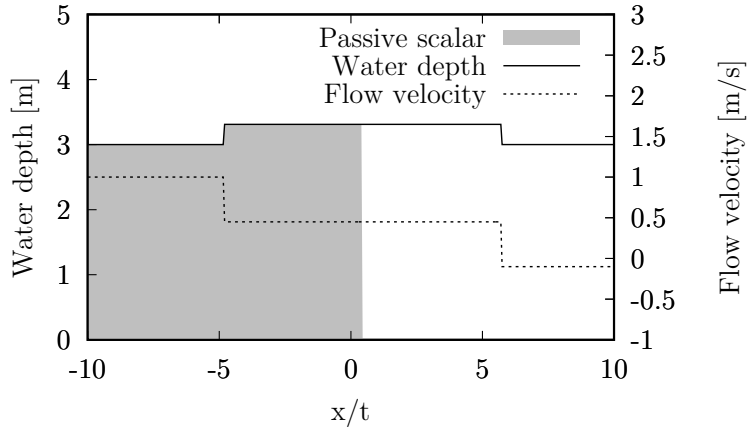
It lies within the right side, when $u_* \leq x/t \leq S_R$ and the vector of unknowns is:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_{*R} = \begin{bmatrix} d_* \\ u_* \\ v_R \\ c_R \end{bmatrix} \tag{3.42}$$

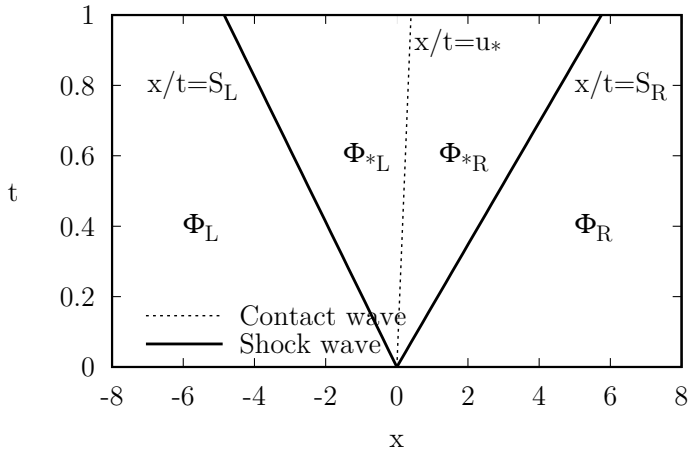
If $S_R \leq x/t$ the sampling point lies to the right of the shock wave:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_R = \begin{bmatrix} d_R \\ u_R \\ v_R \\ c_R \end{bmatrix} \tag{3.43}$$

In Figure 3.5a the water elevation, flow velocity and transport of a passive scalar are given as a function of x/t.



(a) Water depth and flow velocity and transport of passive scalar as a function of x/t



(b) Structure of the solution in the x–t plane

Figure 3.6: Similarity solution of the Riemann problem for the SWE with initially constant water depth and high positive velocity on the left and a low negative velocity on the right

Two shock waves

The Riemann problem given in Figure 3.6 has a constant initial water depth of 3 m, but a high positive flow velocity of 1 m/s on the left and a low negative velocity of −0.1 m/s on the right side. The water on the left-hand side contains a passive tracer. Figure 3.6a shows the similarity solution for the water depth, flow velocity and passive tracer. In Figure 3.6b the solution structure in the x–t plane is given.

The left shock wave speed S_L is computed analogous to Equation 3.36 as:

$$S_L = u_L - c_{\mathrm{w}L} \cdot \sqrt{\frac{1}{2}\frac{(d_* + d_L)\, d_*}{h_L^2}} \tag{3.44}$$

If the sampling point lies in the left or right undisturbed region, i. e. $x/t \leq S_L$ or $S_R \leq x/t$, the solution is given by:

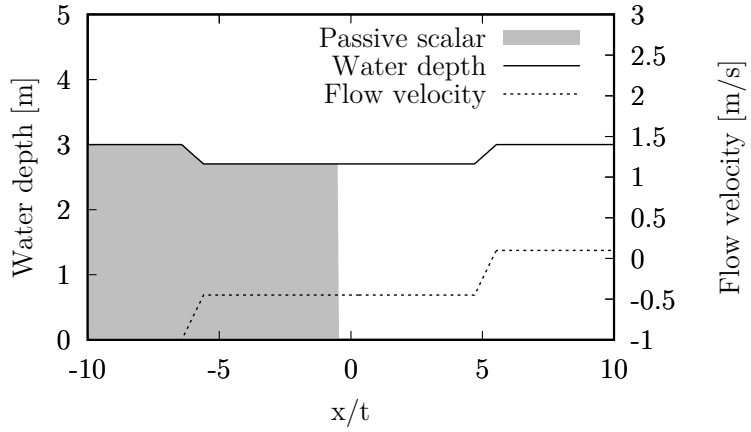
$$\boldsymbol{\phi} = \boldsymbol{\phi}_L = \begin{bmatrix} d_L \\ u_L \\ v_L \\ c_L \end{bmatrix}, \qquad \boldsymbol{\phi} = \boldsymbol{\phi}_R = \begin{bmatrix} d_R \\ u_R \\ v_R \\ c_R \end{bmatrix} \tag{3.45}$$

For $S_L \leq x/t \leq u_*$ and $u_* \leq x/t \leq S_R$, the sampling point lies in the left and right star region, respectively. The solution is given as:

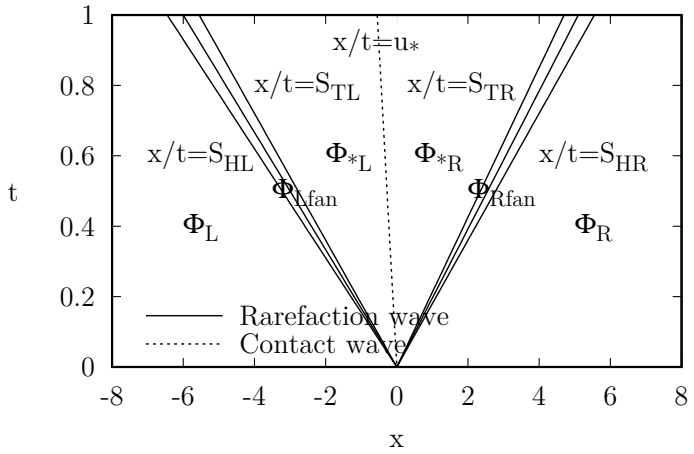
$$\boldsymbol{\phi} = \boldsymbol{\phi}_{*L} = \begin{bmatrix} d_* \\ u_* \\ v_L \\ c_L \end{bmatrix}, \qquad \boldsymbol{\phi} = \boldsymbol{\phi}_{*R} = \begin{bmatrix} d_* \\ u_* \\ v_R \\ c_R \end{bmatrix} \tag{3.46}$$

Two rarefaction waves

As in the previous case, this Riemann problem has 3 m as a constant initial water depth (Figure 3.7). In addition there is a high negative flow velocity of −1 m/s on the left and a low positive flow velocity of 0.1 m/s on the right-hand side. The diverging flow causes two rarefaction waves. The solution structure in the x–t plane is given in Figure 3.7b.



(a) Water depth and flow velocity and transport of passive scalar as a function of x/t



(b) Structure of the solution in the x–t plane

Figure 3.7: Similarity solution of the Riemann problem for the SWE with initially constant water depth and high negative velocity on the left and a low positive velocity on the right

The head and tail wave speed of the right rarefaction wave (S_{HR} and S_{TR}) are computed analogous to the Equations 3.37 and 3.38 as:

$$S_{HR} = u_R + c_{wR} \tag{3.47}$$
$$S_{TR} = u_* + c_{w*} \tag{3.48}$$

The solution for $x/t \leq S_{HL}$ or $S_{HR} \leq x/t$ is given by Equation 3.45. The solution in the star region ($S_{TL} \leq x/t \leq u_*$ and $u_* \leq x/t \leq S_{TR}$) is given by Equation 3.46. If the sampling point lies within the left rarefaction fan ($S_{HL} \leq x/t \leq S_{TL}$) the solution is given by Equation 3.40. Analogous, if $S_{TR} \leq x/t \leq S_{HR}$, the sampling point lies within the right rarefaction fan and the solution is given by:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_{Rfan} = \begin{bmatrix} \frac{1}{g}\left[\frac{1}{3}\left(-u_R + 2\,c_{wR} + \frac{x}{t}\right)\right]^2 \\ \frac{1}{3}\left(u_R - 2\,c_{wR} + 2\,\frac{x}{t}\right) \\ v_R \\ c_R \end{bmatrix} \tag{3.49}$$

Figure 3.7a shows the water depth, flow velocity and transport of a passive scalar as a function of x/t.

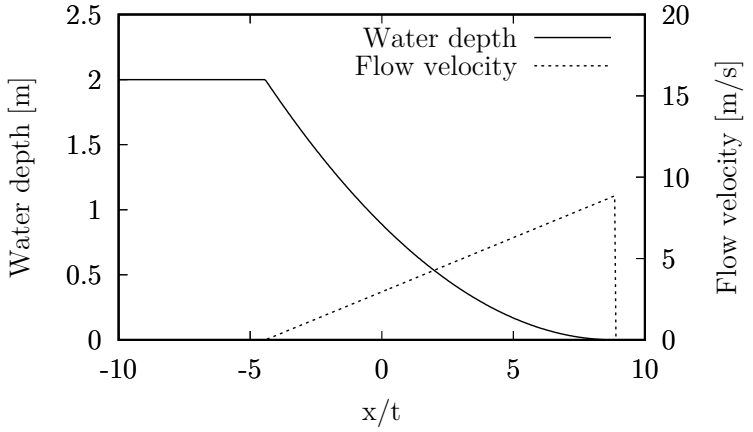
Dry-bed rarefaction wave

Figure 3.8 shows a Riemann problem with an initially dry bed on the right-hand side ($d_R = 0\,\mathrm{m}$). The water depth on the left-hand side is 2 m and the flow velocity is $0\,\mathrm{m/s}$. Figure 3.8b shows the solution structure in the x–t plane. It can be recognized, that there is only a left going rarefaction wave.

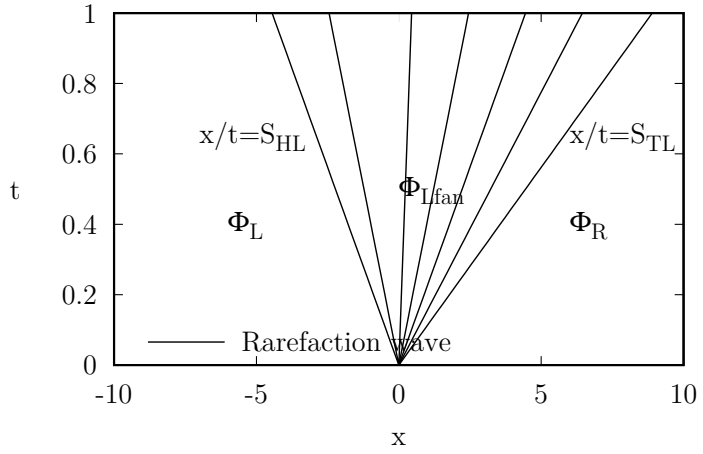
For $x/t \leq S_{HL}$ and $S_{HR} \leq x/t$ the solution is given by Equation 3.45. If the sampling point lies within the rarefaction fan ($S_{HL} \leq x/t \leq S_{TL}$), the solution is given by Equation 3.40.

Supercritical flow

This Riemann problem is similar to the first one. The initial water depth on the left side is 4 m and on the right side 1 m. However, in this case the flow velocity on the

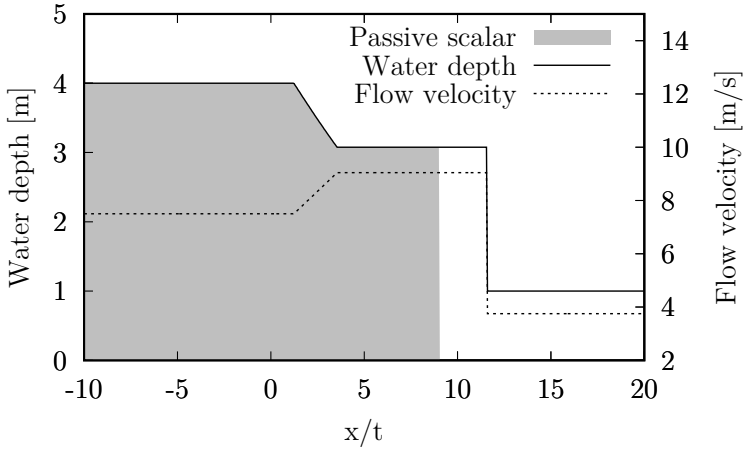


(a) Water depth and flow velocity and transport of passive scalar as a function of x/t

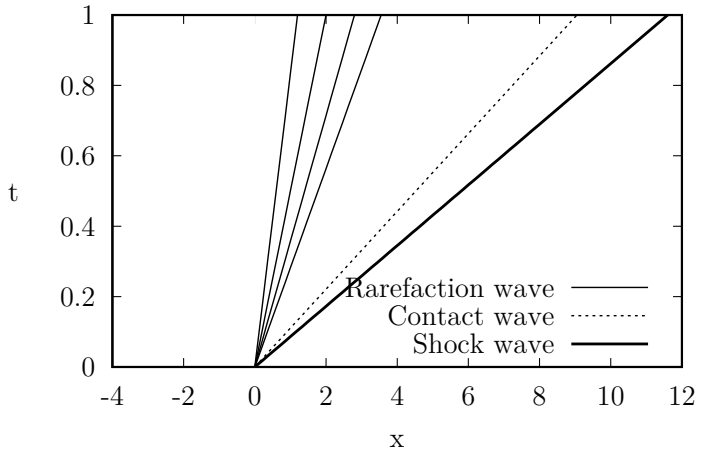


(b) Structure of the solution in the x–t plane

Figure 3.8: Similarity solution of the dry bed Riemann problem for the SWE with initially zero velocity and a water depth of 2 m on the left



(a) Water elevation and flow velocity and transport of passive scalar as a function of x/t



(b) Structure of the solution in the x–t plane

Figure 3.9: Similarity solution of the wet bed Riemann problem for the SWE with supercritical flow velocities and a high water depth on the left and a low water depth on the right

left side is 7.5 m/s and on the right side 3.75 m/s. This will result in supercritical flow conditions on both sides.

The solution procedure is the same as in the first case and the type of waves will be the same. However, as it can be seen in the similarity solution (Figure 3.9a) and the solution structure in the x–t plane (Figure 3.9b) no information at all is propagated to the upstream (left) side. No wave will cross $x = 0$ at any time.

3.2.4 Approximate Riemann solvers for SWE

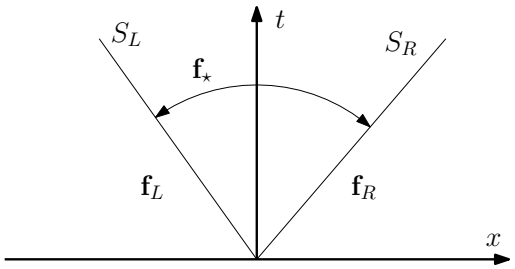
In Godunov's method a local Riemann problem is solved at each cell edge. However, as shown in the previous section the full solution to a Riemann problem is quite expensive. In addition for computing the flux in the finite volume framework only the solution to the Riemann problem at the sampling point $x/t = 0$ is necessary. Harten et al. (1983) proposed to use an approximation of the solution to the Riemann problem. In a *Godunov-type* method usually an approximate Riemann solver is used to obtain the cell edge values. A review on Godunov-type methods is given by Toro and Garcia-Navarro (2007).

In this thesis the widely used *Harten-Lax-van Leer-Contact* (HLLC) Riemann solver was implemented and will be described in detail.

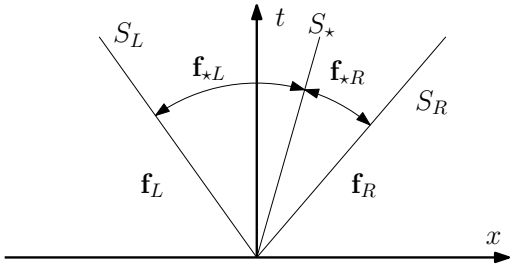
Harten-Lax-van Leer-Contact Riemann solver

In the original *Harten-Lax-van Leer* (HLL) Riemann solver invented by Harten, Lax and van Leer (1983), the solution of the Riemann problem is reduced to three constant states separated by two waves (Figure 3.10a). For the one-dimensional shallow water equations without additional transport this will give a good approximation. However, for the two-dimensional equations and transport processes the solver gives inaccurate results, as the middle wave which is crucial for the transport of the orthogonal flow velocity component and passive scalar is neglected. As a consequence, high numerical diffusion will distort contact discontinuities and shear waves (Toro, 2001).

Toro et al. (1994) developed a modified version of the solver, the so-called *Harten-Lax-van Leer-Contact* (HLLC) Riemann solver. They introduced a third wave which splits the intermediate region in two constant states (Figure 3.10b). The flux at the



(a) HLL solver



(b) HLLC solver

Figure 3.10: Approximated solution structure of HLL and HLLC solver

edge k will be computed as:

$$\mathbf{f}_k = \begin{cases} \mathbf{f}_L & \text{if } 0 \leq S_L \\ \mathbf{f}_{*L} & \text{if } S_L \leq 0 \leq S_* \\ \mathbf{f}_{*R} & \text{if } S_* \leq 0 \leq S_R \\ \mathbf{f}_R & \text{if } S_R \leq 0 \end{cases} \tag{3.50}$$

That means, if the solution is determined by the left or the right region, the flux $\mathbf{f}_k$ is solely computed by the states of the left or right cell, respectively:

$$\mathbf{f}_L = \begin{bmatrix} u_L d_L \\ u_L u_L d_L + \frac{1}{2} g d_L^2 \\ u_L v_L d_L \\ u_L d_L c_L \end{bmatrix}, \quad \mathbf{f}_R = \begin{bmatrix} u_R d_R \\ u_R u_R d_R + \frac{1}{2} g d_R^2 \\ u_R v_R d_R \\ u_R d_R c_R \end{bmatrix} \tag{3.51}$$

The star region is separated into a left and right part by the contact discontinuity or

shear wave and the fluxes will be computed as:

$$\mathbf{f}_{*L} = \begin{bmatrix} f_{1*} \\ f_{2*} \\ f_{1*} v_L \\ f_{1*} c_L \end{bmatrix}, \quad \mathbf{f}_{*R} = \begin{bmatrix} f_{1*} \\ f_{2*} \\ f_{1*} v_R \\ f_{1*} c_R \end{bmatrix} \tag{3.52}$$

where f_{1*} and f_{2*} are the first and second component of the HLL flux, respectively, which is computed as:

$$\mathbf{f}_* = \frac{S_R \mathbf{f}_L - S_L \mathbf{f}_R + S_L S_R (\mathbf{q}_R - \mathbf{q}_L)}{S_R - S_L} \tag{3.53}$$

The wave speeds S_L and S_R are estimations for the exact wave speeds. Using a two-rarefaction approximation, they can be computed as (Liang et al., 2004):

$$S_L = \begin{cases} u_R - 2\sqrt{g d_R} & \text{if } d_L = 0 \\ \min\left(u_L - c_{\mathrm{w}L}, u_* - c_{\mathrm{w}*}\right) & \text{if } d_L > 0 \end{cases} \tag{3.54}$$

$$S_R = \begin{cases} u_L + 2\sqrt{g d_L} & \text{if } d_R = 0 \\ \max\left(u_R - c_{\mathrm{w}R}, u_* - c_{\mathrm{w}*}\right) & \text{if } d_R > 0 \end{cases} \tag{3.55}$$

Here, $c_{\mathrm{w}*}$ is computed using an estimation for the water depth in the star region (cf. Equation 3.34):

$$d_* = \frac{1}{g}\left[\frac{1}{2}\left(c_{\mathrm{w}L} + c_{\mathrm{w}R}\right) + \frac{1}{4}\left(u_L - u_R\right)\right]^2 \tag{3.56}$$

The normal flow velocity in the star region (cf. Equation 3.35) is computed as:

$$u_* = \frac{1}{2}\left(u_L + u_R\right) + c_{\mathrm{w}L} - c_{\mathrm{w}R} \tag{3.57}$$

In the exact Riemann solver, the middle wave speed equals u_*. In the HLLC solver, the middle wave speed S_* is estimated by:

$$S_* = \frac{S_L d_R \left(u_R - S_R\right) - S_R d_L \left(u_L - S_L\right)}{d_R \left(u_R - S_R\right) - d_L \left(u_L S_L\right)} \tag{3.58}$$

which is suitable for dry bed problems, too (Toro, 2001).

3.2.5 2D Riemann solution

So far, all derivations were shown for the flux $\mathbf{f}$ in x direction on a structured mesh. To implement the methods for two-dimensional unstructured meshes, a one-dimensional Riemann problem is constructed at a cell edge by applying a rotational matrix to the flow velocity vector:

$$\begin{bmatrix} v_\perp \\ v_\parallel \end{bmatrix} = \begin{bmatrix} n_x & n_y \\ -n_y & n_x \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} n_x u + n_y v \\ -n_y u + n_x v \end{bmatrix} \tag{3.59}$$

This will result in a flow velocity $v_\perp$ normal to the edge and a flow velocity $v_\parallel$ tangential to the edge. Now, the normal flow velocity is used to compute the wave speeds. The resulting normal and tangential momentum fluxes are converted back to the momentum fluxes in x and y direction using:

$$\mathbf{F}\,\mathbf{n} = \mathbf{T}^{-1}\mathbf{f} \tag{3.60}$$

Using the inverse of the rotational matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & n_x & n_y & 0 \\ 0 & -n_y & n_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & n_x & -n_y & 0 \\ 0 & n_y & n_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.61}$$

results in:

$$\mathbf{F}\,\mathbf{n} = \begin{bmatrix} f_1 \\ n_x f_2 - n_y f_3 \\ n_y f_2 + n_x f_3 \\ f_4 \end{bmatrix} \tag{3.62}$$

The flux $\mathbf{f}$ and its components in Equation 3.60 and 3.62, respectively, are computed using the rotated flow velocities (Equation 3.59).

As an example, using Equations 3.62 and 3.51, the left region HLLC flux for an

arbitrary edge states:

$$\mathbf{F}_L = \begin{bmatrix} v_{\perp L} d_L \\ n_x \left(v_{\perp L} v_{\perp L} d_L + \frac{1}{2} g d_L^2 \right) - n_y \left(v_{\perp L} v_{\parallel L} d_L \right) \\ n_y \left(v_{\perp L} v_{\perp L} d_L + \frac{1}{2} g d_L^2 \right) + n_x \left(v_{\perp L} v_{\parallel L} d_L \right) \\ v_{\perp L} d_L c_L \end{bmatrix} \tag{3.63}$$

The flux in the left star region is:

$$\mathbf{F}_{*L} = \begin{bmatrix} f_{*1} \\ n_x f_{*2} - n_y f_{*1} v_{\parallel L} \\ n_y f_{*2} + n_x f_{*1} v_{\parallel L} \\ f_{1*} c_L \end{bmatrix} \tag{3.64}$$

In full detail Equation 3.64 is written as:

$$\mathbf{F}_{*L} = \begin{bmatrix} \frac{S_R(v_{\perp L} d_L) - S_L(v_{\perp R} d_R) + S_L S_R (d_R - d_L)}{S_R - S_L} \\ n_x \frac{S_R\left(v_{\perp L} v_{\perp L} d_L + \frac{1}{2} g d_L^2\right) - S_L\left(v_{\perp R} v_{\perp R} d_R + \frac{1}{2} g d_R^2\right) + S_L S_R (v_{\perp R} d_R - v_{\perp L} d_L)}{S_R - S_L} \\ n_y \frac{S_R\left(v_{\perp L} v_{\perp L} d_L + \frac{1}{2} g d_L^2\right) - S_L\left(v_{\perp R} v_{\perp R} d_R + \frac{1}{2} g d_R^2\right) + S_L S_R (v_{\perp R} d_R - v_{\perp L} d_L)}{S_R - S_L} \\ \frac{S_R(v_{\perp L} d_L) - S_L(v_{\perp R} d_R) + S_L S_R (d_R - d_L)}{S_R - S_L} c_L \end{bmatrix}$$

$$\begin{bmatrix} \cdots \\ -n_y \frac{S_R(v_{\perp L} d_L) - S_L(v_{\perp R} d_R) + S_L S_R (d_R - d_L)}{S_R - S_L} v_{\parallel L} \\ +n_x \frac{S_R(v_{\perp L} d_L) - S_L(v_{\perp R} d_R) + S_L S_R (d_R - d_L)}{S_R - S_L} v_{\parallel L} \\ \cdots \end{bmatrix} \tag{3.65}$$

The right region and right star region fluxes $\mathbf{F}_R$ and $\mathbf{F}_{*R}$ are computed accordingly.

3.2.6 Hydrostatic reconstruction

As it can be seen in Section 3.2.3, in the Riemann solution only the water depth and flow velocity are considered. The bottom elevation is not taken into account, which means the solution is only valid if the bottom elevation is the same in the left and

right cell. That means, for a simulation involving varying bottom topography, the left and right cell values for the water depth and bottom elevation have to be modified before the Riemann solution. Different topography discretization techniques exist, to deal with this problem (Kesserwani, 2013). The hydrostatic reconstruction is an efficient and robust approach to preserve non-negative water depths and a well-balanced state in a flow field on varying bottom topography including wet-dry interfaces (Audusse et al., 2004; Audusse and Bristeau, 2005). Assuming hydrostatic conditions, the water depth and bottom elevation are hereby modified prior to the Riemann solution (Hou, Simons, Mahgoub and Hinkelmann, 2013). The bed elevation at the edge k is computed as:

$$z_{B,k} = \max\left(z_{B,L}, z_{B,R}\right) \tag{3.66}$$

Then, the lower value between the bed elevation at k and the water elevation on the left hand side is chosen as the new bed elevation at face k:

$$z_{B,k} = \min\left(z_{B,k}, h_L\right) \tag{3.67}$$

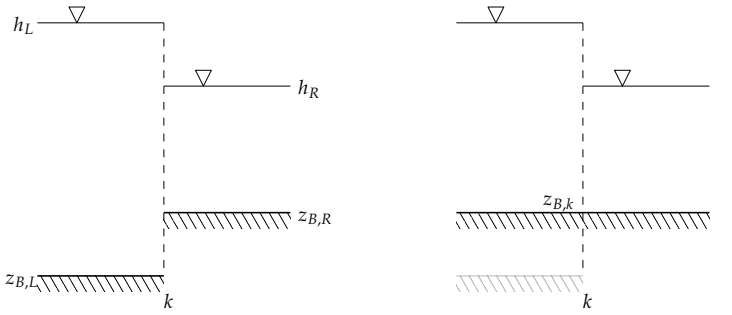
Finally, the water depths are reconstructed by:

$$d_R = \max\left(0,\, h_R - z_{B,k}\right) \tag{3.68}$$

$$d_L = \max\left(0,\, h_L - z_{B,k}\right) \tag{3.69}$$

In Figure 3.11 an example for two cells with water depth higher than the variation in bottom topography is shown. The hydrostatic reconstruction can handle wet and dry fronts, too. This situation is shown in Figure 3.12 where the right cell is dry. After the hydrostatic reconstruction, both cells are handled as dry and therefore zero flux is computed at edge k.

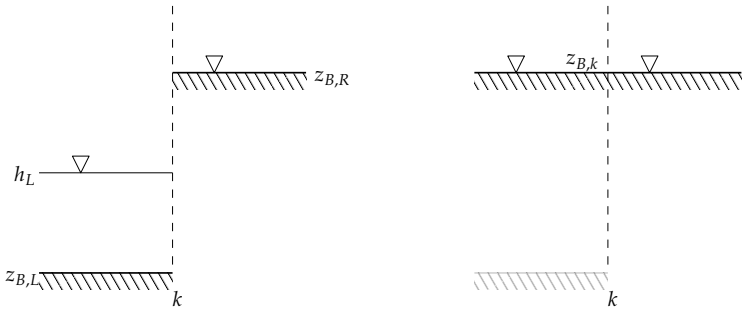
In overland flow applications, the water depth in the sheetlike flow can be smaller than the variation of the bottom topography, e. g. on inclined terrain (Figure 3.13a). The hydrostatic reconstruction will reduce this problem to piecewise flux computations between wet and artificial dry cells (Figure 3.13b) and, as a consequence, a wrong flow field and hydrograph will be computed (Berthon and Foucher, 2012). Simons et al. (2014) show that this problem can be corrected by a high-order scheme. The sketched flow situation will be discretized as a continuous flow problem (Figure 3.13c), and so it is possible to compute flow of thin water on coarse grids with steep bottom gradients.



(a) Piecewise constant discretization

(b) Hydrostatic reconstruction with first-order accurate scheme

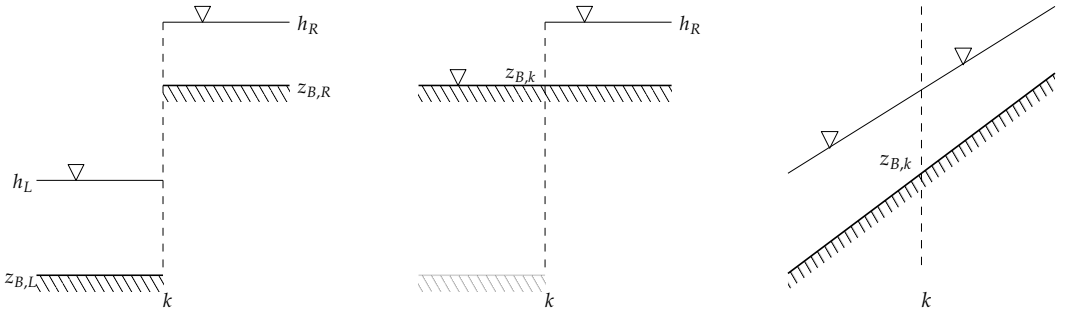
Figure 3.11: Hydrostatic reconstruction for water depth higher than variation in bottom topography



(a) Piecewise constant discretization

(b) Hydrostatic reconstruction with first-order accurate scheme

Figure 3.12: Hydrostatic reconstruction at wet/dry interfaces



(a) Piecewise constant discretization

(b) Hydrostatic reconstruction with first-order accurate scheme

(c) Hydrostatic reconstruction with high-order scheme

Figure 3.13: Hydrostatic reconstruction for a continuous water depth smaller than the variation of the bottom topography

3.3 Monotonic upwind-centered scheme for conservation laws (MUSCL)

In the exact solution of a Riemann problem a shock wave is discontinuous, i. e. there is a perfect jump at the location of the shock front. The ability of a numerical method to resolve such a discontinuity is called the *resolution* of the method. A numerical scheme which smears the discontinuity on less cells than another without introducing numerical oscillations has a higher resolution. In the following, the extension of the Godunov-type methods to *high-resolution methods* is explained.

So far, the face values were always reconstructed first-order accurate, i. e. the face value will be the same as the cell's midpoint value. A high-order scheme can be achieved by piecewise linear approximations of the cell values. However, at discontinuities as jumps or sharp and sudden gradients the linear approximation must be chosen carefully, as over-/undershooting slopes as seen in Figure 3.14 will cause spurious oscillations in the solution. So called total variation diminishing (TVD) limiter allow high-order accuracy and preserve the monotonicity of the solution. This is achieved by using second-order approximations at smooth gradients and reduce the accuracy to first-order at discontinuities. Using high-order reconstruction methods in the framework of the Godunov's method (Section 3.2) results in a monotonic upwind-centered scheme for conservation laws (MUSCL) which was introduced by van Leer in a series of five papers (1973; 1974; 1977*a*; 1977*b*; 1979).

3.3.1 Total variation diminishing methods

Due to overshooting slopes, second-order schemes can introduce new extreme cell values, which have no physical explanation and will produce spurious oscillations in the solution. As a measure for these new extreme values the total variation (TV) is introduced. For a one-dimensional grid the total variation is computed by:

$$\mathrm{TV}\left(\mathbf{q}\right) = \sum_{i=-\infty}^{\infty} \left|\mathbf{q}_{i+1} - \mathbf{q}_i\right| \tag{3.70}$$

A high-order reconstruction method should not introduce new extreme cell values, i. e. the total variation of the new time step should be less or equal than the total variation of the old time step:

$$\mathrm{TV}\left(\mathbf{q}^{n+1}\right) \leq \mathrm{TV}\left(\mathbf{q}^{n}\right) \tag{3.71}$$

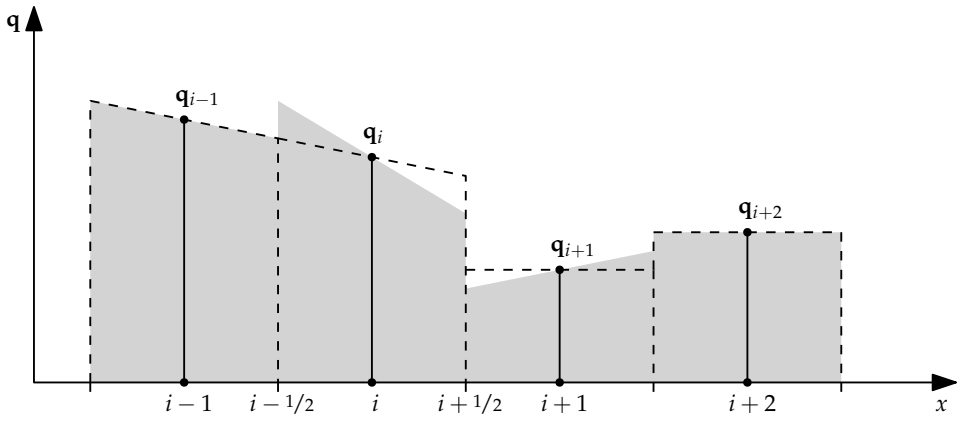


Figure 3.14: Over-/undershooting slopes (grey shaded area) and limited slopes (dashed line)

High-order reconstruction methods fulfilling the TVD condition (Equation 3.71) are called total variation diminishing (TVD) methods.

3.3.2 High-order reconstruction for regular grids

For regular grids the slope limiting approximations of the left and right face value can be written as:

$$\mathbf{q}^{\mathrm{L}}_{i+1/2} = \mathbf{q}_i + \frac{1}{2}\boldsymbol{\Psi}\left(\mathbf{r}^{\mathrm{L}}_{i+1/2}\right)\left(\mathbf{q}_{i+1} - \mathbf{q}_i\right) \tag{3.72}$$

$$\mathbf{q}^{\mathrm{R}}_{i+1/2} = \mathbf{q}_{i+1} + \frac{1}{2}\boldsymbol{\Psi}\left(\mathbf{r}^{\mathrm{R}}_{i+1/2}\right)\left(\mathbf{q}_i - \mathbf{q}_{i+1}\right) \tag{3.73}$$

where $\boldsymbol{\Psi}\left(\mathbf{r}\right)$ is a limiter function depending on the smoothness monitor $\mathbf{r}$:

$$\mathbf{r}^{\mathrm{L}}_{i+1/2} = \frac{\mathbf{q}_i - \mathbf{q}_{i-1}}{\mathbf{q}_{i+1} - \mathbf{q}_i} \tag{3.74}$$

$$\mathbf{r}^{\mathrm{R}}_{i+1/2} = \frac{\mathbf{q}_{i+1} - \mathbf{q}_{i+2}}{\mathbf{q}_i - \mathbf{q}_{i+1}} \tag{3.75}$$

To be TVD the limiter function $\boldsymbol{\Psi}\left(\mathbf{r}\right)$ must fulfill the following condition (Leveque,

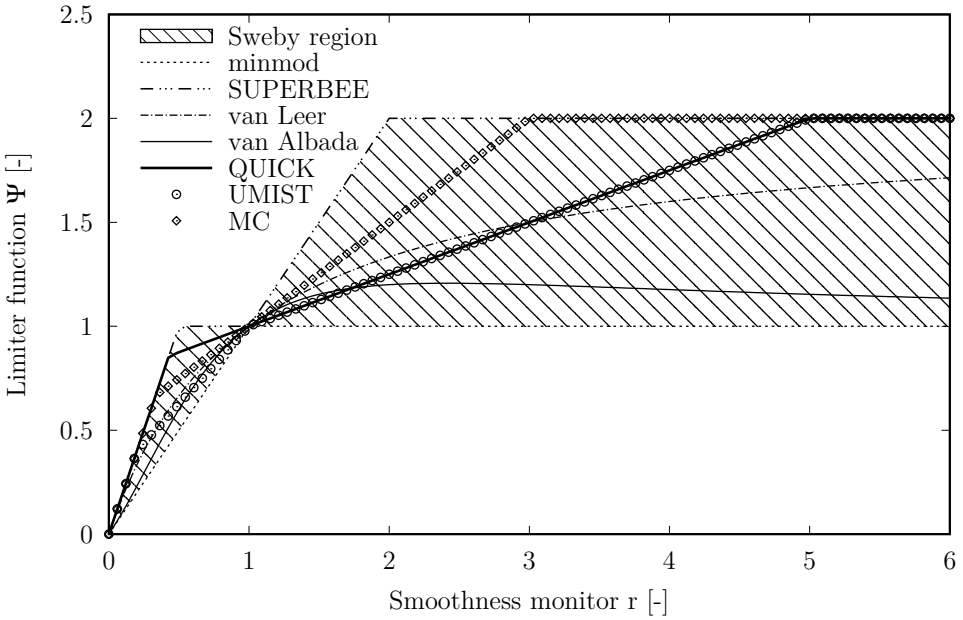


Figure 3.15: Sweby region of second-order total variation diminishing methods and limiter functions

2002):

$$0 \leq \mathbf{\Psi}(\mathbf{r}) \leq \max[0, \min(2, 2\mathbf{r})] \tag{3.76}$$

which states that $\mathbf{\Psi}(\mathbf{r}) \geq 0$, so the limitation is always anti-diffusive. In addition $\mathbf{\Psi}(\mathbf{r})$ should be zero if $\mathbf{r} < 0$ so the limiter function does full limiting at local extrema. Sweby (1984) graphically analyzed the constrains for the limiter function. To guarantee second-order accuracy and avoid strong compression he suggested the shaded region in Figure 3.15. Here, $\mathbf{\Psi}(1) = 1$, so there is no limitation in smooth regions and the method is second-order accurate.

From Equation 3.72 and 3.73 it can be seen, that if the limiter function equals zero or two, the face value equals the left or right cell value respectively and the reconstruction corresponds to the first-order accurate upwind method. This will be the case at sharp, zero or opposing gradients. For a limiter function equal one, a second-order accurate approximation of the face value is achieved.

The transition between these thresholds can be defined differently and many limiter functions exist:

van Leer (van Leer, 1974):

$$\Psi(\mathbf{r}) = \frac{\mathbf{r} + |\mathbf{r}|}{1 + \mathbf{r}} \tag{3.77}$$

van Albada:

$$\Psi(\mathbf{r}) = \max\left(0, \frac{\mathbf{r} + \mathbf{r}^2}{1 + \mathbf{r}^2}\right) \tag{3.78}$$

superbee (Roe, 1985):

$$\Psi(\mathbf{r}) = \max\left[0, \min(2\mathbf{r}, 1), \min(\mathbf{r}, 2)\right] \tag{3.79}$$

UMIST:

$$\Psi(\mathbf{r}) = \max\left[0, \min\left(2, 2\mathbf{r}, \frac{1}{4}(3 + \mathbf{r}), \frac{1}{4}(1 + 3\mathbf{r})\right)\right] \tag{3.80}$$

QUICK:

$$\Psi(\mathbf{r}) = \max\left[0, \min\left(2, 2\mathbf{r}, \frac{1}{4}(3 + \mathbf{r})\right)\right] \tag{3.81}$$

minmod (Roe, 1985):

$$\Psi(\mathbf{r}) = \max\left[0, \min(\mathbf{r}, 1)\right] \tag{3.82}$$

Monotonized central-difference (MC) (van Leer, 1977*a*):

$$\Psi(\mathbf{r}) = \max\left[0, \min\left(2, 2\mathbf{r}, \frac{1}{2}(1 + \mathbf{r})\right)\right] \tag{3.83}$$

All limiter functions listed above lay within the Sweby region as it can be seen in Figure 3.15. The superbee limiter function represents the upper boundary of the Sweby region and has the lowest limitation. The lower boundary is represented by the minmod limiter function, which has the highest limitation and lowest accuracy. Both function are not smooth at $\mathbf{r} = 1$. Full second-order accuracy is achieved by limiter functions which are smooth near $\mathbf{r} = 1$ (Leveque, 2002). This is the case for all other presented limiter functions. The higher the limiter function is located in the

Sweby region, the lower is the limitation and the higher is the accuracy. However, there is also a tendency to spurious oscillations for these methods. Due to its strong limitation, the minmod limiter function has a high numerical stability and is often used.

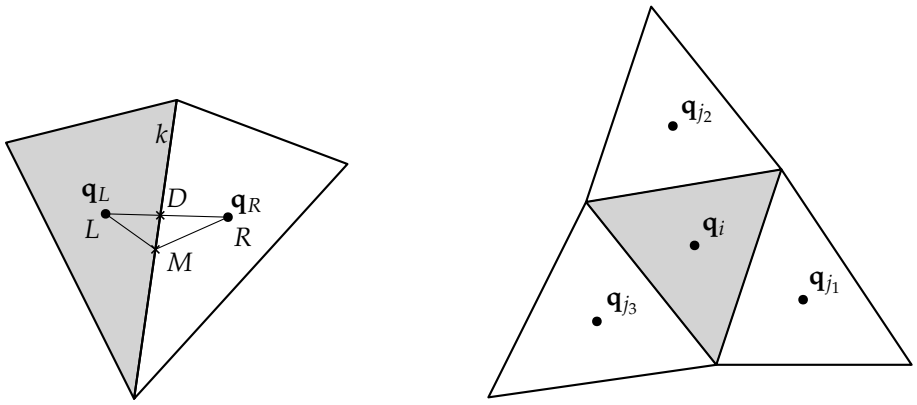
Near the boundary of a computational mesh, the second or the second and direct neighbors of the considered face do not exist. A possibility to deal with such a situation is to switch to first-order reconstruction of the face values. However, in cases, where the bottom topography is sloping towards the boundary, this will result in an artificial impounding of the water at the boundary. Especially for analytical test cases, this an unwanted behavior. Therefore a different approach is followed in this thesis. It is assumed, that the flow field and topography will be smooth and no limitation of the face values is necessary. If either the second or the second and direct neighbors of the considered face is non-existent, the limiter function is always set to $\Psi = 1$.

3.3.3 High-order reconstruction for unstructured meshes

For high-order reconstruction on unstructured meshes two different approaches exist. In the cell-based limiting method a piecewise linear gradient for each cell is computed, which is then limited to fulfill the monotonicity criteria at each cell face. The result is a limited cell gradient which preserves the mean cell values (Buffard and Clain, 2010; Delis and Nikolos, 2013). The disadvantage is, a separate loop to compute the limiter function for each face and for determining the best limiter among those is necessary (Hou, Liang, Simons and Hinkelmann, 2013).

A second approach is the face-based limiting method, where a piecewise linear gradient for each face is computed. This method will not reconstruct one limited cell gradient, but a limited gradient for each face. Several authors showed, that this method is suitable for solving the shallow water equations (Audusse and Bristeau, 2005; Benkhaldoun et al., 2007; Nikolos and Delis, 2009; Liang and Borthwick, 2009; Liang, 2010; Wang et al., 2011; Delis and Nikolos, 2013).

Using the face-based limiting method, the high-order reconstruction is carried out as shown by Hou, Liang, Simons and Hinkelmann (2013). In a first step, the left and right values at the intersection point D of the considered face and the line segment



(a) Considered face k (modified after Hou, Liang, Simons and Hinkelmann, 2013)

(b) Gradient calculation for the ith cell (modified after Hou, Simons, Mahgoub and Hinkelmann, 2013)

Figure 3.16: Variable notation on a triangular grid

LR connecting the left and right cell centroids are reconstructed (cf. Figure 3.16a):

$$\mathbf{q}_D^L = \mathbf{q}_L + \frac{|\mathbf{s}_{LD}|}{|\mathbf{s}_{LR}|}\mathbf{\Psi}\left((\nabla\mathbf{q})_L^{upw}\cdot\mathbf{s}_{LR}, (\nabla\mathbf{q})^{cent}\cdot\mathbf{s}_{LR}\right) \tag{3.84}$$

$$\mathbf{q}_D^R = \mathbf{q}_R + \frac{|\mathbf{s}_{RD}|}{|\mathbf{s}_{RL}|}\mathbf{\Psi}\left((\nabla\mathbf{q})_R^{upw}\cdot\mathbf{s}_{RL}, (\nabla\mathbf{q})^{cent}\cdot\mathbf{s}_{RL}\right) \tag{3.85}$$

where $\mathbf{s}_{LD}$ and $\mathbf{s}_{RD}$ are vectors pointing from the left and right cell centroid to the point D. The subscripts upw and $cent$ denote the upwind and center gradient.

The gradients in Equation 3.84 and 3.85 are computed by:

$$(\nabla\mathbf{q})^{cent}\cdot\mathbf{s}_{RL} = \mathbf{q}_R - \mathbf{q}_L \tag{3.86}$$

$$(\nabla\mathbf{q})_L^{upw} = 2\,(\nabla\mathbf{q})_L - (\nabla\mathbf{q})^{cent} \tag{3.87}$$

$$(\nabla\mathbf{q})_R^{upw} = 2\,(\nabla\mathbf{q})_R - (\nabla\mathbf{q})^{cent} \tag{3.88}$$

In here, $(\nabla\mathbf{q})_L$ and $(\nabla\mathbf{q})_R$ are the unlimited gradients of the flow variables in cell L and R, respectively. These are computed following the approach described by Hou, Simons, Mahgoub and Hinkelmann (2013). The gradient for each component q of the state variable vector $\mathbf{q}$ in cell i which is surrounded by the cells j_1, j_2 and j_3 is

expressed as (cf. Figure 3.16b):

$$(\nabla q)_L = \mathbf{J} \begin{pmatrix} q_{j_2} - q_{j_1} \\ q_{j_3} - q_{j_1} \end{pmatrix} \tag{3.89}$$

with:

$$\mathbf{J} = \frac{1}{(x_{j_2} - x_{j_1})(y_{j_3} - y_{j_1}) - (x_{j_3} - x_{j_1})(y_{j_2} - y_{j_1})} \begin{pmatrix} y_{j_3} - y_{j_1} & y_{j_1} - y_{j_2} \\ x_{j_1} - x_{j_3} & x_{j_2} - x_{j_1} \end{pmatrix} \tag{3.90}$$

For the two-argument limiter function $\mathbf{\Psi}(a, b)$ in Equation 3.84 and 3.85 the following formulas are used:

van Albada:

$$\mathbf{\Psi}(a, b) = \begin{cases} \frac{(a^2+e)b+(b^2+e)a}{a^2+b^2+2e} & \text{if } ab > 0 \\ 0 & \text{if } ab \leq 0 \end{cases} \tag{3.91}$$

where $e = 10^{-16}$ to prevent division by zero (Nikolos and Delis, 2009; Delis et al., 2011; Delis and Nikolos, 2013).

minmod:

$$\mathbf{\Psi}(a, b) = \max[0, \min(a, b)] \tag{3.92}$$

Since the intersection point D does not in general coincide with the midpoint of the considered face M, directly using the reconstructed values at point D for flux computation can cause inaccuracies when the deviation between M and D is high. Therefore a directional correction is proposed by Delis et al. (2011):

$$\mathbf{q}_M^L = \mathbf{q}_D^L + \mathbf{s}_{DM} \cdot (\nabla \mathbf{q})_L \tag{3.93}$$

$$\mathbf{q}_M^R = \mathbf{q}_D^R + \mathbf{s}_{DM} \cdot (\nabla \mathbf{q})_R \tag{3.94}$$

As the unlimited cell gradient is used in Equations 3.93 and 3.94 new local extrema can arise from the correction. Hou, Liang, Simons and Hinkelmann (2013) propose

an additional limitation based on the maximum principle constrain:

$$\min\left(\mathbf{q}_R, \mathbf{q}_L\right) \leq \mathbf{q}_M^L \leq \max\left(\mathbf{q}_R, \mathbf{q}_L\right) \tag{3.95}$$
$$\min\left(\mathbf{q}_R, \mathbf{q}_L\right) \leq \mathbf{q}_M^R \leq \max\left(\mathbf{q}_R, \mathbf{q}_L\right) \tag{3.96}$$

3.3.4 Reconstruction of state variables

To preserve conservation of mass and momentum, not all flow variables in the SWE are directly reconstructed, but only the water depth, water elevation and specific discharge (Audusse and Bristeau, 2005). The bottom elevation will then be computed as (Hou, Simons, Mahgoub and Hinkelmann, 2013):

$$z_{\mathrm{B}M}^L = h_M^L - d_M^L \tag{3.97}$$
$$z_{\mathrm{B}M}^R = h_M^R - d_M^R \tag{3.98}$$

and the flow velocity as:

$$\mathbf{v}_M^L = \frac{\mathbf{q}_M^L}{d_M^L} \tag{3.99}$$
$$\mathbf{v}_M^R = \frac{\mathbf{q}_M^R}{d_M^R} \tag{3.100}$$

where the velocities are set to zero if a threshold water depth value is reached.

Wetting and drying

In case of the water depth in a cell is smaller than a threshold value (default value used in this thesis: $d \leq 1.0 \cdot 10^{-8}$ m), it is considered dry. For flux calculation a zero water depth is assumed for those cells. However, the state variable itself is not modified to guarantee mass conservation. If either the left or right cell is dry, first-order reconstruction is used at the considered cell face. In addition the stability criterion presented by Murillo et al. (2006) is adapted. First-order reconstruction is used in case of:

$$\min(d_R, d_L) < \max(d_R, d_L) - \min(d_R, d_L) \tag{3.101}$$

3.4 Slope source term treatment

To obtain a well-balanced solution which fulfills the C-property, different methods for the slope source treatment exist (Kesserwani, 2013). To simplify the solution on arbitrary meshes, the bottom slope source term of a cell (Equation 2.2) can be transformed into fluxes through its faces (Audusse et al., 2004; Hou, Liang, Simons and Hinkelmann, 2013). The volume integral of the bottom slope source term $\mathbf{s}_b$ can be expressed as the sum of the bottom slope fluxes:

$$\int_{\Omega} \mathbf{s}_b \, \mathrm{d}\Omega = \oint_{\Gamma} \mathbf{F}_{bk}\mathbf{n} \, \mathrm{d}\Gamma = \sum_k \mathbf{F}_{bk}\mathbf{n}_k l_k \tag{3.102}$$

where the bottom slope flux $\mathbf{F}_{bk}$ is equal to:

$$\mathbf{F}_{bk}\mathbf{n}_k = \begin{bmatrix} 0 \\ -n_{xk}\frac{1}{2}g\left(d_k^L + d_L\right)\left(z_{Bk} - z_B^L\right) \\ -n_{yk}\frac{1}{2}g\left(d_k^L + d_L\right)\left(z_{Bk} - z_B^L\right) \end{bmatrix} \tag{3.103}$$

3.5 Friction source term treatment

As a consequence of the very small water depths, the bottom friction has a notable influence in surface runoff applications. To avoid numerical instabilities, a fully-implicit discretization is used for the friction source term. This is achieved by using a splitting point-implicit method described in Bussing and Murman (1988), Yoon and Kang (2004) and Liang and Marche (2009). Writing the equations of momentum (Equations (2.1) and (2.2)) in the point-implicit method gives:

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t} = -\frac{1}{A}\sum_k \mathbf{F}_k^n \mathbf{n}_k l_k + \mathbf{s}_f^{n+1} \tag{3.104}$$

where $\mathbf{q} = (ud, vd)^T$ in this description. $\mathbf{s}_f^{n+1}$ can be expressed using a Taylor series expansion around the n^{th} time level:

$$\mathbf{s}_f^{n+1} = \mathbf{s}_f^n + \left(\frac{\partial \mathbf{s}_f}{\partial \mathbf{q}}\right)^n \Delta\mathbf{q} + O\left(\Delta\mathbf{q}^2\right) \tag{3.105}$$

where $\Delta\mathbf{q} = \mathbf{q}^{n+1} - \mathbf{q}^n$. Following the procedure given by Bussing and Murman (1988), substituting Equation (3.105) into Equation (3.104) yields:

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t} = -\frac{1}{A}\sum_k \mathbf{F}_k^n \mathbf{n}_k l_k + \mathbf{s}_f^n + \left(\frac{\partial \mathbf{s}_f}{\partial \mathbf{q}}\right)^n \left(\mathbf{q}^{n+1} - \mathbf{q}^n\right) \tag{3.106}$$

which can be rewritten as:

$$\mathbf{PI}\left(\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t}\right) = -\frac{1}{A}\sum_k \mathbf{F}_k^n \mathbf{n}_k l_k + \mathbf{s}_f^n \tag{3.107}$$

where the matrix **PI** is equal to:

$$\mathbf{PI} = \mathbf{I} - \Delta t \left(\frac{\partial \mathbf{s}_f}{\partial \mathbf{q}}\right)^n \tag{3.108}$$

Herein, **I** is the identity matrix and $\left(\partial \mathbf{s}_f / \partial \mathbf{q}\right)^n = \mathbf{J}_f$ is the Jacobian matrix of the friction source term. During the simulation, the friction and source terms are computed on the old time level, and are then divided by **PI** to get the final solution:

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \frac{1}{\mathbf{PI}}\left(-\frac{\Delta t}{A}\sum_k \mathbf{F}_k^n \mathbf{n}_k l_k + \Delta t\, \mathbf{s}_f^n\right) \tag{3.109}$$

Bussing and Murman (1988) mention that **PI** acts as a preconditioner for the systems of equations. It should be pointed out, that preconditioners are normally used in the context of implicit solution of system of equations, to facilitate the numerical solution of a given problem. The term preconditioning is adapted in this thesis to describe the point-implicit treatment of the friction term, even it is not completely valid in an explicit context.

3.6 Diffusive flux treatment

So far, all methods dealt with the solution of the advective flux term. In the momentum equations of the shallow water equations and in the transport equation for a dissolved component, diffusive fluxes appear. In general, the diffusive flux normal to the interface is:

$$\mathbf{F}_k \mathbf{n}_k l_k = -D\nabla \mathbf{q} \tag{3.110}$$

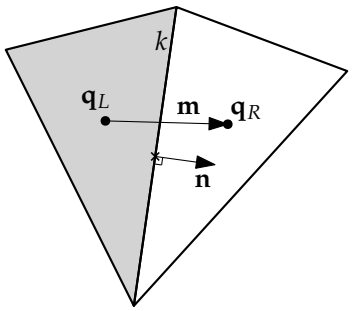


Figure 3.17: Computation of gradient $\nabla \mathbf{q}$ at face k (modified after Luke, 2007)

or, written as fluxes in x and y direction:

$$\mathbf{f} = -D\frac{\partial \mathbf{q}}{\partial x}, \quad \mathbf{g} = -D\frac{\partial \mathbf{q}}{\partial y} \tag{3.111}$$

Solving the diffusive flux term requires the gradient of the conservative variable normal to the considered edge. For structured rectangular meshes, this can be easily determined by the gradient of the two neighboring cells. For example in x direction, the gradient can be written as:

$$\nabla \mathbf{q} = \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{\Delta x} \tag{3.112}$$

where Δx is the cell size in x direction.

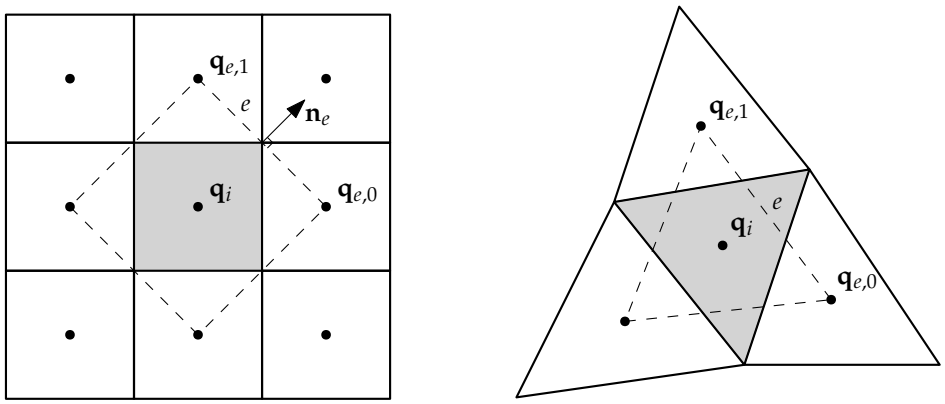
However, on arbitrary unstructured meshes, the connecting line between two cell centroids will not always coincide with the midpoint of the considered edge. Here, a method for computing the gradient normal to a cell edge for arbitrary cells presented by Luke (2007) is adopted. The gradient of $\mathbf{q}$ at the considered edge can be split in a normal and a tangential component:

$$\nabla \mathbf{q} = (\nabla \mathbf{q})_\perp + (\nabla \mathbf{q})_\parallel \tag{3.113}$$

The tangential component is determined by:

$$(\nabla \mathbf{q})_\parallel = \overline{\nabla \mathbf{q}} - \left(\overline{\nabla \mathbf{q}} \cdot \mathbf{n}\right) \mathbf{n} \tag{3.114}$$

In here, $\overline{\nabla \mathbf{q}}$ is the mean cell gradient found by cell area weighted averaging the left



(a) Structured rectangular mesh (b) Arbitrary triangular mesh

Figure 3.18: Computation of gradient $\nabla\mathbf{q}$ at cell i

and right cell gradients:

$$\overline{\nabla\mathbf{q}} = \frac{A_L\left(\nabla\mathbf{q}\right)_R + A_R\left(\nabla\mathbf{q}\right)_L}{A_L + A_R} \tag{3.115}$$

Finally, the normal component of the edge gradient is computed by:

$$\left(\nabla\mathbf{q}\right)_\perp = \frac{\mathbf{q}_R - \left[\mathbf{q}_L + \left(\nabla\mathbf{q}\right)_\parallel \cdot \mathbf{m}\right]}{\mathbf{n}\cdot\mathbf{m}} \tag{3.116}$$

where $\mathbf{m}$ is a vector pointing from centroid of the left cell to the centroid of the right cell (cf. Figure 3.17).

Computing the mean gradient requires the left and right cell gradient. Therefore a method shown by Tamás (2006) is adapted. The method is suitable for arbitrary grids, too. A polygon is constructed by connecting the centroids of all direct neighbor cells of the considered cell (cf. Figure 3.18). The cell gradient is approximated by computing the gradient of this polygon:

$$\left(\nabla\mathbf{q}\right)_i = \frac{1}{\Omega}\oint_\Gamma \mathbf{q}\,\mathbf{n}\,\mathrm{d}\Gamma \approx \frac{1}{A_{\text{path}}}\sum_{e\in\text{edges}}\frac{\mathbf{q}_{e,0}+\mathbf{q}_{e,1}}{2}\mathbf{n}_e l_e \tag{3.117}$$

where e denotes the edges of the polygon and $e,0$ and $e,1$ refer to the starting and ending cell of the edge. $\mathbf{n}_e$ is an outward pointing unit normal vector and l_e is the

length of the edge e. A_{path} is the area of the polygon:

$$A_{\text{path}} = \sum_{e \in \text{edges}} \frac{1}{2} \left(x_{e,0} - x_{e,1}\right) \left(y_{e,0} - y_{e,1}\right) \tag{3.118}$$

If the considered cell is a boundary cell and a neighboring cell does not exist, the centroid of the non-existent cell is found by extrapolating the distance between cell centroid and midpoint of the considered edge and the values from the boundary condition are used.

3.7 Boundary conditions

To determine variables at the boundary edges or gradients and source terms in cells near the boundary, so-called boundary conditions are necessary. They define the state variables of a non-existent neighboring (ghost) cell next to the cell under consideration. Three types of boundary conditions are considered: A *Dirichlet*[3] boundary condition defines a fixed value for a state variable. In a *Neumann*[4] boundary condition, the gradient of state variables is defined which determines the value at the ghost cell. The most common is a zero gradient condition where the boundary value will equal the inner cell value. Finally, combinations of these two types of boundary conditions are possible. For example a fixed value for a state variable is defined if water flows in the domain and a zero gradient condition is used if the water flows out of the domain. In the following the most common boundary conditions for the shallow water equations and for the transport equation for tracer or sediment transport are shown.

3.7.1 Solid (slip) boundary

At a solid wall the flow normal to the boundary edge will be zero. For a frictionless wall the flow tangential to the boundary edge will be not influenced by the wall at

[3] named after P. G. L. Dirichlet (1805 – 1859)
[4] named after C. G. Neumann (1832 – 1925)

all (slip condition). The values of the ghost cell are set to:

$$v_{\perp ghost} = -v_{\perp inner} \tag{3.119}$$
$$v_{\parallel ghost} = v_{\parallel inner} \tag{3.120}$$
$$d_{ghost} = d_{inner} \tag{3.121}$$
$$c_{ghost} = c_{inner} \tag{3.122}$$

The opposing normal flow velocities at the ghost and inner cell cancel each other and result in a zero flux over the boundary edge. The inner normal and tangential flow velocities are determined by Equation 3.59. Finally, the flow velocities in x and y direction in the ghost cell are determined by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} n_x & -n_y \\ n_y & n_x \end{bmatrix} \begin{bmatrix} v_\perp \\ v_\parallel \end{bmatrix} = \begin{bmatrix} n_x v_\perp - n_y v_\parallel \\ n_y v_\perp + n_x v_\parallel \end{bmatrix} \tag{3.123}$$

3.7.2 Free outflow

An ideal free outflow boundary condition would pass all outgoing fluxes without introducing reflections at the boundary and there will be zero ingoing flux.

Setting the normal flow velocity in the ghost cell in Equation 3.119 equal to the inner flow velocity $v_{\perp ghost} = v_{\perp inner}$, the boundary values are determined completely by the inner cell values. In addition, if there is ingoing flux at an edge ($\mathbf{v}\,\mathbf{n} < 0$), the boundary condition is switched to the solid boundary, i. e. $v_{\perp ghost} = -v_{\perp inner}$.

This will give good results for supercritical flow (Fr ≥ 1). But, recalling the theory of the Riemann problem, there is always a wave traveling upstream in subcritical conditions (Fr < 1). Thus, there will be always an influence of the outflow boundary on the solution inside the domain and as a consequence, the flow will tend to supercritical conditions. To improve this situation, the water depth in the ghost cell can be limited to a critical value:

$$d_{ghost} = \max\left(d_{inner}, d_{critical}\right) \tag{3.124}$$

If the inner water depth drops below the critical value, the water depth in the ghost cell is limited.

In case of a transport equation, at a free outflow boundary the concentration in the

ghost cell is set equal to the concentration in the inner cell:

$$c_{ghost} = c_{inner} \tag{3.125}$$

3.7.3 Constant water depth

At a constant water depth boundary, the water depth is set to a fixed value:

$$d_{ghost} = d_{boundary} \tag{3.126}$$

The determination of the flow velocity in the ghost cell depends on the local Froude number. For subcritical flow (Fr $<$ 1), the state of the ghost and inner cell are connected via a rarefaction wave. As the normal vector is by definition always pointing outwards at the boundary, the left cell is the inner cell and the right cell is the ghost cell. Using the Riemann invariant of the 3-wave (Equation 3.23) gives:

$$v_{\perp ghost} + 2\sqrt{gd_{ghost}} = v_{\perp inner} + 2\sqrt{gd_{inner}} \tag{3.127}$$

which can be written as:

$$v_{\perp ghost} = v_{\perp inner} + 2\sqrt{gd_{inner}} - 2\sqrt{gd_{ghost}} \tag{3.128}$$

The tangential flow velocity in the ghost cell equals the inner cell value:

$$v_{\parallel ghost} = v_{\parallel inner} \tag{3.129}$$

For supercritical flow conditions (Fr $\geq$ 1) the flow velocity at the boundary has to be given as fixed values:

$$v_{\perp ghost} = v_{\perp boundary} \tag{3.130}$$

$$v_{\parallel ghost} = v_{\parallel boundary} \tag{3.131}$$

Again, the flow velocities in x and y direction are computed using Equation 3.123.

3.7.4 Constant orthogonal flow velocity

At a boundary with constant orthogonal flow velocity, the flow velocity normal to the boundary edge is set to a fixed value in the ghost cell:

$$v_{\perp ghost} = v_{\perp boundary} \tag{3.132}$$

By using the Riemann invariant (Equation 3.127), the water depth in the ghost cell is computed as the following for Fr < 1 (subcritical flow):

$$d_{ghost} = \frac{\left(v_{\perp inner} - v_{\perp ghost} + 2\sqrt{g d_{inner}}\right)^2}{4g} \tag{3.133}$$

and:

$$v_{\parallel ghost} = v_{\parallel inner} \tag{3.134}$$

For supercritical flow (Fr ≥ 1), the water depth and the tangential velocity have to be given as fixed values:

$$d_{ghost} = d_{boundary} \tag{3.135}$$

$$v_{\parallel ghost} = v_{\parallel boundary} \tag{3.136}$$

3.7.5 Constant orthogonal specific discharge

Instead of directly giving the water depth or flow velocity, the specific discharge normal to the boundary edge $q_{\perp boundary}$ can be given:

$$d_{ghost} \cdot v_{\perp ghost} = q_{\perp boundary} \tag{3.137}$$

Substituting $v_{\perp ghost} = q_{\perp boundary}/d_{ghost}$ in Equation 3.127 and rewriting the equation gives (Song et al., 2011):

$$c^3_{\mathrm{w}ghost} - \frac{1}{2}\left(v_{\perp inner} + 2\sqrt{g d_{inner}}\right) c^2_{\mathrm{w}ghost} + \frac{1}{2} g q_{\perp boundary} = 0 \tag{3.138}$$

with $c_{\mathrm{w}ghost} = \sqrt{g d_{ghost}}$.

Equation 3.138 is solved iteratively for the unknown $c_{\mathrm{w}ghost}$ using the Newton-Raphson method and the water depth and flow velocity in the ghost cell are determined by:

$$d_{ghost} = \frac{c^2_{\mathrm{w}ghost}}{g} \tag{3.139}$$

$$v_{\perp ghost} = \frac{q_{\perp boundary}}{d_{ghost}} \tag{3.140}$$

$$v_{\parallel ghost} = v_{\parallel inner} \tag{3.141}$$

3.7.6 Constant concentration

For a constant concentration of a dissolved component or sediment concentration, the concentration value in the ghost cell will be set to a fixed value:

$$c_{ghost} = c_{boundary} \tag{3.142}$$

3.7.7 Bottom elevation at boundary

For the computation of the bottom slope source term (Section 3.4), the bottom elevation of the neighboring cells of an edge is necessary. At the boundary, the easiest way to determine the bottom elevation in the ghost cell is to mirror the value of the inner cell:

$$z_{\mathrm{B}ghost} = z_{\mathrm{B}inner} \tag{3.143}$$

However, in cases, where the bottom topography is sloping towards the boundary, this will result in an artificial impounding of the water at the boundary. To avoid this, the bottom elevation in the ghost cell can e. g. determined by linear extrapolation from the inner cell values or using data of the underlying digital elevation model or using a function describing the bottom topography. Together with the handling of second-order accuracy at the boundary presented in Section 3.3.2, this will result in a smooth solution.

3.8 Conclusions

In this chapter robust numerical methods for the solution of the governing equations were presented. First, the spatial and temporal discretization of the partial differential equations were shown. Applying Godunov's method the computation of the numerical fluxes was regarded as solution of a Riemann problem. The exact and approximated solutions of a Riemann problem were given. The hydrostatic reconstruction was presented as an efficient approach to deal with wet/dry interfaces. The spatial accuracy of the scheme was increased to second-order using total variation diminishing methods which guarantee the monotonicity of the solution. Different limiter function were explained. Then, the discretization of the slope source, friction source and diffusive flux terms was shown and the chapter was closed with a description of the most common boundary conditions.

4 Development of a flexible software concept

As shown in Chapter 3, a robust solution to surface water flow and transport problems requires complex numerical algorithms for the solution of the governing partial differential equations. Also, there is no single "perfect" approach, but different numerical approaches, with each having its own advantages and disadvantages regarding accuracy and computational efficiency. If further processes as e. g. transport or runoff generation are regarded, their solution has to be integrated in the numerical solution process, too, and the interactions between these processes have to be considered.

The goal of `hms` is to develop a software framework which allows the implementation of numerical algorithms and to apply these implementations for simulating single and multiple physical processes in the context of water and environment related problems. In this thesis the initial software architecture presented by Busse (2017) will be enhanced by a flexible numerical framework.

In this chapter an introduction to `hms` will be given, the new components in the software architecture will be described and as a proof of concept a prototype implementation will be described.

4.1 Introduction to `hms`

4.1.1 Software infrastructure

Packages allow grouping parts of a large software project depending on their purpose. In Figure 4.1, the top-level package structure of `hms` is given. The basic infrastructure, which can not only be used in the context of `hms`, is located in the `terma`[1] package. It includes the following subpackages and libraries:

[1] The term "terma" is used in Tibetan Buddhism and means "hidden treasure".

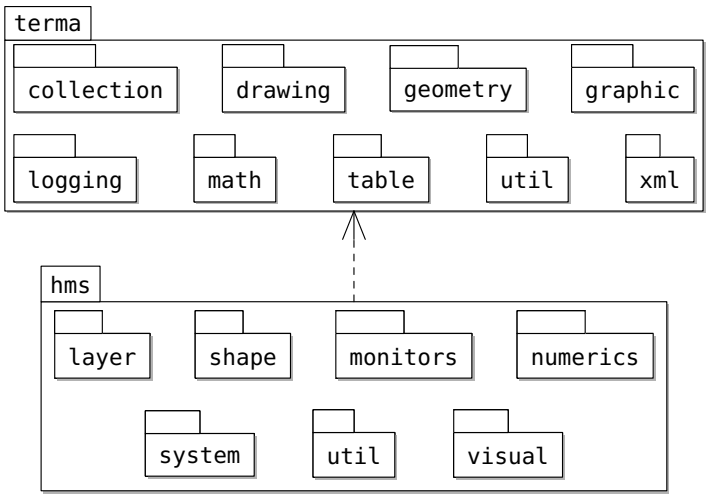


Figure 4.1: Top-level package structure of `hms`

`collection` Collection and iterator classes which allow parallel iteration over elements.

`drawing` Basic infrastructure for creating viewers.

`geometry` Two-dimensional geometry types: point, line, polyline and polygon etc. and geometrical operations. Details are given by Busse (2017). A new and well-tested implementation of the geometrical operations has been contributed by the author of this thesis.

`graphic` Colors and fonts libraries.

`logging` Logging information to standard output.

`math` Mathematical operations as Newton's iteration, Gauss-Seidel elimination etc.

`table` Data model to represent tabular data.

`util` Utilities to work with files, to represent time, and to run multithreaded applications.

`xml` Basic XML functionality.

Dependent on these packages the core of `hms` is developed. It is subdivided into several subpackages:

`layer` In `hms`, layers are used for model integration (see Section 4.1.2). The layer package includes the layer definition and all available specific layer imple-

mentations (e. g. layers for surface water flow, sediment transport, infiltration etc.).

shape Based on the geometry library, the shape package includes implementations of different mesh types which can be used for finite volume computations (see Section 4.1.3).

monitors Monitors allow monitoring various parameters at simulation runtime and writing them to files.

numerics All numerical algorithms are located in the numerics package. These are the base components and all specific numerical schemes implemented so far. The package's structure and content have been developed as part of this thesis.

system The system package contains all components which help to build up a running application. These are for example the layer manager and time representation (see Section 4.1.3).

util The utils package contains various readers and writers for mesh and digital elevation data etc.

visual Based on the drawing package, a sophisticated viewer for runtime visualization was developed.

4.1.2 Integrated modeling concept

In an integrated modeling concept, several numerical models describing specific processes are combined to create a new numerical model describing interactions between processes and their overall impact. For example, a numerical model could describe surface runoff, i. e. the shallow water equations are solved to determine the state variables *flow velocities* and *water depth* inside a considered domain. Combining this model with other numerical models describing rainfall, infiltration and perhaps evaporation creates a complex numerical model which can be used to describe a hydrological catchment area.

In hms layers are used for model integration (Figure 4.2). A layer can be understood as a conceptual container that contains all the components needed for a specific numerical model (Busse, 2017). A layer contains geometric information about the spatial discretization and the topology. Attributes such as geospatial information or physical state variables can be attached as attributes to the geometric elements. A complex application is build up of several layers. Interactions and computations

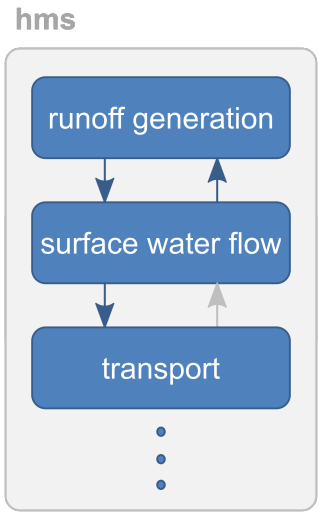


Figure 4.2: Model integration using layers

are coordinated by a layer manager. An in-depth description of the layer concept is given by Busse (2017).

4.1.3 `hms` as a framework for finite volume computations

So far, a layer is a container for any kind of model. For example its components could be used to develop an application based on cellular automata. However, `hms` has been primarily developed as a framework for the solution of two-dimensional partial differential equations using the finite volume method. For doing this, data-structures for representing structured and unstructured mesh types and the simulation domain including boundary conditions have been developed. Based on the `geometry` package, data structures to represent structured and unstructured cell-centered finite volume meshes have been implemented (cf. Section 3.1.1). A `Mesh` is build up from `MeshCells` (Figure 4.3). Each `MeshCell` has three or more `MeshCellEdges`. A `MeshCellEdge` holds reference to both of its direct neighbor cells. For second-order accuracy in space, the references to the second neighbor cells can be stored, too. Both, `MeshCell` and `MeshCellEdge` are derived from classes in the `geometry` package, i. e. a `MeshCell` is a `Polygon` and and `MeshCellEdge` is a `Line`. Using the infrastructure of the `geometry` package, it is simple to determine cell area, cell center, edge length, edge center and normal vectors.

`MeshCells` and `MeshCellEdges` can store `AttributeVariables`, which represent the

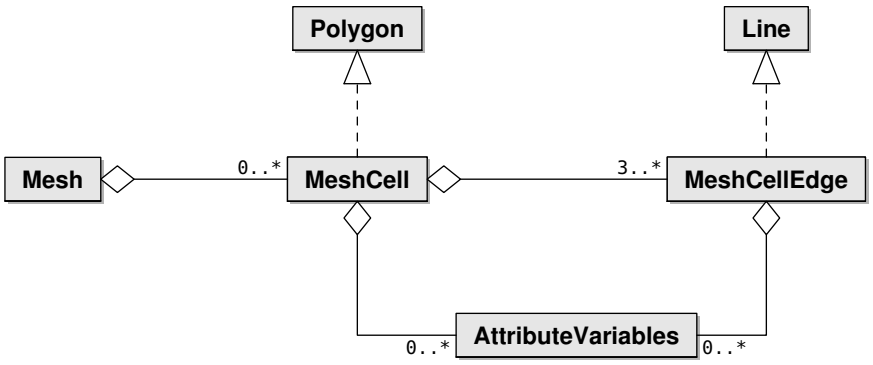


Figure 4.3: Representation of unstructured meshes in hms

conserved state variables in the FVM.

Time is represented in hms as TimeStamps, which represent one specific point in time and TimeIntervals, which represent the time interval between two TimeStamps. That means a TimeInterval has a start and end time and can return the time step size in between. In the numerical algorithms, both can be converted to primitive double values.

The time loop is coordinated by a layer manager. In the end of a computational step, each layer involved in the running application returns its desired time step size for the next step. The smallest time step size of all layers will be used for the next computational step.

The numerical algorithms are not directly implemented into the layer but in separate components, to make them reusable and to allow them to act on multiple layers.

4.1.4 Model coupling

hms supports a generalized way of layer or model coupling by defining interfaces for data exchange. The interaction consist of three phases: establishing a connection, mapping the locations of interest in both layers and transforming the data (Busse, 2017). This generalized coupling approach has several advantages: Each layer and its computational algorithms can be developed and used autonomously. The coupling is possible, even if the models use different spatial scales. And finally, using the Open Modeling Interface (OpenMI) standard (Gregersen et al., 2007; OpenMI Association, 2018), a coupling is even possible with software which is not part of the hms framework (Mieth, 2008; Stadler, 2016; Busse, 2017).

However, the generalized coupling approach puts some drawbacks on the performance of the simulation, or in other words, some optimizations are not possible in such an environment. For example, typically transport processes are based on a flow field, i. e. the tracer transport equation is coupled with the shallow water equations. If, in addition, both model use the same spatial resolution, it would be possible to use the same numerical mesh and, as described in Chapter 3, compute all numerical fluxes at once using a Riemann solver. The generalized coupling approach using data mapping between layers, does not support these optimizations and leads therefore to memory and computational overhead.

As `hms` aims to provide a fully flexible framework for model development, it is preferable to enhance the flexibility for the development of performant numerical algorithms. As a consequence, in this thesis, the existing software architecture was extended by a sophisticated generic finite volume solver which allows direct performant coupling of numerical models within the initial design and structure of `hms`.

4.2 New generic finite volume solver

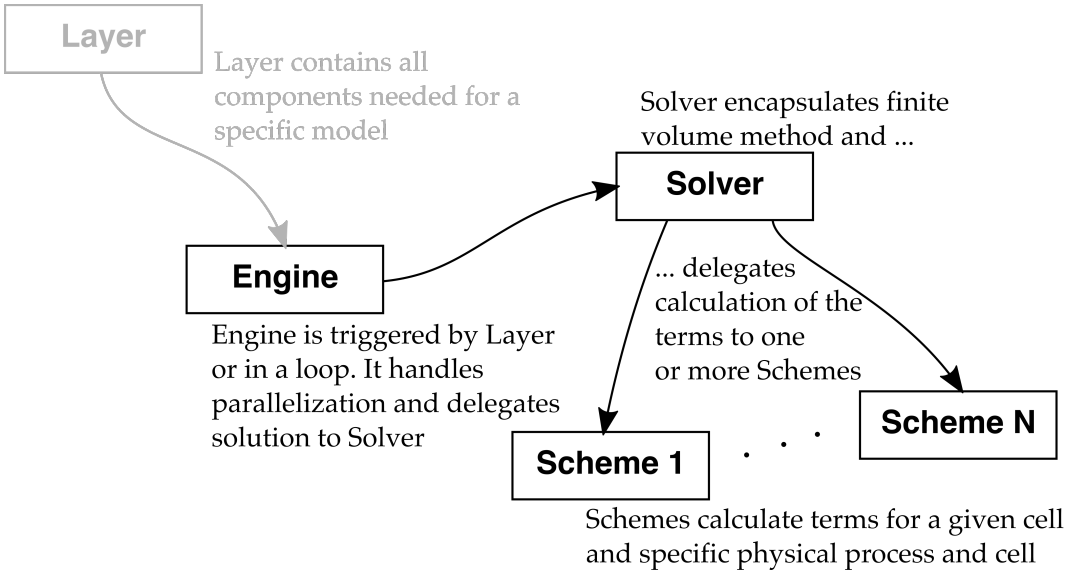


Figure 4.4: Architecture of numerics framework

By the design of `hms`, the computation of a new time step is broken up in three steps

which are `prepare`, `compute` and `update` (Busse, 2017). Each time step, these steps are called for each layer sequentially. That means, first the `prepare` method of all computational layers is called by the layer manager. Than the `compute` method and finally the `update` method is called. A layer which represents a time-dependent numerical model delegates these steps to an `Engine`. Based on these three steps, the explicit cell-centered finite volume method, which was presented in Chapter 3, has to be applied to one or multiple processes. The goal is an abstract design, which reduces the application to new processes to the implementation of the discretized mathematical formulation of the flux and source terms of the considered process.

Based on Chapter 3, using the explicit cell-centered finite volume method, the discretized partial differential equation for the general conservation law (Equation 2.1) can be written as:

$$\underbrace{\mathbf{q}^{n+1}}_{\text{new storage}} = \underbrace{\mathbf{q}^{n}}_{\text{old storage}} - \underbrace{\mathbf{P}}_{\text{preconditioner}} \left(\frac{\Delta t}{A} \underbrace{\sum_{k=1}^{n_b} \mathbf{F}_k^n \mathbf{n}_k l_k}_{\text{flux}} - \Delta t \underbrace{\mathbf{s}^n}_{\text{source}} \right) \tag{4.1}$$

Equation 4.1 can be subdivided into five terms, which are the new and old storage or conserved state variables, flux term, source term and a potential "preconditioning" term acting on the flux and source terms (cf. Section 3.5). For all mathematical models presented in Chapter 2, Equation 4.1 is the same. However, the concrete computation of each term is different for each model. For example, the conserved state variable of the surface water flow model is the water depth d (cf. Equation 2.2) whereas for a model for transport of a single component it is concentration times water depth cd (cf. Equation 2.33). The source term in the momentum equation of the surface water flow model is composed of the bed friction and the bottom slope source term. In the transport equation the source term can be a chemical reaction source/sink term.

To simplify the implementation of different physical processes and the examination of different numerical solution methods, in the software design the generic finite volume solver is separated from the concrete mathematical computation of the terms in Equation 4.1. Therefore, the following components are considered (Figure 4.4):

`Engine` The `Engine` component encapsulates the numerical computations. It is triggered by a layer, but can also live on itself. The `Engine` component handles the parallelization of the numerical solution and delegates the actual solution process to a `Solver`.

`Solver` The `Solver` component gets a range of cells for which it will run the prepare, compute and update step. In the compute step it solves Equation 4.1 for one or more physical processes and delegates the calculation of the terms to one or more `Schemes`.

`Scheme` The `Scheme` component calculates the terms of Equation 4.1 of one physical process for a given cell.

In the following, the design of these three components is regarded in detail.

4.2.1 Engine

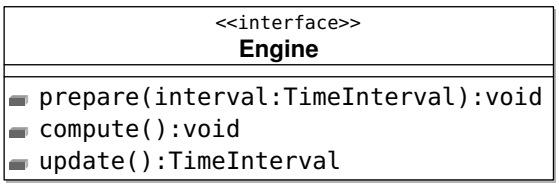


Figure 4.5: The `Engine` interface

The `Engine` component encapsulates all numerical computations. It has the three methods `prepare`, `compute` and `update` which have to be called sequentially in a loop or which are triggered by the layer containing the `Engine`. In Figure 4.5 the UML representation of the interface is given. The `prepare` method has a `TimeInterval` as an argument, i. e. the current time and time step size. The `update` method has to return the preferred time step size for the next computational step as a `TimeInterval`, which can be considered in the global time loop.

The `Engine` component handles the parallelization of all numerical computations to make use of modern multicore computers. The computational mesh is decomposed to different sections with equal size. Each section is assigned to its own computational thread. The engine creates a copy of the `Solver` component for each available thread (`solver1`, `solver2` ... `solverN`) and calls their methods in parallel. To improve the computational performance, creating copies is necessary, to allow each `Solver` instance to cache fields for reuse in numerical algorithm. In the UML sequence diagram in Figure 4.6, the parallel execution is demonstrated. The `prepare`, `compute` and `update` methods of each copy of the `Solver` are called in parallel and the computation itself is done sequentially for the specified mesh section.

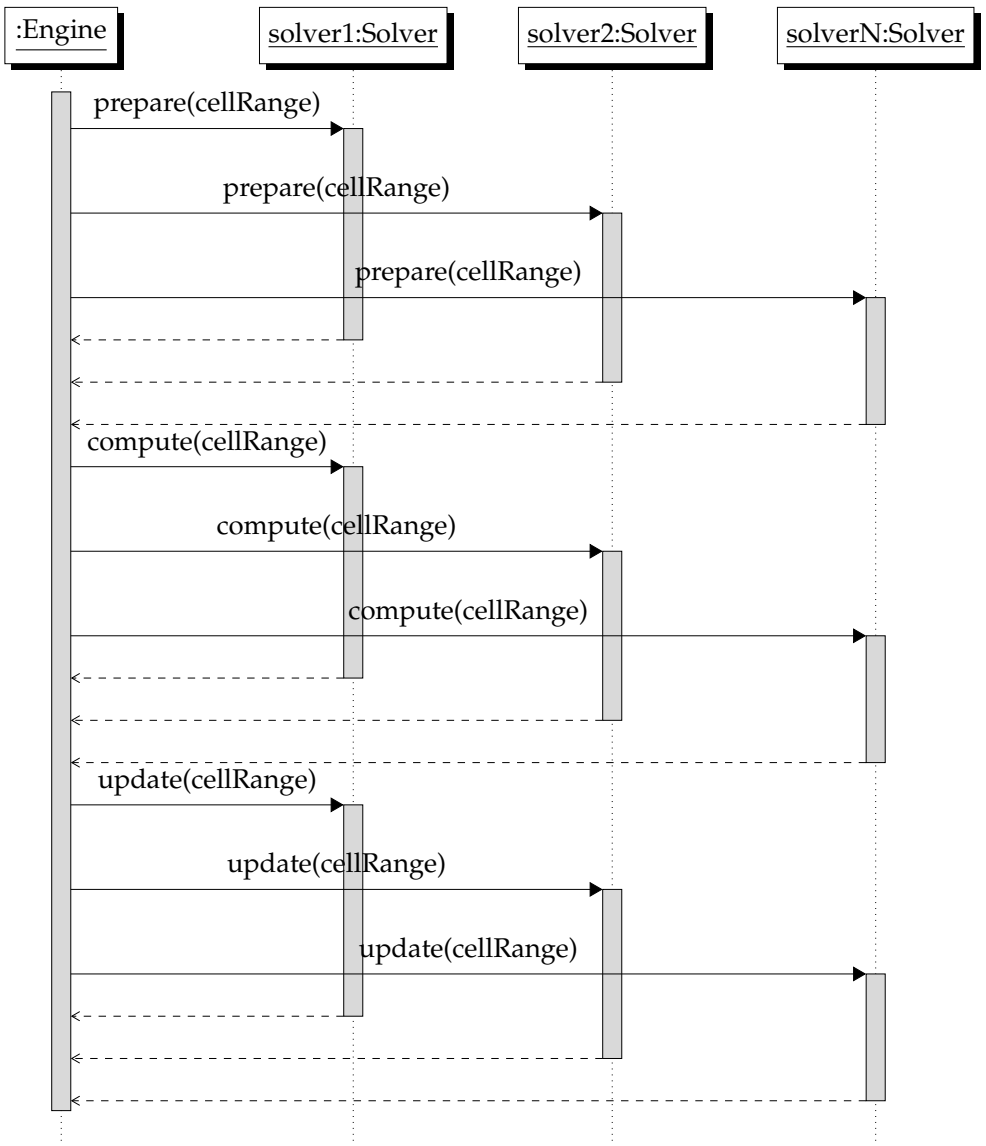


Figure 4.6: Parallel execution of the copies of a `Solver`

4.2.2 Solver

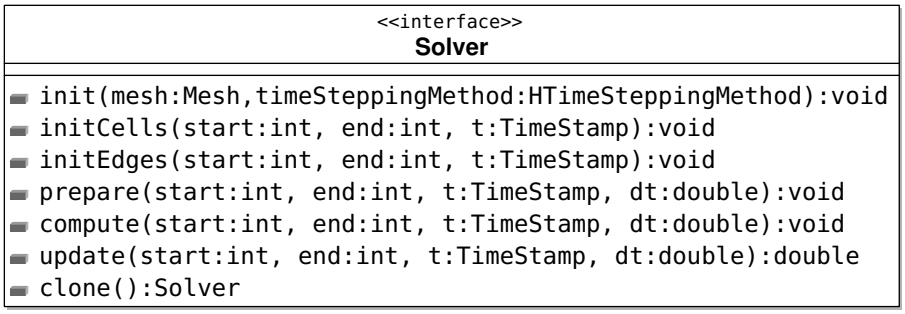


Figure 4.7: The `Solver` interface

The `Solver` interface is given in Figure 4.7. The `Solver` component encapsulates the cell-centered finite volume method and solves Equation 4.1 for a specified section and for all given physical processes. The mesh section is given by the indices of the first and last cell which have to be considered. Three initialization methods are called once in the beginning of a simulation: In the `init` method, second neighbor relations can be initialized if a second-order accurate numerical scheme is used. The `initCells` and `initEdges` methods set the initial conditions in the cells and, if necessary, at the edges. The methods are separated to allow a initialization of the edges based on the cell values. Similar to the `Engine` interface, the `Solver` interface consists of the three methods `prepare`, `compute` and `update`, too. Each method gets the specified mesh section, the current time and time step size as arguments. The `TimeInterval` is split up to a `TimeStamp` and `double` value for the time step size. Doing this, the conversion to the `double` value which is necessary many times in the numerical computations is done only once in each computational step and performance is increased. In addition, the interface includes the method `clone`, which creates a copy of the current instance.

The calculation of the terms in Equation 4.1 is delegated to one ore more `Scheme` implementations. The strategy design pattern is a common approach in object-oriented design for this situation. It is a behavioral design pattern intended to encapsulate algorithms and make them interchangeable (Gamma et al., 1995). Its UML representation is shown in Figure 4.8. An algorithm is not directly implemented in the current context, but an interface common to all implementations of the algorithm is defined (`Strategy` interface). The `Context` class is calling methods of the `Strategy` object it holds a reference to and which is configured at runtime with a concrete implemen-

tation, e. g. `ConcreteStrategyA`. In the context of the `Solver`, the `Scheme` interface represents the `Strategy` interface, and `ConcreteStrategyA` and `ConcreteStrategyB` are different implementations of the `Scheme` interface, e. g. schemes for the solution of the shallow water equations, transport equations etc. More details will be given in the prototype implementation in Section 4.3.4.

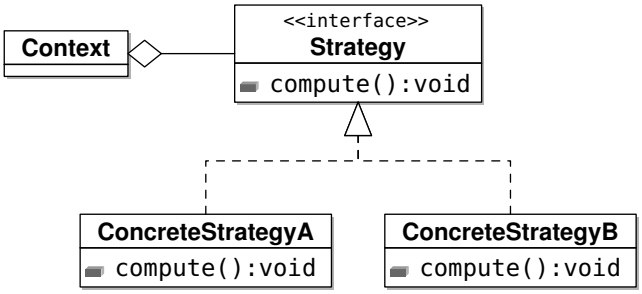


Figure 4.8: UML representation of the strategy pattern

4.2.3 Scheme

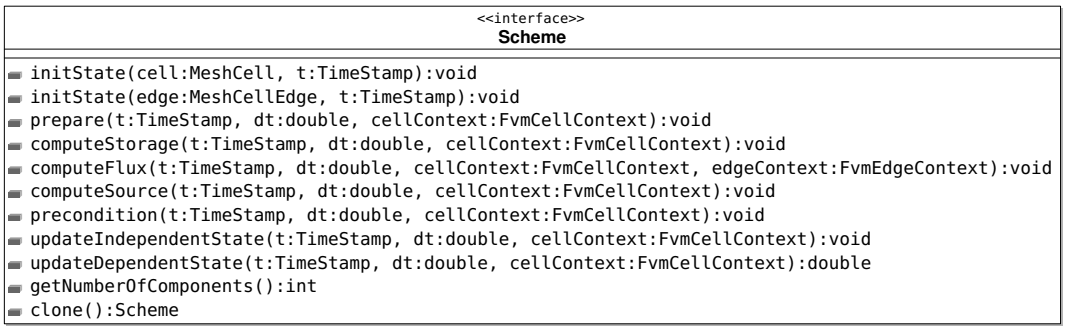


Figure 4.9: The `Scheme` interface

In Figure 4.9 the UML representation of the `Scheme` interface is given. This component handles the actual solution of the terms in Equation 4.1 for one specific physical process and for one given cell. Two initialization methods are called by the `Solver` once in the beginning of a simulation for each cell and edge of the computational mesh to set the initial conditions for the solution of the partial differential equations. Then the time loop is started and the computation of the storage or conserved state variables, flux and source terms is triggered. Afterwards, the preconditioning

method is called, to allow the manipulation of the terms of Equation 4.1 prior to the computation of the new conserved state variables. The update step is divided into updating independent and dependent state variables. The value of an independent variable can be computed independently from other processes involved in the simulation. As an example, this is the water depth at the new time step d^{n+1} in the shallow water equations (Equation 2.2). The value of a dependent variable depends on the values of other processes. This is e. g. the concentration in the transport equation (2.33) at the new time step c^{n+1} which is computed from the conserved state variable at the new time time step $(cd)^{n+1}$ and is therefore dependent on the water depth at the new time step d^{n+1} (Equation 2.33). To allow consistent computation of the state variables on the new time level, first, all state variables which are independent of other processes are computed and, second, all variables which are dependent on other processes are computed.

By calling the `getNumberOfComponents` method of the `Scheme` interface, the `Solver` gets information on the number of conserved state variables the `Scheme` computes. As in the `Solver` interface, the `clone` method returns a copy of the current instance.

The UML sequence diagram in Figure 4.10 shows the order of operations to solve the FVM problem for a new time step. For each cell of the given cell range, the `Solver` calls the `prepare`, `computeStorage`, `computeFlux`, `computeSource`, `precondition`, `updateIndependentState` and `updateDependentState` methods of the registered `Schemes` sequentially. Here, the procedure for a single thread is shown. For an multithread environment, Figure 4.10 has to be combined with Figure 4.6.

In Figure 4.9 several so-called context classes can be recognized in the arguments of the methods (not to be confused with the current context in the strategy pattern in Figure 4.8). The numerical algorithms encapsulated by these methods depend on a lot of different fields, as cell and edge properties, state variables and many more. To avoid long and unclear method signatures, so-called context objects can be used (Kelly, 2004). Context classes are only used to store values, which can be accessed by get and set methods. There is no computational logic in theses classes. In the `Scheme` interface, context interfaces are used for cell values (`FvmCellContext`) and for edge values (`FvmEdgeContext`) (Figure 4.11).

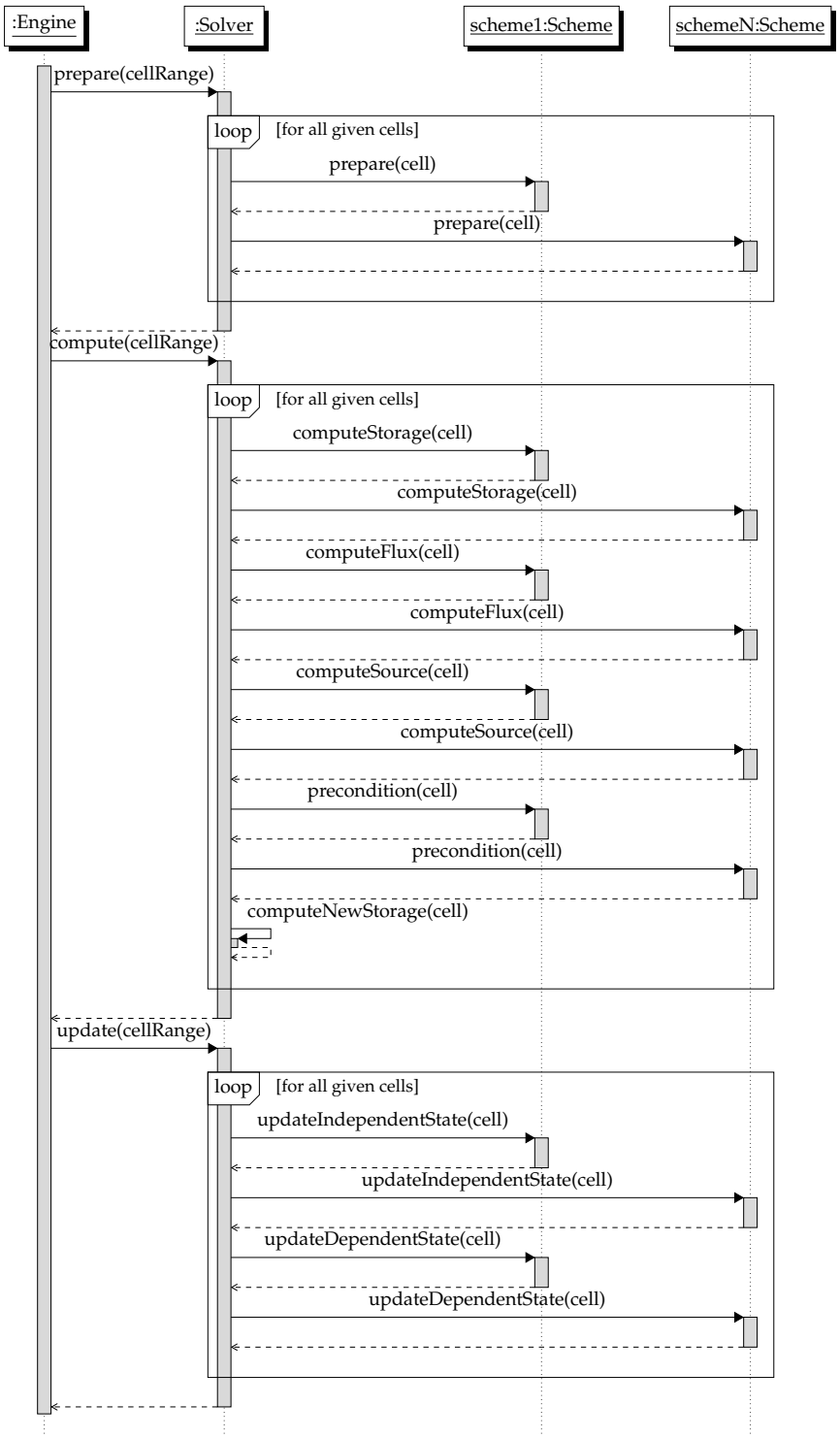


Figure 4.10: Sequential solution of FVM problem

Figure 4.11: Interface definitions of `FvmCellContext` and `FvmEdgeContext` (truncated)

4.3 Prototype implementation of numerical framework

On the basis of the presented conceptual software architecture a prototype of the numerical framework was developed within the existing implementation of `hms` as a proof of concept. As with the existing implementation, the object-oriented Java programming language was used. The first public implementation of the Java language was presented in 1995 (Wikipedia, 2017*a*). Since then, there is a continuous development and extension by new programming concepts. A key feature of Java is the platform independency of compiled code. This is achieved by running the compiled code on a Java virtual machine native to the used system architecture. The prototype was implemented based on the Java Standard Edition (SE) Development Kit 7 (Oracle, 2018).

In this chapter, details on the prototype implementation are given.

4.3.1 The `numerics` package

The top-level package structure of `hms` was already shown in Figure 4.1. The `numerics` package is a nested package of the `core` package. It strongly depends on the `terma` package and the `shape` subpackage of the core. Its structure and content are designed based on the concepts presented in Section 4.2. The package is subdivided into five subpackages to sort the classes according to their context. Figure 4.12 shows an overview of the package structure. In the following the subpackages will be described. As the adaptive quadtree grid is not in the scope of this thesis, the `adaption` package will be skipped in the description. Details are given by Simons (2008).

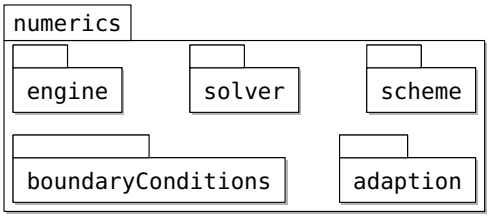


Figure 4.12: The `numerics` package

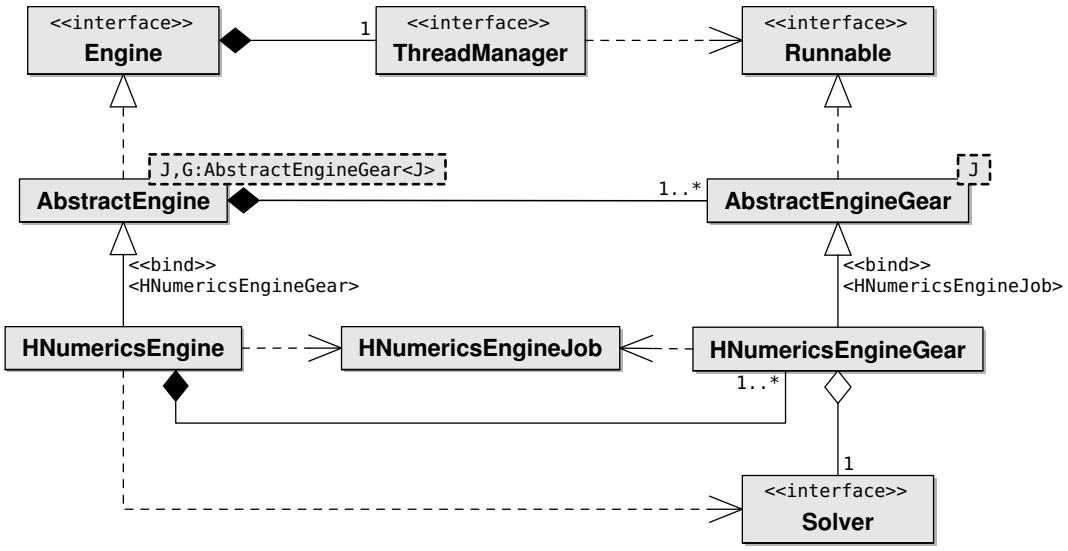


Figure 4.13: UML representation of the `Engine` implementation

4.3.2 The `engine` package

The `engine` package consists of all classes used for the prototype implementation of the `Engine` interface presented in Section 4.2.1. Figure 4.13 shows an UML class diagram of the `Engine` implementation. In here, it is shown:

- the `Engine` interface itself (cf. Figure 4.5 in Section 4.2.1). In the prototype implementation the `Engine` has a composition association to the `ThreadManager` interface.
- the `ThreadManager` interface and its implementation are part of the `terma.util` package. The `ThreadManager` is dependent on the `Runnable` interface which is part of the Java standard library and marks classes which intend to be executed by a thread.
- the `AbstractEngine` class is an abstract generic implementation of the `Engine` interface (Figure 4.14). Depending on the available threads on the computer, it is composed by one or more generic types `G` extending the `AbstractEngineGear`. The `AbstractEngineGear` is a skeleton implementation of the `Runnable` interface. The job to execute is defined by the generic type `J`. So far, the implementation is independent of the FVM. The `AbstractEngine` can be used as a base for arbitrary implementations of the `Engine` interface which run any job on any item count which can be distributed on multiple threads.

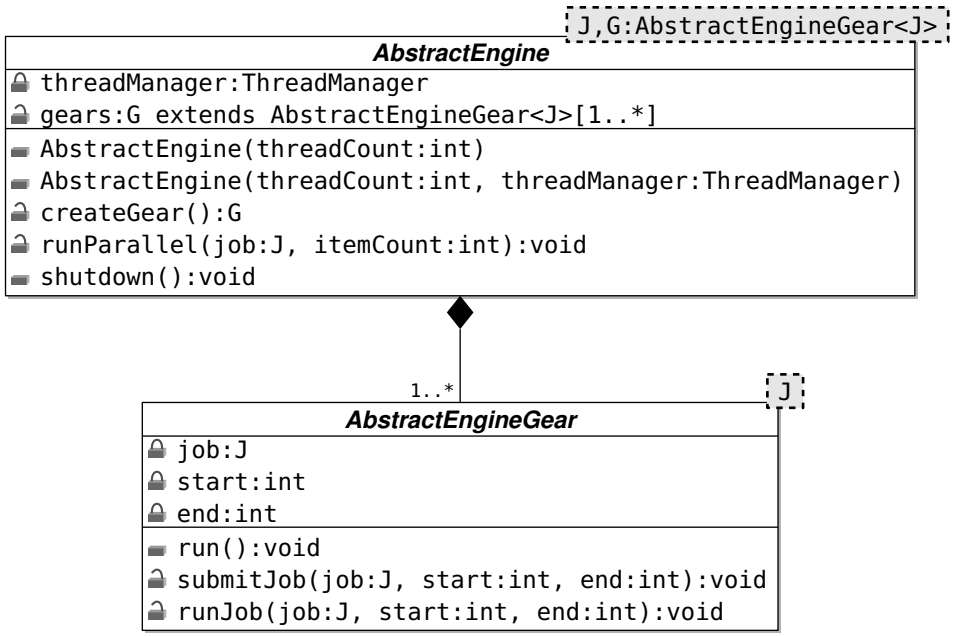


Figure 4.14: Class diagram of `AbstractEngine` and `AbstractEngineGear`

- the class `HNumericsEngine` binds `G` to the class `HNumericsEngineGear` and `J` to the class `HNumericsEngineJob` (Figure 4.15). It is an `Engine` for solving FVM problems and will control the execution of the `Solver` methods. The `HNumericsEngineGear` class has an aggregation association to a `Solver` object. In the prototype implementation the `HNumericsEngineJob` is realized as an enumeration to define which task should be executed. Each `HNumericsEngineGear` determines the desired size for the next time step in the `update` method. This time step size is forwarded to the `HNumericsEngine` via the `setTimestep` method in the `HNumericsEngine` class. To avoid concurrency exceptions in the parallel runtime environment the Java concept of synchronized methods is used. Methods with the `synchronized` keyword in the signature cannot be invoked by multiple threads at the same time. The threads have to wait on each other before calling these methods.

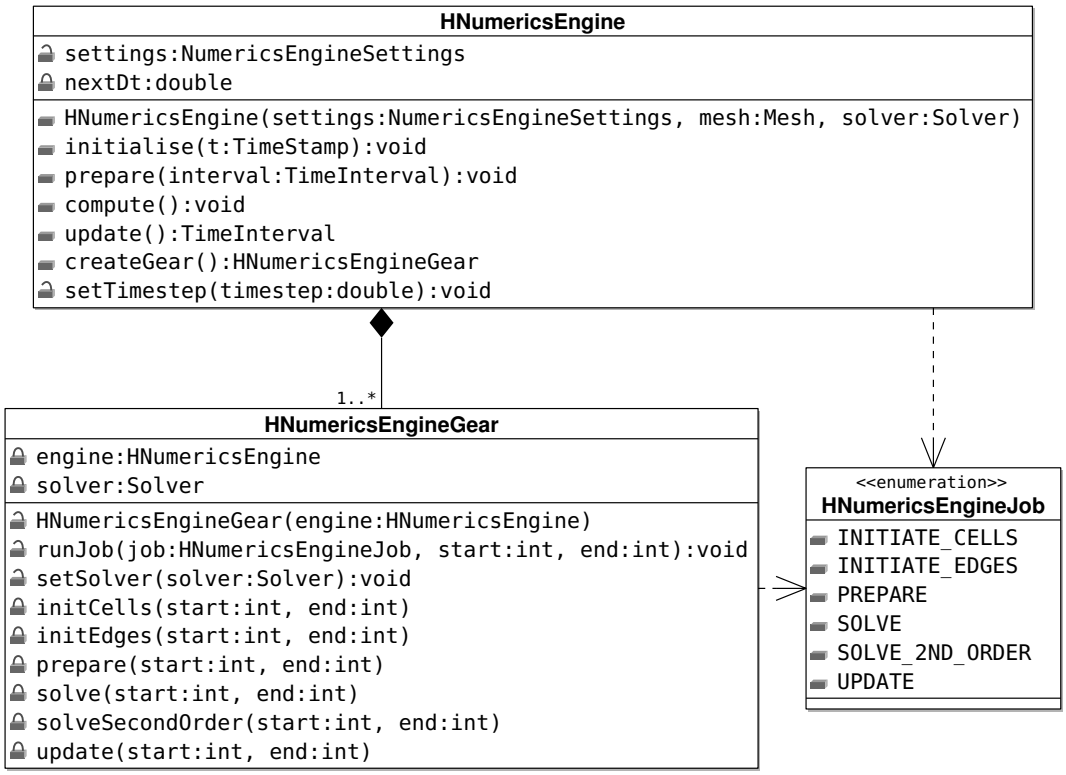


Figure 4.15: Class diagram of `HNumericsEngine`, `HNumericsEngineGear` and `HNumericsEngineJob`

4.3.3 The `solver` package

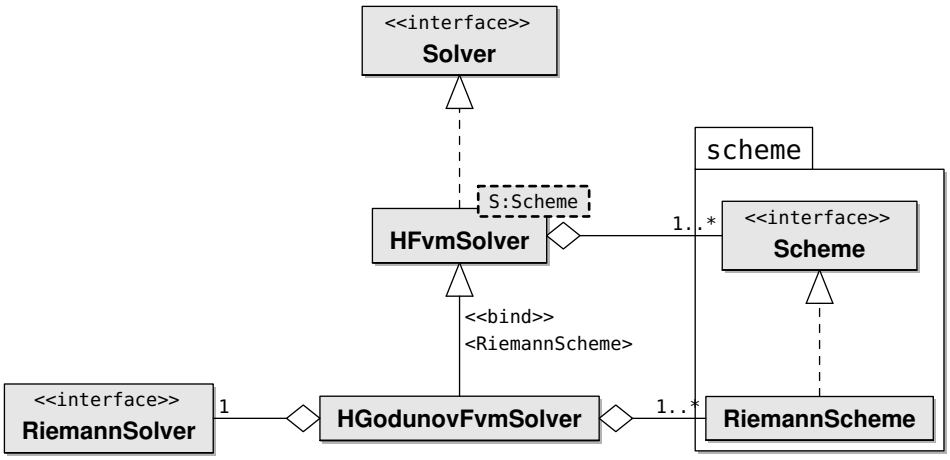


Figure 4.16: UML representation of the `Solver` implementations

The `solver` package consists of all classes used for the prototype implementation of the `Solver` interface presented in Section 4.2.2. Figure 4.16 shows an UML class diagram of the `Solver` implementation. In here, it is shown:

- the `Solver` interface as defined in Section 4.2.2 (cf. Figure 4.7).
- the `HFvmSolver` class, which is an implementation of the `Solver` interface. It maintains references to one or more generic types `S` extending the `Scheme` interface. The class diagram of the `HFvmSolver` and `HGodunovFvmSolver` is shown in Figure 4.17. The `addScheme` method allows adding schemes in the configuration phase of the `HFvmSolver` instance.
  The solution step defined by the `compute` method is subdivided into the solution of storage or conserved state variables, flux and source terms. The actual computation of these terms is delegated to one or multiple `Schemes` in the `scheme` package (Section 4.3.4). A code snippet is shown in Listing 4.1.

  Listing 4.1: Condensed implementation of `compute` method in `HFvmSolver` class

```
1 @Override
2 public void compute(int start, int end, TimeStamp t,
3   double dt) {
4   for (int i = 0; i < cellContexts.length; i++) {
5     for (int s = 0; s < schemes.size(); s++) {
```

```
6         schemes.get(s).computeStorage(t, dt, context);
7       }
8
9       MeshCell cell = cellContext.getCell();
10      for (int e = 0; e < cell.getCellEdgeCount(); e++) {
11        // prepare edge context ...
12        for (int s = 0; s < schemes.size(); s++) {
13          schemes.get(s).computeFlux(t, dt, cellContext,
14                  edgeContext);
15        }
16      }
17      // ...
18    }
19  }
```

- the `HGodunovFvmSolver` class, which has an aggregation association to a `RiemannSolver` instance and binds the generic type `S` to a `RiemannScheme`. The `HGodunovFvmSolver` triggers the preparation of the `RiemannProblem` and calls a `RiemannSolver` to solve the given problem.
- the `RiemannSolver` interface: The different Riemann solvers presented in Chapter 3 are implemented as realizations of this interface. The class diagram of the interface is given in Figure 4.18. The `solve` method has the flux term and a `RiemannProblem` context object describing the Riemann problem itself as arguments. The UML class diagram in Figure 4.19 shows all currently implemented Riemann solvers.
- The `clone` methods in the `Solver` and `RiemannSolver` components are implemented using copy constructors: In a parallel runtime environment, each computational thread needs access to the numerical algorithms and data. Read-only data, as the mesh, settings and boundary conditions, can be shared among the threads. However, all objects which change their state in a computational cycle have to be copied for each thread. These are for example the `Solver`, `Scheme` and `Context` objects. All Java classes have a `clone` method. The default implementation will return a so-called shallow copy of the object. That means, the object itself and all primitive fields (e. g. `double` and `boolean` values) are copied. However, a complex field will still reference to the original object. For parallel computations not only the object itself has to copied, but

all objects it depends on, too. For example, when copying the `Solver` object, a copy of all corresponding `Schemes` and `Context` objects is necessary. This is called a deep copy.
To simplify this process, copy constructors can be implemented. These are special constructors which get an object of the same class type as the considered class as an argument. The implemented constructor copies the considered object itself and calls the copy constructors of the dependent objects. In the Java language, the default `clone` method can be overridden to make use of the implemented copy constructor. An example implementation is shown in the Listing 4.2.

Listing 4.2: Example implementation of a copy constructor

```
1  public class ExampleClass {
2
3    private double exampleValue;
4    private OtherClass otherObject;
5
6    /** Standard constructor **/
7    public ExampleClass(double value, OtherClass o) {
8      this.exampleValue = value;
9      this.otherObject = o;
10   }
11
12   /** Copy constructor **/
13   protected ExampleClass(ExampleClass rhs) {
14     // copy from right-hand side
15     this.exampleValue = rhs.exampleValue;
16     this.otherObject = rhs.otherObject.clone();
17   }
18
19   /** Override default Java clone method **/
20   @Override
21   public ExampleClass clone() {
22     return new ExampleClass(this);
23   }
24 }
```

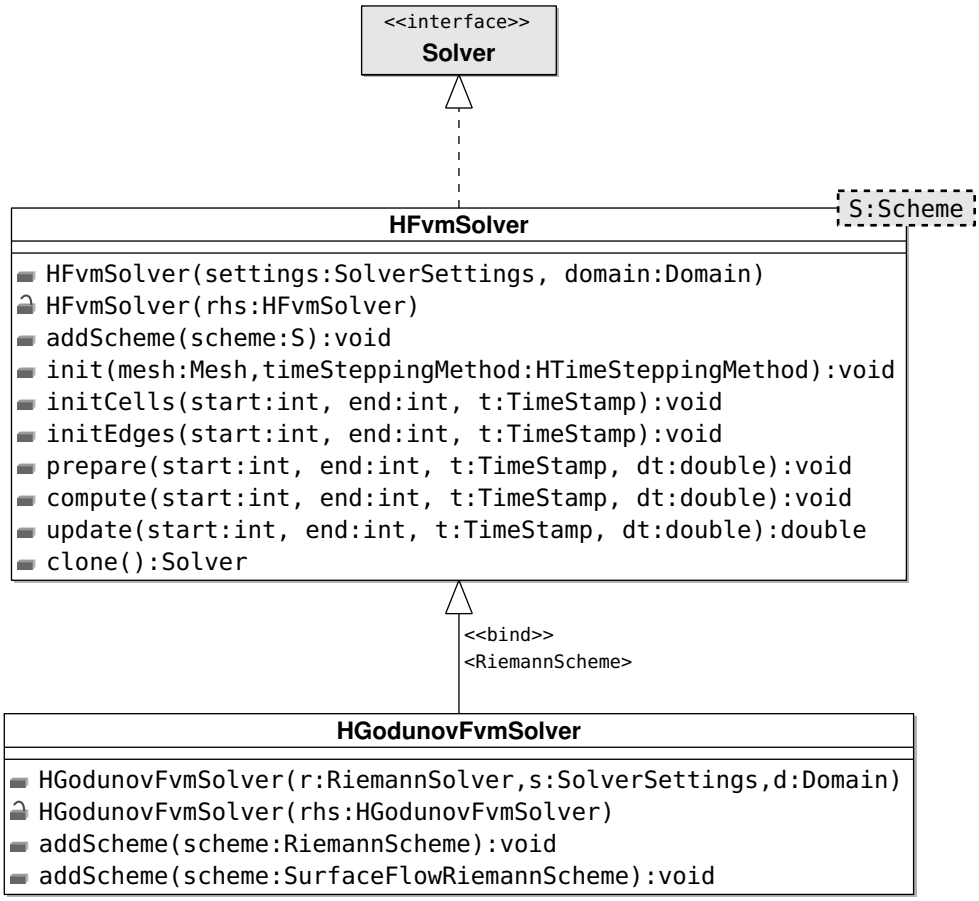


Figure 4.17: Class diagram of `HFvmSolver` and `HGodunovSolver`

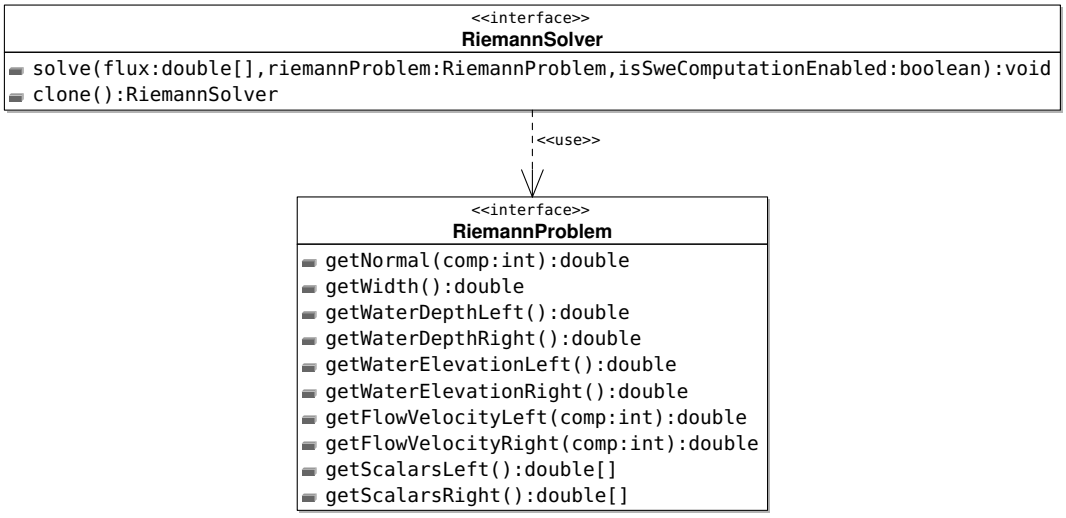


Figure 4.18: Class diagram of `RiemannSolver` and `RiemannProblem`

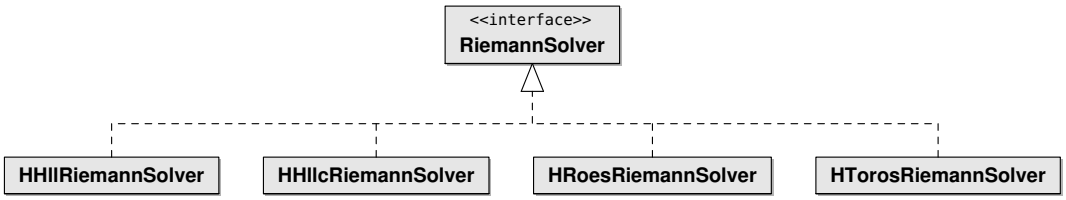


Figure 4.19: UML representation of the `RiemannSolver` implementations

4.3.4 The `scheme` package

The `scheme` package consists of all classes used for the prototype implementation of the `Scheme` interface presented in Section 4.2.3. Figure 4.20 shows an UML class diagram of the `Scheme` implementations. In here, it is shown:

- the `Scheme` interface as shown in Section 4.2.3 (cf. Figure 4.9) and a derived `RiemannScheme` interface. As described in Section 4.2.3, long and unclear method signatures are avoided by using context classes in the `Scheme` interface. In Figure 4.21 the UML class diagrams of the implementations of the `FvmCellContext` and `FvmEdgeContext` interfaces are given, respectively. In the interface only get methods were defined. In the actual implementations set methods are added to be able to modify the class properties at runtime.
- the `RiemannScheme` interface: As seen in Figure 4.22, a class realizing the `RiemannScheme` interface has to provide two additional methods. The `computeRiemannStates` method instructs the scheme to compute the Riemann states necessary for the solution of the Riemann problem. Here, a `FvmRiemannContext` context object implementing the `RiemannProblem` interface is given as an argument. For solving the Riemann problem in the surface water flow context, the water depth and flow velocities are essential. In a coupled simulation, these will be provided by a surface water flow scheme. However, to be able to test a scheme independently from a surface water flow scheme, a `RiemannScheme` can be instructed to compute its own flow states using the `setComputeFlowStatesEnabled` method.
- the `AbstractFiniteVolumeScheme` is a skeleton implementation of the `Scheme` interface. It implements methods which are common to all derived `Schemes`. The same is true for the `AbstractSoilScheme` class. As the computation of the infiltration into the vadose zone differs strongly from the typical solution of the conservation law (cf. Chapter 2), this scheme is not derived from the `AbstractFiniteVolumeScheme`.
- the `AbstractSurfaceFlowScheme`, `AbstractTransportScheme` and `AbstractMorphologyScheme` classes: For surface water flow, transport and morphology another layer of abstraction implements all methods beside the computation of the flux term, as the computational strategy differs in the derived implementations. Here, the second-order accurate determination of face values using the MUSCL method presented in Section 3.3 is implemented.

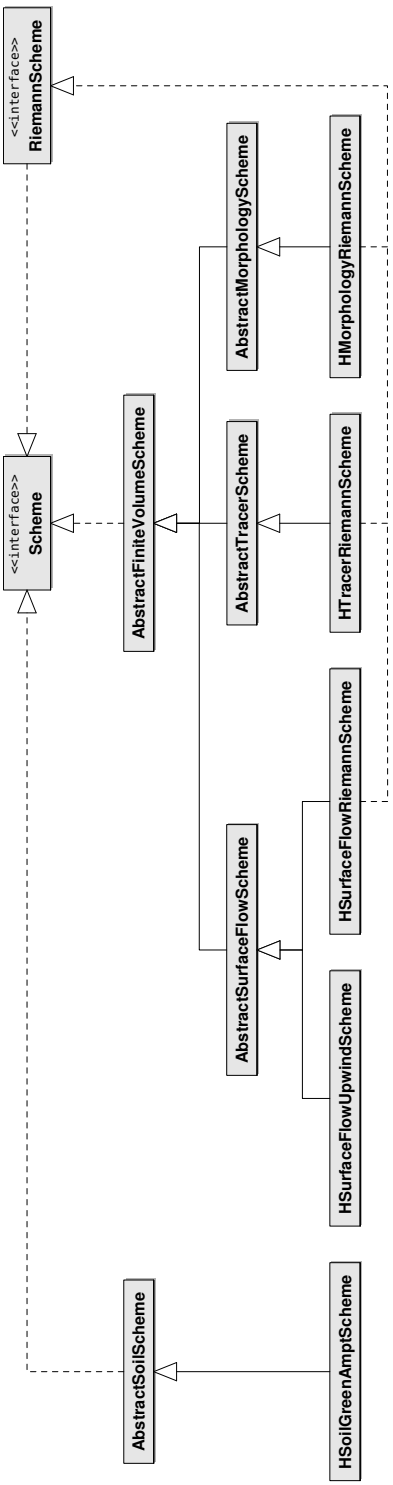
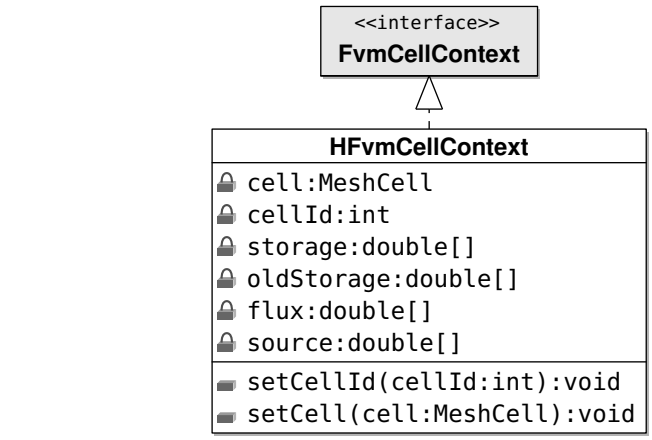
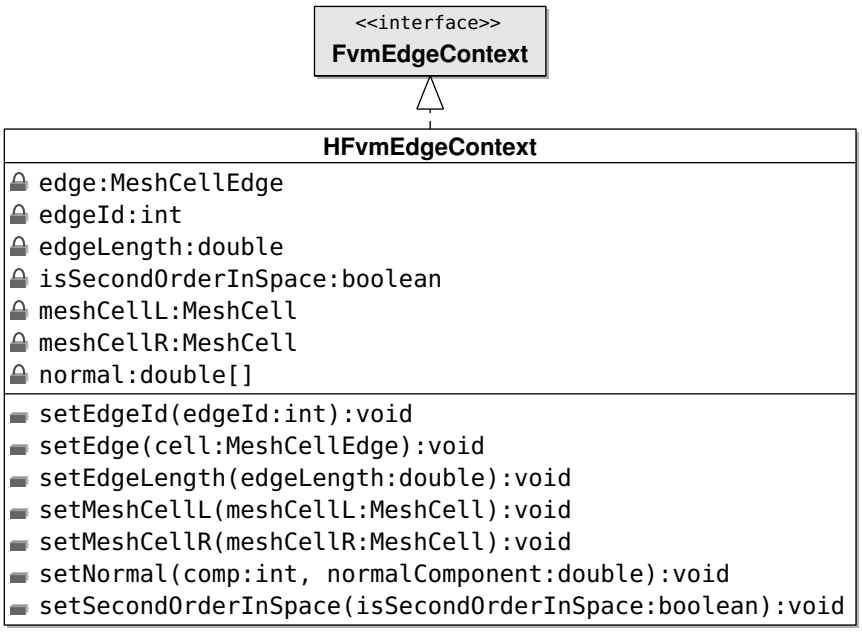


Figure 4.20: UML representation of the Scheme implementations



(a)



(b)

Figure 4.21: Class diagrams of `FvmCellContext` and `FvmEdgeContext` context classes

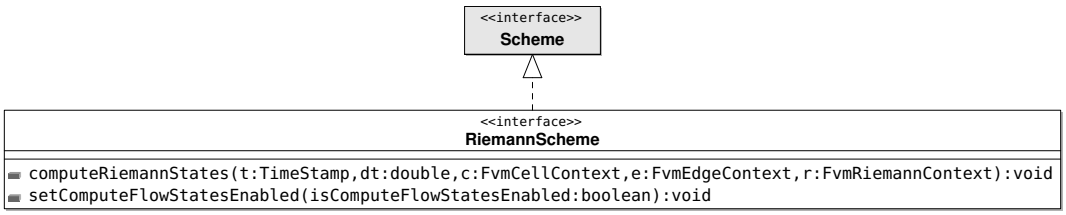


Figure 4.22: Class diagram of the `RiemannScheme` interface

- In the lowest layer in the UML class diagram, the concrete implementations are shown. They are different in their strategy to compute the numerical flux term. The `HSurfaceFlowUpwindScheme` implements the first-order upwind method shown in Section 3.2.1. If the flux is computed using a Riemann solver, the schemes also implement the `RiemannScheme` interface. The `HSurfaceFlowRiemannScheme`, `HTracerRiemannScheme`, and `HMorphologyRiemannScheme` implement flux calculations based on the approximate Riemann solver shown in Section 3.2.4.

4.3.5 The `boundaryConditions` package

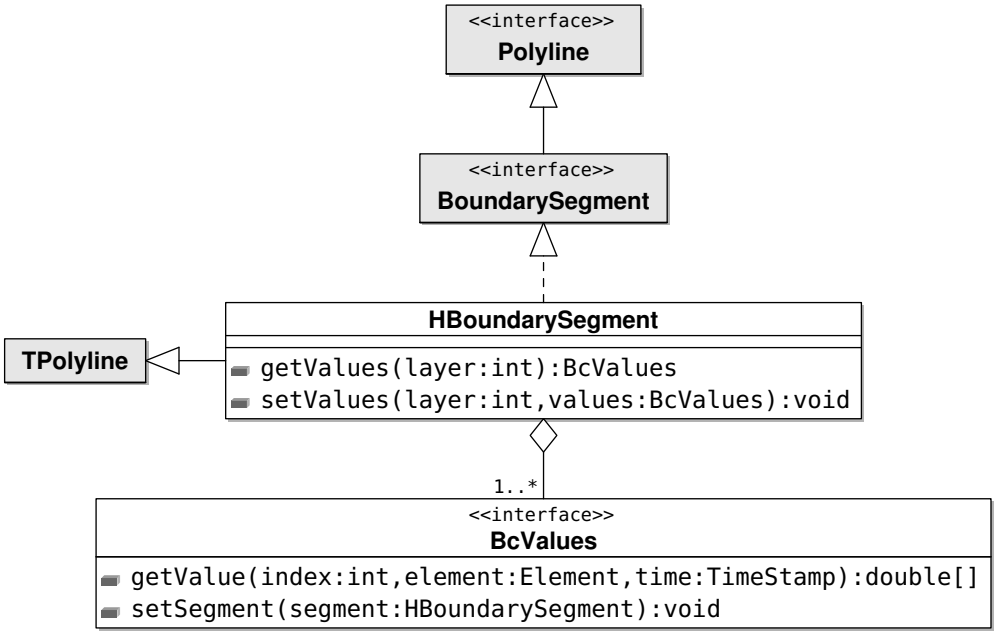


Figure 4.23: UML representation of the `boundaryConditions` package

The `boundaryConditions` package consists of all classes necessary for the boundary handling in the prototype implementation. Figure 4.23 shows an UML class diagram of the package content. In here, it is shown:

- the `BoundarySegment` interface, which is a specialization of the `Polyline` interface of the `geometry` package and part of the `layer` package in the core of `hms`.
- the `HBoundarySegment` class, which is derived from the `TPolyline` class from the `geometry` package and implements the `BoundarySegment` interface. It implements the `getValues` method defined by the `BoundarySegment` interface and defines a corresponding `setValues` method to get and set the boundary values at the considered segment for a specified layer.
- the `BcValues` interface: This interface is realized by all boundary condition implementations. In `hms`, the most common boundary conditions types are already implemented. Figure 4.24 shows existing implementations for the surface water flow numerics. These can be used directly or can be customized by overriding the implementations. For example the provided implementation

of a fixed water depth boundary condition can be overridden to allow for a time dependent water depth. In addition own implementations of the `BcValues` interface are possible.

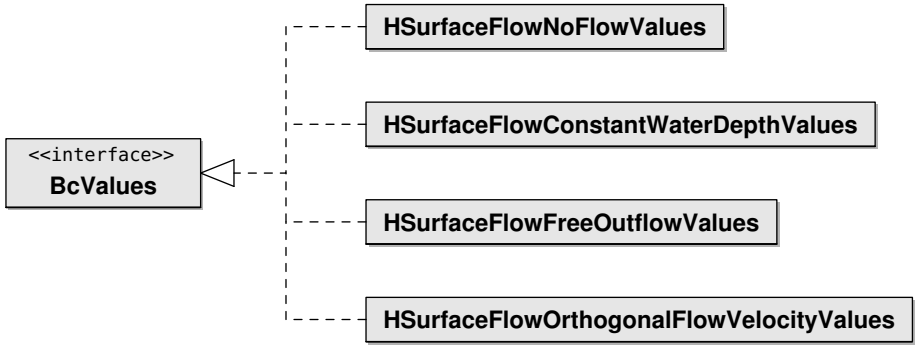


Figure 4.24: UML representation of exisiting `BcValues` implementations for the surface water flow numerics

4.3.6 Settings

All `Engine`, `Solver` and `Scheme` implementations depend on various settings which have to be defined at runtime. These are e. g. the maximum thread count for the `Engine`, if a computation should be carried out second-order in space for the `Solver`, the source term for the `Scheme` and the Chézy coefficient in case of the `SurfaceFlowScheme`. Interfaces with the corresponding get methods were defined for all settings. In addition, default implementations were given for all interfaces which define set methods, too. To implement specialized settings, e. g. a spatial varying roughness or complex time-dependent source terms, the default implementations of those methods can be overridden. In the settings and also throughout the `hms` framework only standard SI units are used.

All default implementations were annotated using the Java Architecture for XML Binding (JAXB). The Extensible Markup Language (XML) is a standardized textual data format which allows to describe structured data in human-readable text files. The JAXB framework binds automatically the Java classes to the XML structure by using annotations in the Java source code. It is not necessary to write or to parse the XML files manually. In Listing 4.3 a part of the annotated `DefaultSurfaceFlowSettings` class is shown.

Listing 4.3: Annotated fields of the `DefaultSurfaceFlowSettings` class using the JAXB framework

```
1  import javax.xml.bind.annotation.*;
2
3  @XmlAccessorType(XmlAccessType.NONE)
4  @XmlRootElement(name = "SurfaceFlowSettings")
5  public class DefaultSurfaceFlowSettings implements
6        SurfaceFlowSettings {
7
8     @XmlAttribute(required = true)
9     protected final static String version = "2012-06-18";
10
11    @XmlElement(required = true, defaultValue = "9.81")
12    protected double gravity = 9.81;
13
14    @XmlElement(required = true, defaultValue = "false")
15    protected boolean isFrictionEnabled = false;
16
17    ...
18 }
```

Listing 4.4 shows writing/reading an instance of this class to/from an XML file using the JAXB framework. For simplicity, the check of the casting operation in line 15 is not shown.

Listing 4.4: Simplified example for writing/reading an instance the `DefaultSurfaceFlowSettings` class to/from XML file

```
1  // initialize settings
2  DefaultSurfaceFlowSettings settings =
3   new DefaultSurfaceFlowSettings();
4
5  // initalize JAXB context
6  JAXBContext context = JAXBContext.newInstance(obj.getClass());
7
8  // write settings to XML file
9  Marshaller m = context.createMarshaller();
10 m.marshal(settings, new File("path/to/mySettings.xml"));
```

```
11
12 // read settings from file
13 Unmarshaller um = context.createUnmarshaller();
14 DefaultSurfaceFlowSettings settingFromFile =
15  (DefaultSurfaceFlowSettings)um.unmarshal("path/to/mySettings.xml");
```

In Listing 4.5 the XML representation of an instance of the `DefaultSurfaceFlowSettings` class is shown.

Listing 4.5: XML representation of an instance of the `DefaultSurfaceFlowSettings` class

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <SurfaceFlowSettings version="2014-07-11">
3      <gravity>9.81</gravity>
4      <isFrictionEnabled>false</isFrictionEnabled>
5      <frictionApproach>STRICKLER</frictionApproach>
6      <frictionCoefficient>40.0</frictionCoefficient>
7      <isDiffusionEnabled>false</isDiffusionEnabled>
8      <turbulentViscosity>0.0</turbulentViscosity>
9      <isTurbulenceModelEnabled>false</isTurbulenceModelEnabled>
10     <isSourceEnabled>false</isSourceEnabled>
11     <massSource>0.0</massSource>
12     <momentumSource>0.0 0.0</momentumSource>
13     <maxCourantNumber>0.5</maxCourantNumber>
14     <limiterFunction>MIN_MOD</limiterFunction>
15     <isFroudeDependentBcEnabled>true</isFroudeDependentBcEnabled>
16     <waterdepthDry>1.0E-8</waterdepthDry>
17     <isVariableBedEnabled>false</isVariableBedEnabled>
18 </SurfaceFlowSettings>
```

4.4 Setting up applications

On the basis of the presented software framework, applications for the simulation of water and environment related problems can be developed. In the following, two sample applications are given. In the listings minimal examples are shown. Features

of hms, as system and layer management, online visualization and monitoring the simulation results are not used here to focus on the essentials necessary to build an application.

4.4.1 First-order upwind shallow water simulation

In the first example a simulation based on the solution of the shallow water equations is shown. The first-order accurate upwind scheme and the forward Euler method are used in the simulation. In Listing 4.6 the minimal code to build this application is given.

Listing 4.6: First-order upwind shallow water simulation

```
1  // initialize domain
2  Domain domain = new MyDomain();
3
4  // generate regular mesh with 10m cell size
5  Mesh mesh = ShapesFactory.getRegularGrid(domain, 10);
6
7  // initialize settings
8  NumericsSettings settings = new DefaultNumericsSettings()
9
10 // initialize solver
11 HFvmSolver<Scheme> solver = new HFvmSolver<>(settings, domain);
12
13 // add first-order upwind surface water flow scheme to solver
14 solver.addScheme(
15   new HSurfaceFlowUpwindScheme(new DefaultSurfaceFlowSettings()));
16
17 // initialize engine
18 Engine engine = new HNumericsEngine(settings,mesh,solver);
19
20 // set initial time step to 0.1s
21 TimeInterval time = new HTimeInterval(0.1);
22
23 // run time loop until 10s are reached
24 while (time.getStart() <= 10.) {
25   engine.prepare(time);
```

```
26    engine.compute();
27    time = engine.update();
28  }
```

4.4.2 Coupled shallow water and transport simulation

In the second example, a coupled simulation of shallow water and transport processes is shown (Listing 4.7). For both, shallow water and transport simulation the HLLC schemes are used. Second-order accuracy in space is achieved by using the minmod TVD limiter. Both `Schemes` are added to an instance of the `HGodunovFvmSolver`.

Listing 4.7: Coupled shallow water and transport simulation

```
1  // initialize domain
2  Domain domain = new MyDomain();
3
4  // generate regular mesh with 10m cell size
5  Mesh mesh = ShapesFactory.getRegularGrid(domain, 10);
6
7  // initialize settings
8  NumericsSettings settings = new DefaultNumericsSettings()
9
10 // set second-order in space
11 settings.setSecondOrderInSpace(true);
12
13 // initialize solver
14 HGodunovFvmSolver solver = new HGodunovFvmSolver(
15   new HHllcRiemannSolver(), settings, domain);
16
17 // add Riemann surface water flow scheme to solver
18 solver.addScheme(
19   new HSurfaceFlowRiemannScheme(new DefaultSurfaceFlowSettings()));
20 // add Riemann transport scheme to solver
21 solver.addScheme(
22   new HTracerRiemannScheme(new DefaultTracerSettings()));
23
```

```
24 // initialize engine
25 Engine engine = new HNumericsEngine(settings,mesh,solver);
26
27 // set initial time step to 0.1s
28 TimeInterval time = new HTimeInterval(0.1);
29
30 // run time loop until 10s are reached
31 while (time.getStart() <= 10.) {
32   engine.prepare(time);
33   engine.compute();
34   time = engine.update();
35 }
```

4.5 Conclusions

In the beginning of this chapter an introduction to the infrastructure and concepts of `hms` has been given. The existing framework has been extended by a new numerical framework. Three base components were described: `Engine`, `Solver` and `Scheme`. The `Engine` component encapsulates all numerical methods and handles their parallelization. The `Solver` component is an explicit cell-centered finite volume solver, based on the mathematical concepts presented in Chapter 2 and the numerical methods presented in Chapter 3. The actual computations for a physical process are encapsulated in a `Scheme` component. The new software architecture aims for supporting prototyping of numerical algorithms by allowing a flexible extension by new physical processes and numerical methods.

Based on the presented software architecture `hms` supports three different coupling mechanisms, where each of them has its own advantages and disadvantages and can be explained by means of the transport model: First, the transport model can stand on its own. This means, there is no computation of the flow velocity and water depth. The flow field has to be given as constant or time-dependent values in the setup of the simulation. This allows independent testing of a the implemented model. Secondly, using the OpenMI standard, a coupling of different layers and even with different softwares is possible. It is a loose coupling, which means the numerical algorithms are developed independently. The third coupling mechanism integrated in the new software architecture allows direct coupling without destroying the

functionality of the two other mechanisms. Here, the same solver can be used for coupled models which allows a more efficient computation.

Then details on the prototype implementation of the conceptual software architecture were given. Object-oriented programming allows a simplified implementation of the concepts with different levels of abstraction, which enable a flexible and extendible software framework. Several different numerical schemes were implemented on the basis of the prototype implementation using either the plain FVM method or Godunov's method and the principal functionality of the software architecture could be proved. In two minimal examples the application of the software framework was demonstrated. In the remaining chapters, the implemented numerical algorithms are verified and applied to analytical test cases and engineering applications. As a proof of the software concept, in Section 5.6 the implementation of the sediment transport and morphology equations (Section 2.5) is explained.

5 Model verification

In the following chapter several test cases verify the accuracy of the implemented numerical algorithms. The test cases involve different flow conditions i. e. sub-, trans- and supercritical flow, discontinuities, very low water depth and wetting/drying processes. To obtain reliable results on the accuracy only test cases with analytical solutions were chosen. In all test cases, turbulence was neglected. After verification the numerical model can be used for problems, which can be solved only by numerical simulation. In addition, a proof of the software concept presented in Chapter 4 is carried out. As an example, the implementation of the mathematical model of sediment transport and morphological evolution (Section 2.5) is shown in the last test case.

The rainfall-runoff test case investigated in Section 5.3 is in large parts extracted from Simons et al. (2014).

5.1 1D hydraulic jump

MacDonald et al. (1997) developed several sophisticated analytic benchmark solutions for open-channel flows. The analytic solutions to these test problems were obtained by constructing inverse problems. That means, first, the channel geometry, the discharge and a smooth water depth profile were defined and then the corresponding bed slope was determined. The details of this process and the resulting analytic solutions are given by MacDonald et al. (1997).

Here, the results of the implemented algorithms for their test problem 4 are shown. This test case consisted of a 1000 m long channel with a width of 10 m. The discharge was $20\,\mathrm{m^3\,s^{-1}}$ and the Manning's roughness coefficient was $0.02\,\mathrm{s\,m^{-1/3}}$. The flow is subcritical at inflow and critical at $1/3$ distance. There is a hydraulic jump at $2/3$ distance and the outflow is again subcritical.

For the numerical computation, the domain was discretized with a regular grid and a cell size of 10 m, 5 m and 2 m. A constant orthogonal specific discharge of $2\,\mathrm{m^2\,s^{-1}}$

was set as a boundary condition for the upstream boundary and a constant water depth of 1.35 m was set at the downstream end. All other boundaries are solid. The initial water depth was set to 2 m and the initial flow velocity was $0\,\mathrm{m\,s^{-1}}$. The CFL criterion was set to 0.3 and the time step was computed adaptively using Equation 3.15. The results of the first- and second-order accurate HLLC schemes (Section 3.2.4) were compared. Second-order accuracy was achieved by using the minmod TVD limiter (Equation 3.82) and the improved Euler time stepping method (Section 3.1.2). The simulation was run until steady state of the water depth was reached with a remaining variation of $1 \cdot 10^{-8}$ m.

5.1.1 Results

In Figure 5.1 the free surface profile at steady state for the first- and second-order accurate schemes are given. Figure 5.1a shows the results for the regular grid with a cell size of 10 m. Both, the first- and second-order scheme smoothed the hydraulic jump. The results for the grid with a cell size of 2 m showed very good agreement with the exact solution for both schemes (Figure 5.1b). Table 5.1 shows the absolute (L^1) errors computed from the deviation of exact and simulated values of the water elevation of all cells N:

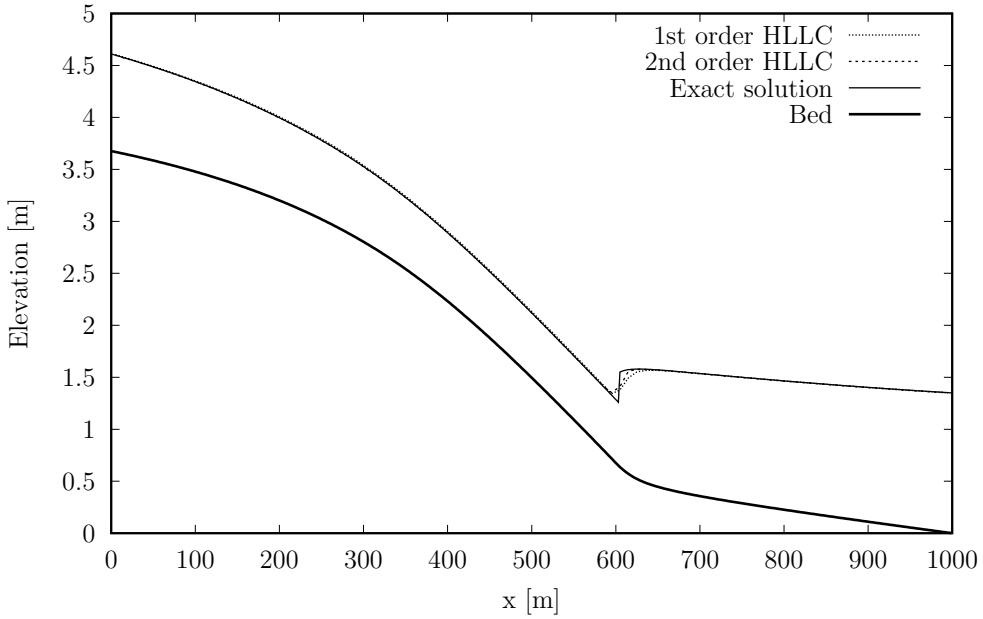
$$L^1 = \frac{\sum_{i=1}^{N} \left(A_i \cdot |\mathbf{q}_{i,\text{exact}} - \mathbf{q}_{i,\text{simulated}}| \right)}{\sum_{i=1}^{N} A_i} \tag{5.1}$$

To characterize the flow regime, the dimensionless Froude number can be used. It is defined by the ratio of flow velocity to wave speed:

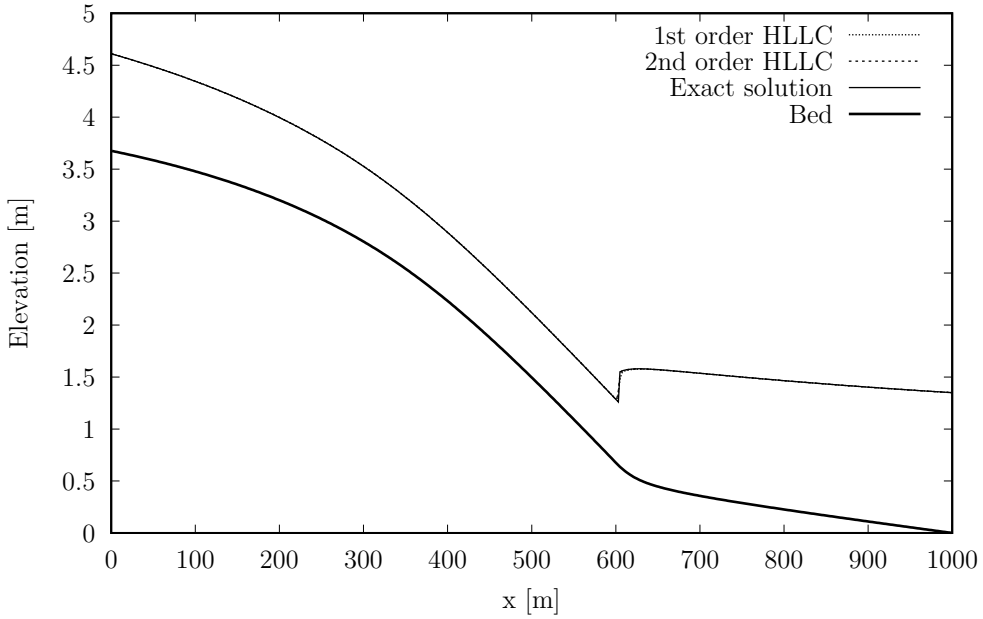
$$\mathrm{Fr} = \frac{|\mathbf{v}|}{\sqrt{g \cdot d}} \tag{5.2}$$

Table 5.1: L^1-errors (m) of free surface profile for first- and second-order accurate HLLC scheme

cell size	2nd-order	1st-order
10 m	$3.1 \cdot 10^{-3}$	$9.9 \cdot 10^{-3}$
5 m	$1.3 \cdot 10^{-3}$	$4.7 \cdot 10^{-3}$
2 m	$4.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-3}$



(a) Cell size of 10 m



(b) Cell size of 2 m

Figure 5.1: Comparison of exact and computed free surface profile at steady state

The following classification of the flow regime is determined by the Froude number:

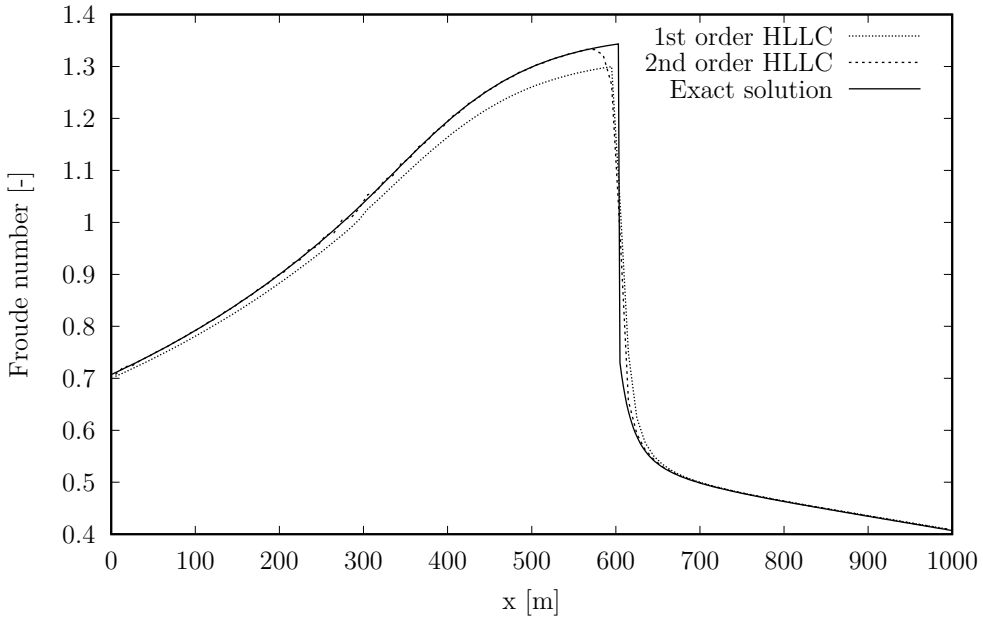
$$\mathrm{Fr} \begin{cases} < 1 & \rightarrow \text{subcritical flow} \\ = 1 & \rightarrow \text{critical flow} \\ > 1 & \rightarrow \text{supercritical flow} \end{cases} \tag{5.3}$$

Figure 5.2 shows the Froude numbers based on the exact and numerical solution for a cell size of 10 m and 2 m. As expected, $\mathrm{Fr} = 1$ at $1/3$ distance, i. e. critical flow conditions. At $2/3$ distance at the hydraulic jump there was a sudden jump in the Froude number.

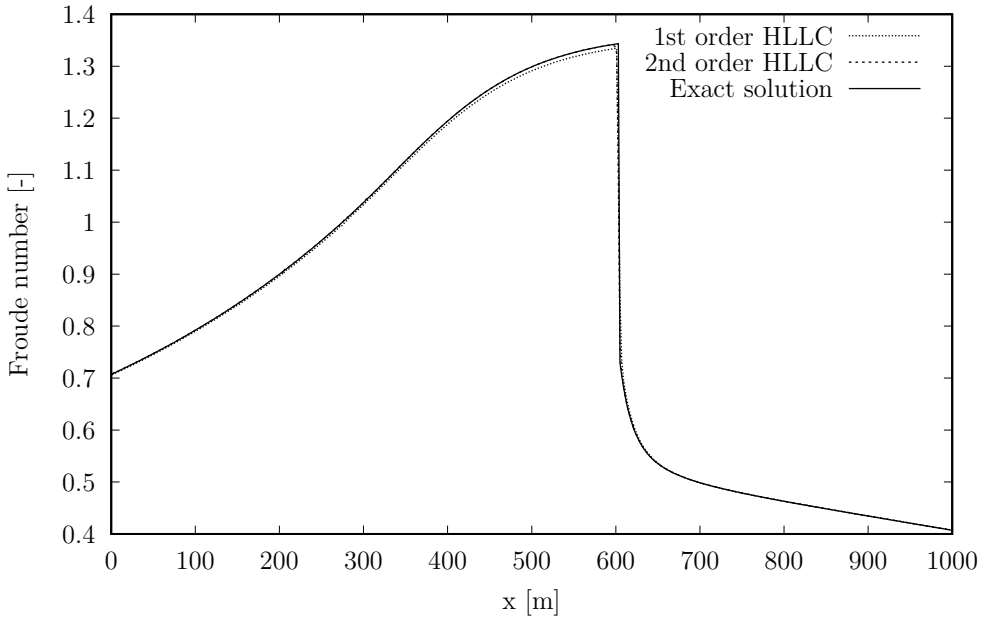
For a cell size of 10 m the first-order scheme showed a noticeable deviation until the hydraulic jump was reached (Figure 5.2a). The flow velocities in this part were too small compared to the exact solution. Very good agreement with the exact solution was observed for both schemes for 2 m cell size (Figure 5.2b).

5.1.2 Discussion

This test case involved the influence of friction, a mildly variation in the bed profile and varying flow conditions. Both, the first- and second-order accurate HLLC scheme can handle the critical flow conditions. On the coarser grid, both schemes showed a smoothing of the sharp discontinuity at the hydraulic jump, which did however not influence the overall result. In summary, the second-order scheme showed better agreement with the exact solution.



(a) Cell size of 10 m



(b) Cell size of 2 m

Figure 5.2: Comparison of Froude numbers for the exact and numerical solution

5.2 1D dam break

To check that the numerical scheme can handle discontinuities in the initial conditions, quasi one-dimensional dam break test cases with initially dry and wet downstream conditions were carried out. The results were compared to the solution of the exact Riemann solver presented in Chapter 3.2.3. The frictionless domain was $20\,\mathrm{m} \times 2\,\mathrm{m}$. The water depth was 4 m upstream of $x = 10\,\mathrm{m}$ and 0 m (dry case) and 1 m (wet case), respectively, downstream. The initial flow velocity is set to $0\,\mathrm{m\,s^{-1}}$. The domain was discretized with a regular grid; the cell size was 0.05 m. All boundaries were closed; the simulation was stopped before the waves reached the downstream and upstream walls. All test cases were carried out using the first-order accurate upwind scheme (Section 3.2.1), and the first- and second-order accurate HLLC scheme (Section 3.2.4). To achieve second-order accuracy in space, the minmod TVD limiter was used (Equation 3.82). The CFL criterion in all cases was set to 0.3 and the time step was computed adaptively using Equation 3.15. In Figures 5.3 and 5.4 the numerical results at $t = 0.3\,\mathrm{s}$ and 0.6 s are qualitatively compared with the analytical solutions.

5.2.1 Dry bed initial conditions

In Figure 5.3a the temporal progress of the water elevation profile in the domain for the dry bed initial conditions is shown. In Table 5.2 the corresponding L^1-errors (Equation 5.1) of the water level are given. All numerical schemes showed an overall good agreement. All schemes could handle the initial discontinuity and dry bed conditions. The first-order upwind schemes shows slightly better results than the first-order HLLC scheme. The most accurate representation of the propagating front was given by the second-order HLLC scheme. This can be seen in Figure 5.3b, which shows the temporal progress of the flow velocity profile. Both the first-order upwind and the first-order HLLC scheme did highly underestimate the flow velocity at the propagating front and therefore showed a delay in the rise of the water. Although the second-order HLLC scheme showed much better agreement with the solution of the exact Riemann solver, there was a deviation at the front, too. This can be explained by the fact that the spatial accuracy of the high-order reconstruction was reduced to first-order at the wet/dry front (cf. Section 3.3.4).

Table 5.2: L^1-errors (m) of water level profile for dry bed initial conditions using the first- oder accurate upwind scheme and the first- and second-order accurate HLLC schemes

time	1st-order upwind	1st-order HLLC	2nd-order HLLC
0.3 s	$1.1 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$5.4 \cdot 10^{-3}$
0.6 s	$1.4 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$5.6 \cdot 10^{-3}$

5.2.2 Wet bed initial conditions

Figure 5.4a shows the temporal progress of the water elevation profile in the wet bed case and in Table 5.3 the corresponding L^1-errors of the water level are given. A shock wave was propagating to the right and a rarefaction wave was propagating to the left side of the domain. In comparison to the previous case, one could immediately recognize the spurious oscillations at the shock front produced by the first-order upwind scheme. The same was true for the temporal progress of the flow velocity profile given in Figure 5.4b. The upwind scheme was not able to capture the shock front without introducing spurious oscillations. However, despite the discontinuity, it stayed stable.

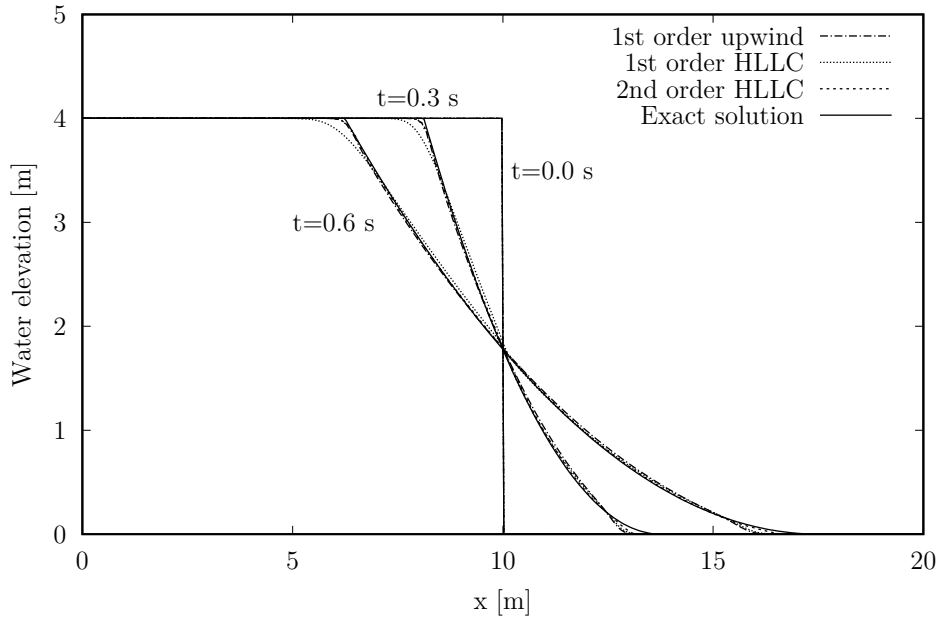
Numerical diffusion was the highest for the first-order HLLC scheme. The best agreement and the best representation of the propagating shock wave was obtained by the second-order HLLC scheme.

Table 5.3: L^1-errors (m) of water level profile for wet bed initial conditions using the first- oder accurate upwind scheme and the first- and second-order accurate HLLC schemes

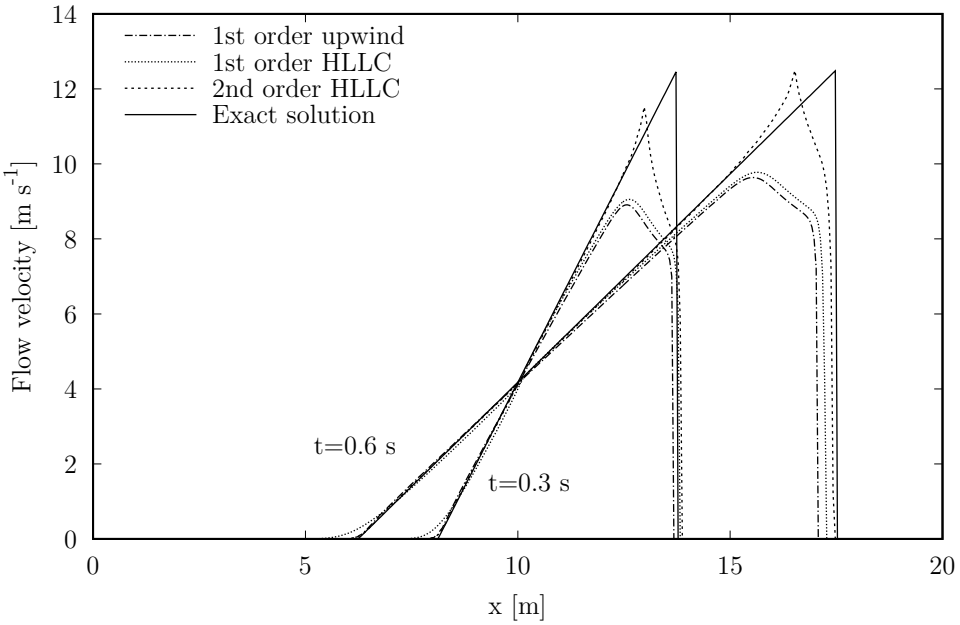
time	1st-order upwind	1st-order HLLC	2nd-order HLLC
0.3 s	$8.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-2}$	$3.8 \cdot 10^{-3}$
0.6 s	$1.1 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$4.0 \cdot 10^{-3}$

5.2.3 Discussion

All schemes gave good or very good agreement with the solution of the exact Riemann solver in the dry bed test case. They handled well the initially dry bed conditions and the discontinuity in the initial water elevation. In the wet bed test case, spurious oscillations at the shock front could be observed for the first-order upwind scheme. The best solution was obtained in both test cases with the second-order accurate HLLC scheme.

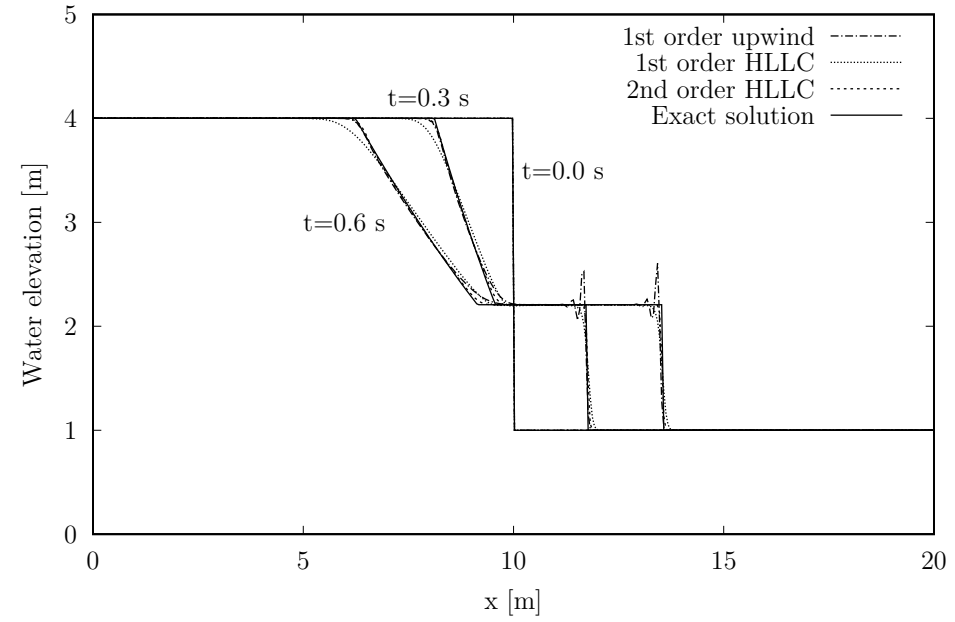


(a) Temporal progress of water elevation profile

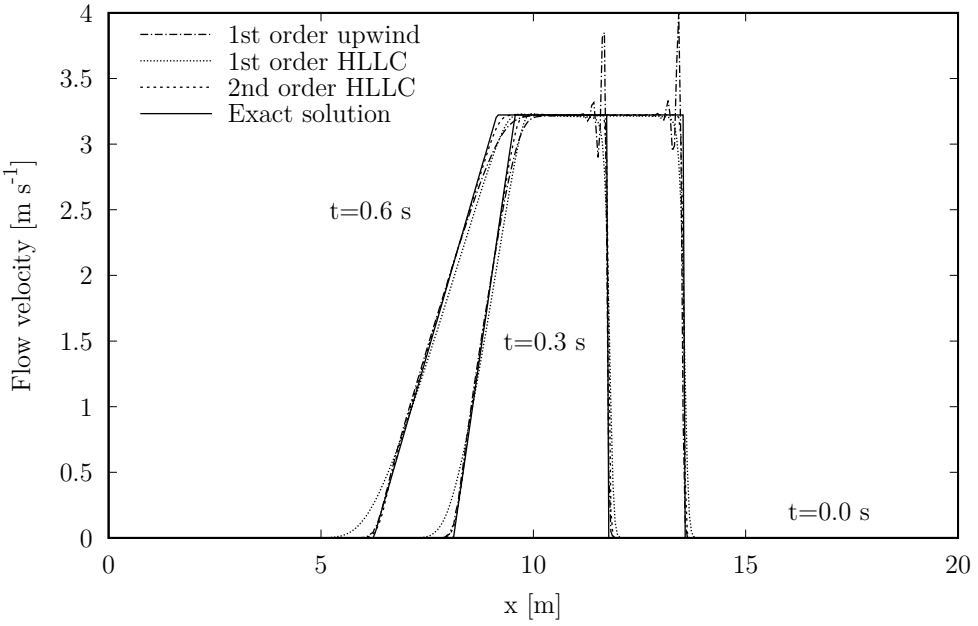


(b) Temporal progress of flow velocity profile

Figure 5.3: Comparison of exact and numerical solutions of first- and second-order schemes for dam break on dry bed test case



(a) Temporal progress of water elevation profile



(b) Temporal progress of flow velocity profile

Figure 5.4: Comparison of exact and numerical solutions of first- and second-order schemes for dam break on wet bed test case

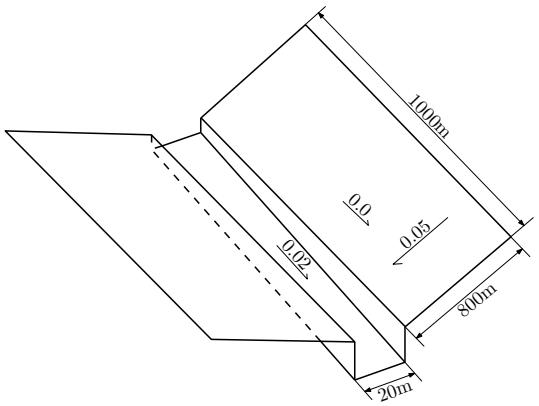


Figure 5.5: Geometry of the V-shaped schematic catchment

5.3 2D rainfall-runoff simulation

This test case aimed to show the accuracy of the presented numerical model in simulating shallow surface runoff and it allowed the comparison of the first- and second-order schemes. Overton and Brakensiek (1970) presented an approximated kinematic solution of the hydrograph for a long steady and uniform rain event for a V-shaped schematic catchment. The geometry is given in Figure 5.5, where all parameters used in this simulation were taken from Di Giammarco et al. (1996). The slope of the catchment was 0.05 and the channel slope was 0.02. The channel had a width of 20 m and the depth varied from 1 m at the upstream end to 20 m at the downstream end. The Manning's roughness coefficient was $0.015\,\mathrm{s\,m^{-1/3}}$ for the hillsides and $0.15\,\mathrm{s\,m^{-1/3}}$ for the channel. The domain was discretized by a uniform rectangular mesh with cell size of 10 m. All boundaries beside the channel flow were closed boundaries, except the outlet, where critical flow depth was assumed. The initial water depth and flow velocity was set to zero. The CFL criterion was set to 0.1 and the time step was computed adaptively using Equation 3.15. A constant rainfall with an intensity of $10.8\,\mathrm{mm\,h^{-1}}$ and a duration of 1.5 h was imposed on the hillsides. The results of the first- and second-order accurate HLLC schemes (Section 3.2.4) were compared. Second-order accuracy was achieved by using the minmod TVD limiter (Equation 3.82).

5.3.1 Results

In Figure 5.6, the numerical solutions of the first- and second-order HLLC schemes are compared to the analytical solution (Overton and Brakensiek, 1970; Stephenson and Meadows, 1986) at the lower end of one hillside and at the outlet of the domain. Hereby, there is no analytical solution for the descending phase of the discharge at the channel outlet. A relative L^1-error was computed from the deviation of the exact and simulated discharge normalized by the averaged exact value:

$$\text{Relative } L^1\text{-error} = \frac{\sum_{i=1}^{N} \left(A_i \cdot |\mathbf{q}_{i,\text{exact}} - \mathbf{q}_{i,\text{simulated}}| \right)}{\sum_{i=1}^{N} \left(A_i \cdot \mathbf{q}_{i,\text{exact}} \right)} \cdot 100 \ \ (\%) \tag{5.4}$$

where N is the total number of cells.

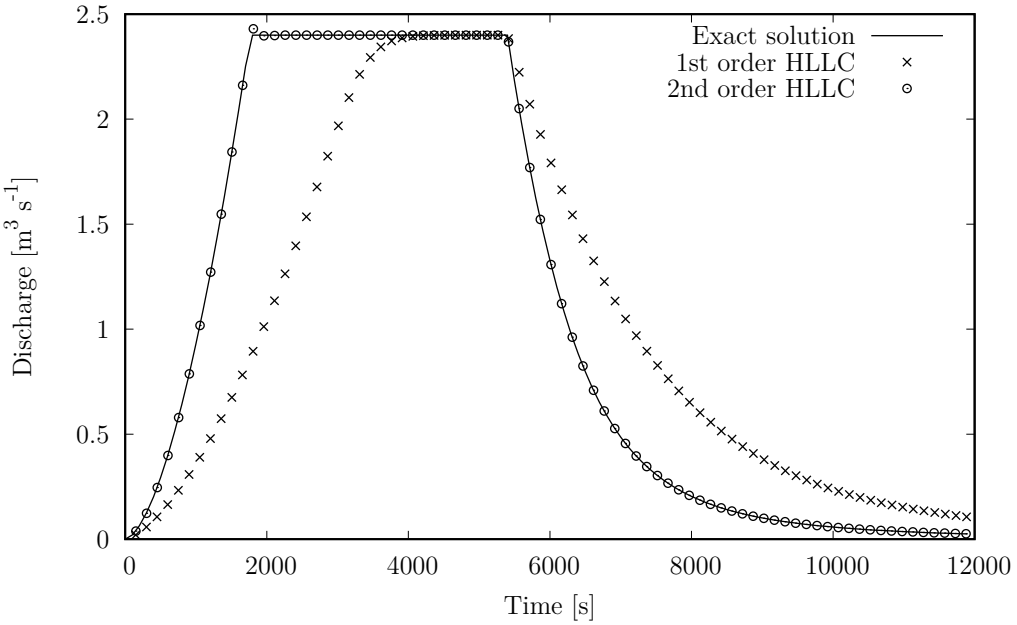
The relative errors according to Equation 5.4 are given in Table 5.4. The results showed a very good agreement of the second-order scheme with the analytical solution. For the first-order scheme the results for the ascending and descending phase were poor and strong detention was recognized. However, by comparing the area below the hydrographs, it was recognized that mass balance was also guaranteed for the first-order scheme.

5.3.2 Discussion

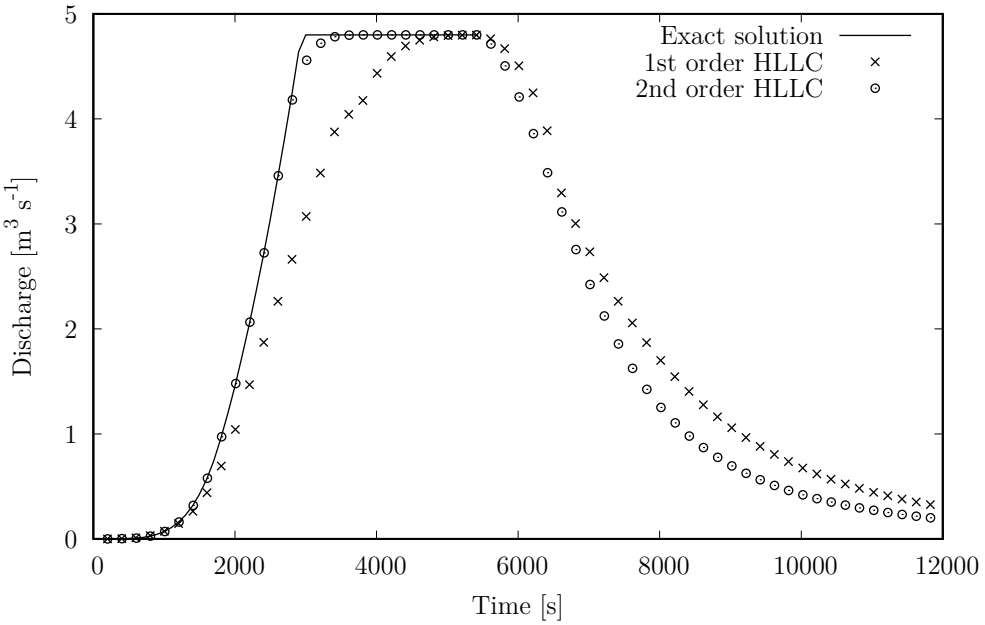
This test case involved shallow surface runoff due to rainfall in a sloping catchment with friction. The results showed an artificial detention of the ascending and descending hydrographs and poor agreement with the exact solution for the first-order accurate HLLC scheme. This is due to the hydrostatic reconstruction, which in the first-order accurate scheme reduces the continuous sheetlike flow to piecewise flux computations between wet and artificially dry cells. As a consequence, a wrong flow field and hydrograph is computed (compare Section 3.2.6). The second-order

Table 5.4: Relative L^1-errors (%) for first- and second-order scheme

	1st-order	2nd-order
Hydrograph at one hillside	34.5	0.1
Hydrograph at channel outlet	14.5	0.7



(a) Hydrograph at one hillside



(b) Hydrograph at channel outlet

Figure 5.6: Comparison of analytical and numerical solution of the first- and second-order scheme

accurate scheme discretized the sloping domain as a continuous flow problem and so it is possible to compute flow of thin water on course grids with steep bottom gradients. A very good agreement of numerical and exact solution was recognized.

5.4 2D parabolic bowl

The analytical solution for oscillatory flow in a 2D parabolic bowl has been originally presented by Thacker (1981). It was extended to include friction in the 1D case by Sampson et al. (2006). Wang et al. (2011) presented the analytical solution for the frictional 2D parabolic bowl.

The bed topography of the bowl is given by:

$$z_{\mathrm{B}}\left(x,y\right) = h_0\left(x^2+y^2\right)/a^2 \tag{5.5}$$

with the two constants h_0 and a. The origin is assumed at the center of the bowl. The solutions for the water elevation and the flow velocities are given as:

$$\begin{aligned} h\left(x,y,t\right) &= h_0 - \frac{1}{2g}B^2e^{-\tau t} - \frac{1}{g}Be^{-\tau t/2}\left(\frac{\tau}{2}\sin st + s\cos st\right)x \\ &\quad -\frac{1}{g}Be^{-\tau t/2}\left(\frac{\tau}{2}\cos st - s\sin st\right)y \end{aligned} \tag{5.6}$$

$$u\left(t\right) = Be^{-\tau t/2}\sin st \tag{5.7}$$

$$v\left(t\right) = -Be^{-\tau t/2}\cos st \tag{5.8}$$

with the constant B, the bed friction parameter τ and:

$$s = \sqrt{p^2-\tau^2}/2 \tag{5.9}$$

where the peak amplitude parameter is expressed as:

$$p = \sqrt{8gh_0/a^2} \tag{5.10}$$

The computational domain was $5\,000\,\mathrm{m} \times 5\,000\,\mathrm{m}$. The initial water elevation and flow velocities were given by Equations 5.6, 5.7 and 5.8 for $t = 0\,\mathrm{s}$. The oscillatory flow was parameterized with $B = 5\,\mathrm{m\,s^{-1}}$, $h_0 = 10\,\mathrm{m}$ and $a = 3\,000\,\mathrm{m}$. The case was carried out frictionless and with friction using $\tau = 0.002\,\mathrm{s^{-1}}$, where the Chézy

coefficient was computed as:

$$C = \sqrt{\frac{g|\mathbf{v}|}{d \cdot \tau}} \tag{5.11}$$

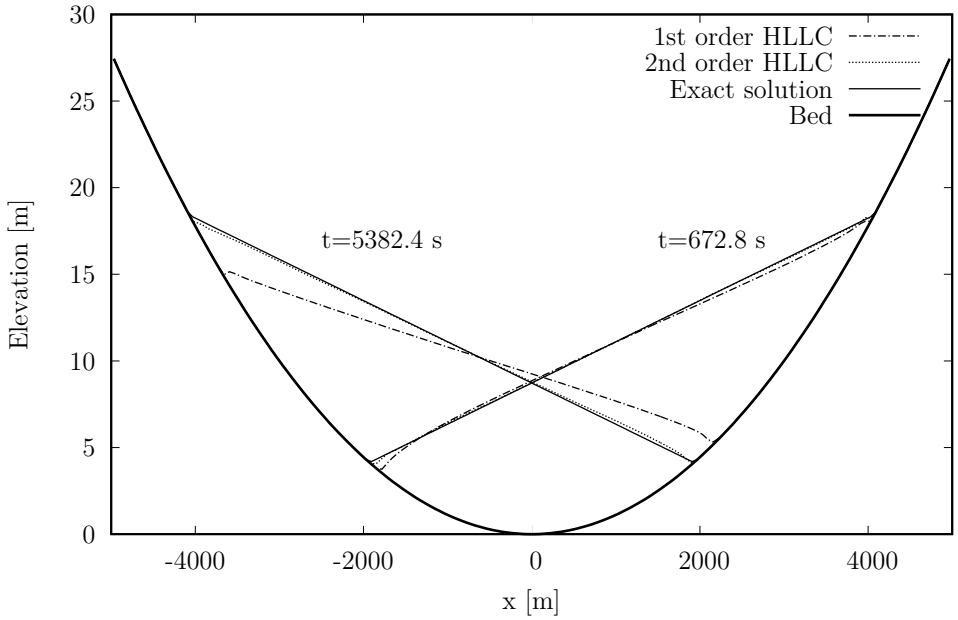
The domain was discretized with a regular grid; the cell size is 62.5 m (160 $\times$ 160 cells). All boundaries were closed. The CFL criterion was set to 0.25 and the time step was computed adaptively using Equation 3.15. The results of the first- and second-order accurate HLLC schemes (Section 3.2.4) were compared. Second-order accuracy was achieved by using the monotonized central-difference (MC) TVD limiter (Equation 3.83) and the improved Euler time stepping method (Section 3.1.2). The limiting water depth for dry cells was set to $1 \cdot 10^{-3}$ m to guarantee stable results (Section 3.3.4).

5.4.1 Results

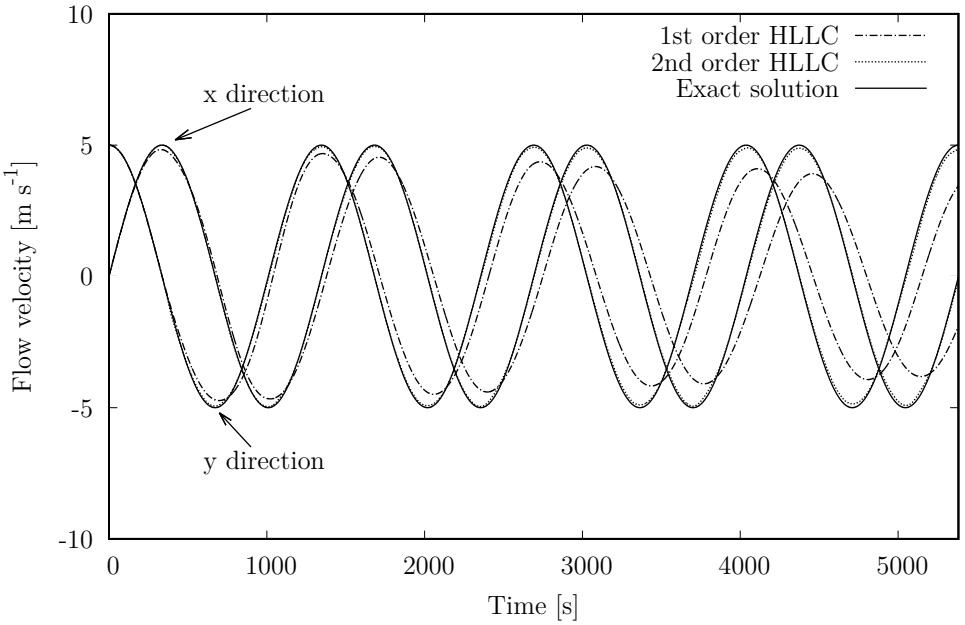
In Figure 5.7 and 5.8 the exact and the numerical solutions of the first- and second-order accurate HLLC schemes for the frictionless case and the case with friction are compared. The second-order accurate HLLC scheme showed very good qualitative agreement in both cases. Figure 5.7a and 5.8a show the water elevation in x direction ($y = 0\,\mathrm{m}$) after half a period and after four periods. In the frictionless case a distinct deviation of exact solution and the numerical solution of the first-order solution was observed after four periods. With friction, the deviation was much smaller for the first-order scheme. The damping due to bed friction was much higher than the influence of the numerical diffusion. The same was the case for the temporal progress of the flow velocity. Figure 5.7b and 5.8b show the flow velocity in x and y direction at the point (0 m,1 000 m). Again, the first-order accurate HLLC scheme had a strong retardation in the frictionless case.

5.4.2 Grid convergency

Next, the grid convergency were studied by varying the cell size between 250 m (20 $\times$ 20 cells) to 7.8125 m (640 $\times$ 640 cells). In this test case, the root mean square (L^2) error was computed for each simulation from the deviation of exact and simulated

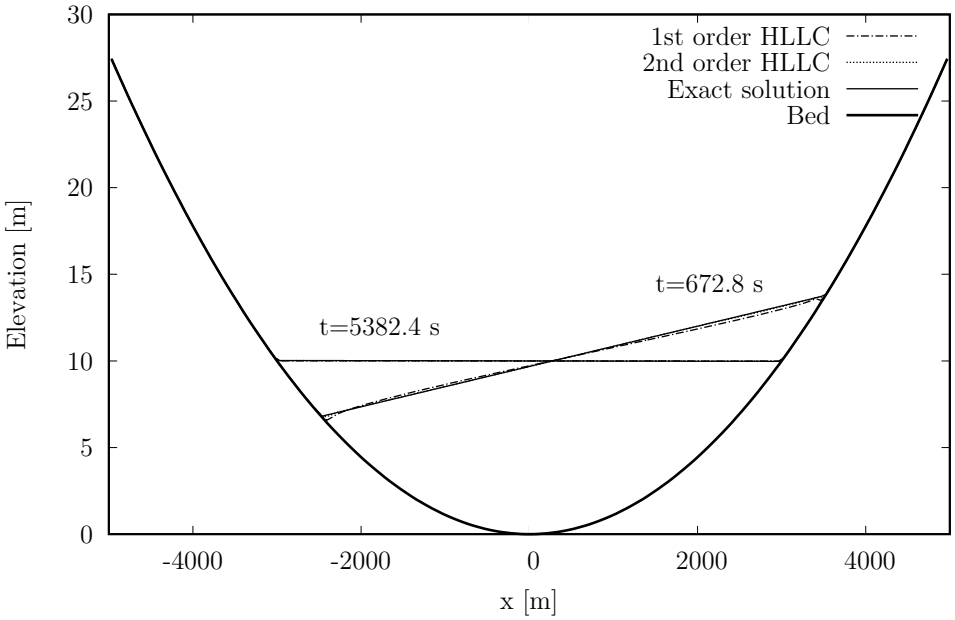


(a) Water elevation after half a period ($t = 672.8\,\mathrm{s}$) and four periods ($t = 5382.4\,\mathrm{s}$) for $y = 0\,\mathrm{m}$

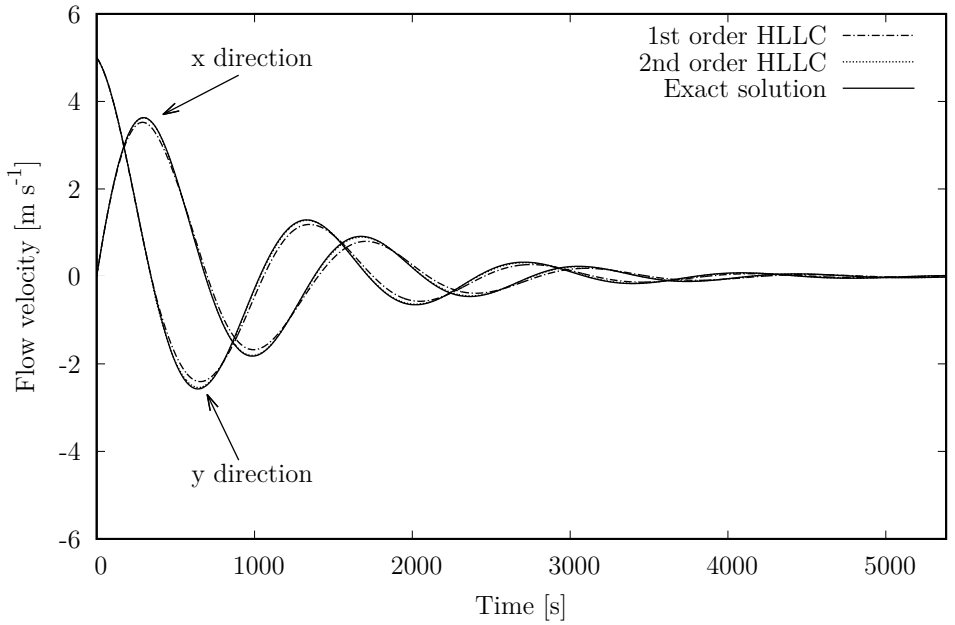


(b) Temporal progress of flow velocity in x and y direction over four periods at (0 m,1000 m)

Figure 5.7: Comparison of exact and numerical solutions of first- and second-order schemes for frictionless oscillatory flow in parabolic bowl



(a) Water elevation after half a period ($t = 672.8\,\mathrm{s}$) and four periods ($t = 5382.4\,\mathrm{s}$) for $y = 0\,\mathrm{m}$



(b) Temporal progress of flow velocity in x and y direction over four periods at (0 m,1000 m)

Figure 5.8: Comparison of exact and numerical solutions of first- and second-order schemes for oscillatory flow in parabolic bowl with friction

values of all cells N:

$$L^2 = \sqrt{\frac{\sum_{i=1}^{N} \left(A_i \cdot |\mathbf{q}_{i,\text{exact}} - \mathbf{q}_{i,\text{simulated}}|^2 \right)}{\sum_{i=1}^{N} A_i}} \tag{5.12}$$

The order of accuracy is then computed by the binary logarithm of the ratio of the L^2-errors of the coarse and the refined grid:

$$\text{order} = \text{lb} \left(\frac{L^2_{\Delta x}(h)}{L^2_{\Delta x/2}(h)} \right) \tag{5.13}$$

Table 5.5 and 5.6 show the L^2-errors based on the water elevation of the first- and second-order accurate HLLC scheme depending on the cell count. In addition the order of accuracy computed by Equation 5.13 is shown. As expected from the results presented previously, the L^2-errors of the case with friction (Table 5.6) are always below the ones of the frictionless case (Table 5.5).

The highest order of accuracy for the second-order accurate scheme was 1.5 and for the first-order scheme it was 1.0. This proves the improved accuracy of the second-order scheme. At the same time, it can be seen, that the theoretical value of 2.0 can not be reached due to the reduction of the order at the wet/dry fronts.

5.4.3 Discussion

In the presented test case the implementation of the friction and the bottom slope source term were tested. In addition an ongoing wetting and drying of the domain took place. The results prove the correct and accurate handling of friction, varying bed elevation and the wetting/drying. The simulation was stable over the whole time. The second-order accurate scheme showed its superiority over the first-order scheme in terms of accuracy.

Table 5.5: L^2-errors (m) based on the water elevation and order of accuracy for frictionless case after half a period ($t = 672.8\,\mathrm{s}$)

	2nd-order		1st-order	
cell count	L^2-error	order	L^2-error	order
20×20	$4.1 \cdot 10^{-1}$	-	$7.9 \cdot 10^{-1}$	-
40×40	$1.8 \cdot 10^{-1}$	1.2	$4.7 \cdot 10^{-1}$	0.7
80×80	$7.8 \cdot 10^{-2}$	1.2	$2.7 \cdot 10^{-1}$	0.8
160×160	$2.9 \cdot 10^{-2}$	1.4	$1.5 \cdot 10^{-1}$	0.8
320×320	$1.0 \cdot 10^{-2}$	1.5	$8.1 \cdot 10^{-2}$	0.9
640×640	$4.1 \cdot 10^{-3}$	1.3	$4.1 \cdot 10^{-2}$	1.0

Table 5.6: L^2-errors (m) based on the water elevation and order of accuracy for case with friction after half a period ($t = 672.8\,\mathrm{s}$)

	2nd-order		1st-order	
cell count	L^2-error	order	L^2-error	order
20×20	$2.7 \cdot 10^{-1}$	-	$4.1 \cdot 10^{-1}$	-
40×40	$1.2 \cdot 10^{-1}$	1.1	$2.5 \cdot 10^{-1}$	0.7
80×80	$4.9 \cdot 10^{-2}$	1.3	$1.4 \cdot 10^{-1}$	0.9
160×160	$1.8 \cdot 10^{-2}$	1.4	$7.3 \cdot 10^{-2}$	0.9
320×320	$6.7 \cdot 10^{-3}$	1.5	$3.8 \cdot 10^{-2}$	0.9
640×640	$2.6 \cdot 10^{-3}$	1.3	$1.9 \cdot 10^{-2}$	1.0

5.5 1D advective-diffusive transport

The following test cases prove the validity and accuracy of the implemented transport schemes. The one-dimensional advective-diffusive transport of a constant instantaneous injection was simulated.

The frictionless domain was 20 m × 2 m. The water depth was 1 m. There was an instantaneous injection with a constant concentration of 1.0 at 9.95 m $< x <$ 10.05 m. To clearly demonstrate the effect of diffusion, the diffusion coefficient D was set to a high constant value of 0.01 $\mathrm{m^2\,s^{-1}}$. The domain was discretized with a regular grid; the cell size was 0.05 m. All boundaries were closed. For the simulation a fixed time step of 0.001 s was used.

The analytical solution to these problems is given by:

$$c\left(x,t\right) = \frac{c_0}{2}\left[\mathrm{erf}\left(\frac{x + {}^{\Delta w}/_{2} - u \cdot t}{\sqrt{4Dt}}\right) - \mathrm{erf}\left(\frac{x - {}^{\Delta w}/_{2} - u \cdot t}{\sqrt{4Dt}}\right)\right] \tag{5.14}$$

with the error function:

$$\mathrm{erf}\left(x\right) = \frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-\xi^2}\,\mathrm{d}\xi \tag{5.15}$$

In here, the initial tracer concentration is denoted as c_0 and Δw is the initial width of the polluted area. In the solution, the center of the polluted area is assumed at $x = 0$ m, so, here it was shifted to the right by 10 m.

5.5.1 Diffusive transport

First, only diffusive transport was simulated and the flow velocity was set to 0 $\mathrm{m\,s^{-1}}$. Figure 5.9 shows the development of the concentration profile over time in comparison to the analytical solution obtained by Equation 5.14. The initially constant injection was spread rapidly. Very good qualitative agreement of the numerical and the analytical solution was observed. In Table 5.7 the L^1-errors according to Equation 5.1 are given.

5.5.2 Advective-diffusive transport

In the next test case, a constant flow velocity 0.1 $\mathrm{m\,s^{-1}}$ in positive x direction was assumed. Here, the first- and second-order accurate HLLC schemes (Section 3.2.4)

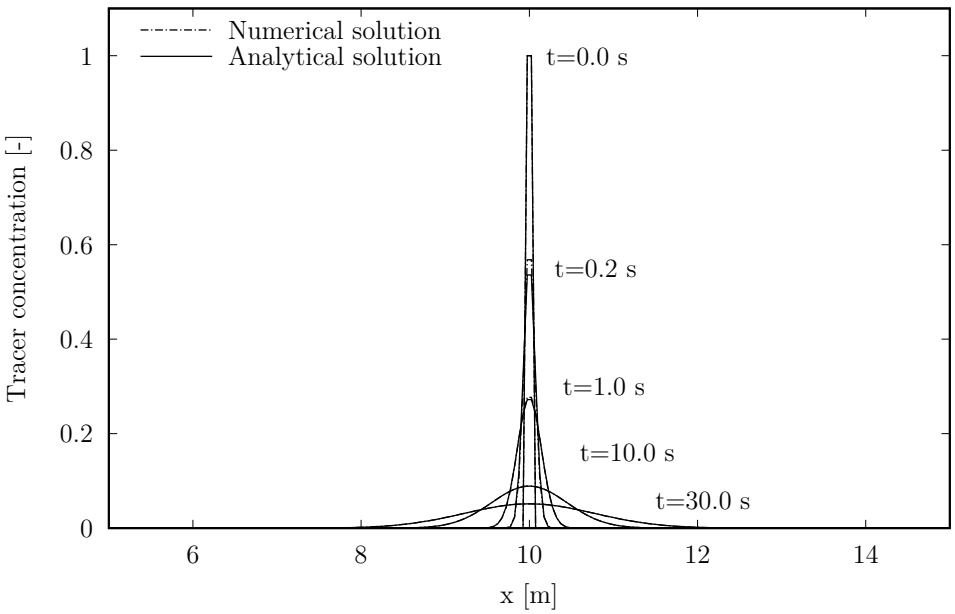


Figure 5.9: Temporal development of the concentration profile for sole diffusive transport of an instantaneous injection

Table 5.7: L^1-errors (-) of concentration profile for sole diffusive transport

time	0.2 s	1.0 s	10.0 s	30.0 s
L^1-error	$4.3 \cdot 10^{-4}$	$7.5 \cdot 10^{-5}$	$7.8 \cdot 10^{-6}$	$2.6 \cdot 10^{-6}$

were compared to the analytical solution. Second-order accuracy was achieved by using the minmod TVD limiter (Equation 3.82).

In Figure 5.10 the numerical solution and the analytical solution are compared at different times. In Table 5.8 the L^1-errors are given. The comparison showed perfect qualitative agreement for the second-order accurate HLLC scheme. The first-order accurate scheme showed good agreement, too, however a stronger spreading of the injection cloud was observed. This can be explained by artificial numerical diffusion which adds to the diffusion process.

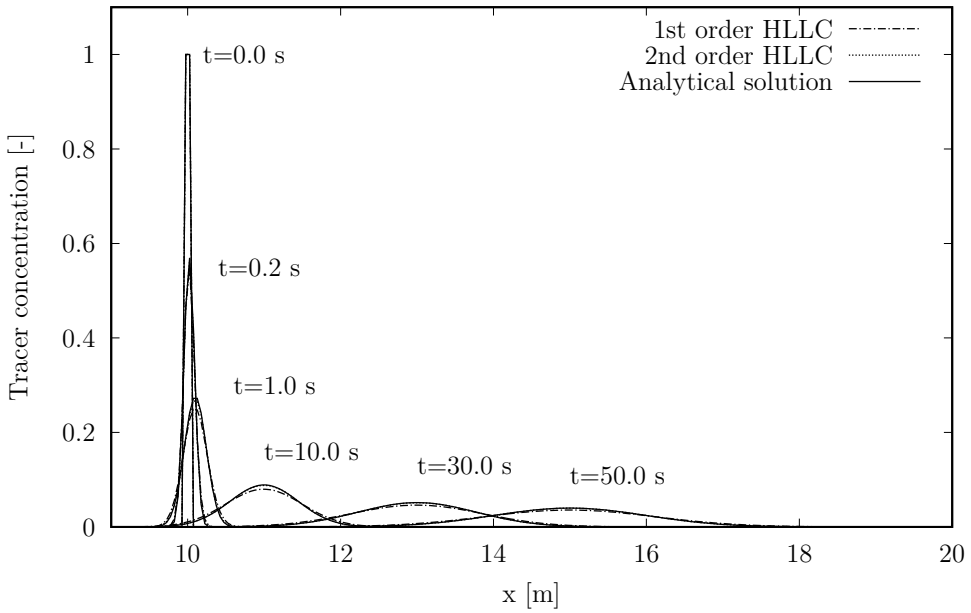


Figure 5.10: Temporal development of the concentration profile for advective-diffusive transport of an instantaneous injection

Table 5.8: L^1-errors (-) of concentration profile for advective-diffusive transport using the first- and second-order accurate HLLC schemes

time	2nd-order	1st-order
0.2 s	$1.6 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$
1.0 s	$1.1 \cdot 10^{-4}$	$4.7 \cdot 10^{-4}$
10.0 s	$5.3 \cdot 10^{-5}$	$5.3 \cdot 10^{-4}$
30.0 s	$3.2 \cdot 10^{-5}$	$5.3 \cdot 10^{-4}$
50.0 s	$2.5 \cdot 10^{-5}$	$5.3 \cdot 10^{-4}$

5.5.3 Discussion

Pure diffusive transport was simulated well by the implemented scheme. For advective-diffusive transport the second-order accurate HLLC scheme showed very good agreement with the analytical solution. The first-order accurate HLLC scheme introduced slight numerical diffusion which added to the diffusion process. In summary, the correctness and accuracy of the implemented schemes could by verified.

5.6 Sediment transport and morphological evolution

Here, the implementation of the mathematical sediment transport and morphological evolution model described in Section 2.5 is given as a proof of concept. The accuracy of the implemented scheme is shown by simulating a one-dimensional dam break over mobile bed. As reference solution, the numerical results of Cao et al. (2004) were used.

5.6.1 Implementation in `hms`

Implementing a new layer for finite volume computation starts by writing down meta data which describes the new layer. These are the layer name, names and units of all considered state variables and the total variable count. The meta data is used in `hms` to reference state variables stored at mesh cell centers and to provide information to the runtime visualization and for data exchange with other software. All meta data is defined in a class which has the same name as the described layer and which is located in a specific layer package. In Listing 5.1, a truncated version of the `Morphology` meta data definition class is given which is provided in the `de.tuberlin.wahyd.hms.layer.morphology` package. The full source code of the implemented code is available in the GIT repository.

Listing 5.1: Truncated meta data definition for `Morphology` layer

```
1  package de.tuberlin.wahyd.hms.layer.morphology;
2
3  public class Morphology {
4
5    // Layer-Name
6    public final static String LAYER_SIMPLE_NAME =
7      Morphology.class.getSimpleName();
8    public final static String LAYER_FULL_NAME =
9      "HMS:" + LAYER_SIMPLE_NAME;
10   public final static int LAYER_INDEX =
11     HMS.indexLayer(LAYER_FULL_NAME);
12
13   // Sediment concentration
14   public final static String SIMPLE_NAME_SEDIMENT_CONCENTRATION =
15     "Sediment␣concentration";
```

```
16    public final static String FULL_NAME_SEDIMENT_CONCENTRATION =
17      LAYER_FULL_NAME + ":" + SIMPLE_NAME_SEDIMENT_CONCENTRATION;
18    public final static int INDEX_SEDIMENT_CONCENTRATION = 0;
19    public final static int DIMENSION_SEDIMENT_CONCENTRATION = 1;
20    public final static VariableAnnex UNIT_SEDIMENT_CONCENTRATION =
21      HMS.UNIT_DIMENSIONLESS;
22    public final static VariableProperty
23      PROPERTY_SEDIMENT_CONCENTRATION = VariableProperty.INTENSIVE;
24
25    // more state variables...
26
27    // Variables-Count
28    public final static int VARIABLES_COUNT = 8;
29
30    // Meta-Data
31    public final static CalcLayerMetaData META_DATA =
32      new HCalcLayerMetaData();
33
34    static {
35      final HCalcLayerMetaData meta = (HCalcLayerMetaData) META_DATA;
36      {
37        final HVariable variable = new HVariable();
38        variable.setFullName(FULL_NAME_SEDIMENT_CONCENTRATION);
39        variable.setSimpleName(SIMPLE_NAME_SEDIMENT_CONCENTRATION);
40        variable.setDimension(DIMENSION_SEDIMENT_CONCENTRATION);
41        variable.setAnnex(UNIT_SEDIMENT_CONCENTRATION);
42        variable.setProperty(PROPERTY_SEDIMENT_CONCENTRATION);
43        meta.addVariable(variable);
44      }
45      // adding more state variables...
46    }
47  }
```

In addition to the meta data definition an implementation of the `AttributeVariables` interface (Section 4.1.3) which is specific for the new layer is necessary. The `HMorphologyValues` class is implementing the `AttributeVariables` interface and stores all variables defined in the meta data. Each variable is identified by the `INDEX_` constant

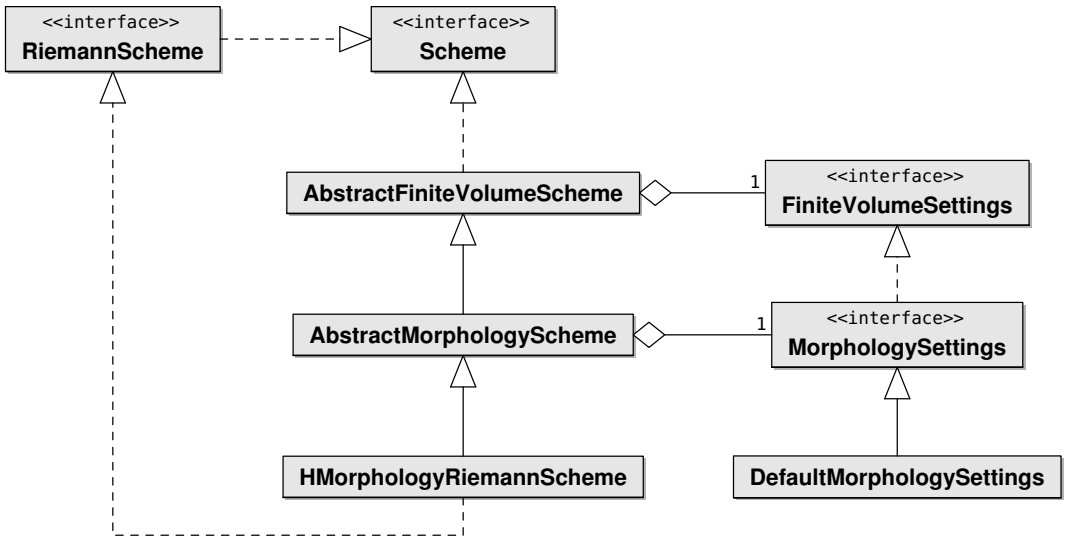


Figure 5.11: UML representation of the morphology scheme

(cf. Listing 5.1, line 18). In the finite volume framework each cell will hold one instance of the `HMorphologyValues` class to store the state variables.

Next, an implementation of the `Scheme` interface has to be given, i. e. a discretized solution of the flux and source terms and the storage variables (cf. 4.1). The `HMorphologyRiemannScheme` was implemented as an extension of the `AbstractMorphologyScheme` and as a realization of the `RiemannScheme` interface. The `AbstractMorphologyScheme`, which is an extension the `AbstractFiniteVolumeScheme`, was introduced as another layer of abstraction to allow different implementations of the flux calculation and of the computations of deposition and erosion. All necessary settings were defined in the `MorphologySettings` interface and a default implementation was given. An UML representation of the dependencies is given in Figure 5.11.

As an example, the implementation of the `computeSource` method will be explained in detail. The method was chosen as it allows to show the usage of context objects and settings, the interaction with other schemes and the abstraction layers.

The `computeSource` method gives the discretized solution of the source terms in Equation 2.36 for the current time step and cell. The current cell can be retrieved from the given cell context object (cf. line 5 in Listing 5.2). In line 7, the `AttributeVariables` object for the morphology layer is retrieved from the current cell. In the following all state variables can be accessed using this object (e.g. line 10). The

current source term used in the finite volume solution is also retrieved from the cell context object (line 31). In the lines 33–35 the actual source term entries are computed (Equation 2.36). In the following lines 37–40, the source term components which belong to the shallow water equations are extended by the mass exchange between flow and erodible bed (Equation 2.37) and the momentum transfer due to sediment exchange (Equations 2.38 and 2.39).

Listing 5.2: Implementation of `computeSource` method in the `AbstractMorphologyScheme`

```
1  @Override
2  public void computeSource(final TimeStamp t, final double dt,
3    final FvmCellContext cellContext) {
4
5    final MeshCell cell = cellContext.getCell();
6    final AttributeVariables vals =
7      cell.getValues(Morphology.LAYER_INDEX);
8    final double entrainment = computeEntrainment(cell, t, dt);
9    final double[] entrainmentCellValue =
10     vals.getValue(Morphology.INDEX_SUBSTRATE_ENTRAINMENT);
11   entrainmentCellValue[0] = entrainment;
12
13   double deposition = 0.0;
14   if (settings.isDepositionEnabled()) {
15     deposition = computeDeposition(cell, t, dt);
16     final double[] depositionCellValue =
17     vals.getValue(Morphology.INDEX_SUBSTRATE_DEPOSITION);
18     depositionCellValue[0] = deposition;
19   }
20
21   final double[] exchangeRateCellValue =
22     vals.getValue(Morphology.INDEX_EXCHANGE_RATE);
23   exchangeRateCellValue[0] = entrainment - deposition;
24
25   final double bedSedimentPorosity =
26     settings.getBedSedimentPorosity(cell, t);
27
28   final double[] momentumSource =
```

```
29      computeMomentumSource(cell, t, dt);
30
31    final double[] source = cellContext.getSource();
32
33    source[startIndex + 0] += deposition - entrainment;
34    source[startIndex + 1] += (entrainment - deposition)
35      / (1.0 - bedSedimentPorosity);
36
37    source[SURFACE_FLOW_INDEX] += (deposition - entrainment)
38      / (1.0 - bedSedimentPorosity);
39    source[SURFACE_FLOW_INDEX + 1] += momentumSource[0];
40    source[SURFACE_FLOW_INDEX + 2] += momentumSource[1];
41  }
42
43  protected abstract double[] computeMomentumSource(MeshCell cell,
44    TimeStamp t, double dt);
45
46  protected abstract double computeEntrainment(MeshCell cell,
47    TimeStamp t, double dt);
48
49  protected abstract double computeDeposition(MeshCell cell,
50    TimeStamp t, double dt);
```

The computation of the entrainment (line 8), deposition (line 15) and momentum source (line 29) is not implemented directly in the `AbstractMorphologyScheme`, but declared as abstract methods (lines 43–50). This allows to implement different schemes which use different approaches to compute these values. In Listing 5.3 the implementation of the `computeEntrainment` method in the `HMorphologyRiemann-Scheme` is shown. The method implements the computation of the entrainment for non-cohesive material (Equation 2.42) and for cohesive material (Equation 2.45). In the settings it can be decided, which entrainment equation will be used (line 25 in Listing 5.3).

Listing 5.3: Implementation of `computeEntrainment` method in the `HMorphologyRiemannScheme`

```
1  @Override
2  protected double computeEntrainment(final MeshCell cell,
3    final TimeStamp t, final double dt) {
4
5    double entrainment = 0.0;
6
7    final double vNorm = LinearAlgebras.getNorm(cell.getValues(
8    Morphology.LAYER_INDEX).getValue(Morphology.INDEX_FLOW_VELOCITY));
9    final double h = cell.getValues(Morphology.LAYER_INDEX).getValue(
10     Morphology.INDEX_WATER_DEPTH)[0];
11   final double z = cell.getValues(Morphology.LAYER_INDEX).getValue(
12     Morphology.INDEX_BOTTOM_ELEVATION)[0];
13
14   double erodibleLayerHeight = Double.MAX_VALUE;
15
16   if (settings.isNonErodibleBedEnabled()) {
17     erodibleLayerHeight =
18       z - settings.getNonErodibleBedElevation(cell, t);
19   }
20
21   if (erodibleLayerHeight < 1.0e-6 || h < waterDepthDry) {
22     return 0.0;
23   }
24
25   if (settings.isCohesive()) {
26     // COHESIVE LAW (Izumi & Parker, 2000)
27     final double betha = settings.getEntrainmentCoefficientBetha();
28     final double gamma = settings.getEntrainmentExponentGamma();
29     final double uC = settings.getThresholdEntrainmentVelocity();
30
31     if (vNorm <= uC) {
32       return 0.0;
33     } else {
34       entrainment = betha * Math.pow(vNorm / uC - 1., gamma);
35
```

```
36        if (entrainment * dt > erodibleLayerHeight) {
37          entrainment = erodibleLayerHeight / dt;
38        }
39        return entrainment;
40      }
41    } else {
42      // NONCOHESIVE LAW (Cao et al., 2004)
43      final double s = (settings.getSedimentDensity(cell, t) /
44        settings.getFluidDensity(cell, t)) - 1.0;
45      final double d = settings.getGrainDiameter(cell, t);
46      final double g = settings.getGravityConstant();
47
48      if (h <= settings.getWaterDepthDry()) {
49        return 0.0;
50      } else {
51        final double theta = Math.pow(vNorm, 2/(Math.pow(
52          settings.getChezyCoefficient(cell,t), 2) * s * d);
53        final double thetaC = settings.getThresholdShieldsParameter();
54        if (theta <= thetaC) {
55          return 0.0;
56        }
57
58        final double nu = settings.getKinematicViscosityOfWater();
59        final double r = d * Math.sqrt(s * g * d) / nu;
60        final double uInf = 7.0 * vNorm / 6.0;
61
62        entrainment = (160.0 / Math.pow(r, 0.8)) * ((1.0
63          - settings.getBedSedimentPorosity(cell, t)) / (thetaC))
64          * ((d * (theta - thetaC) * uInf) / h);
65
66        if (entrainment * dt > erodibleLayerHeight) {
67          entrainment = erodibleLayerHeight / dt;
68        }
69        return entrainment;
70      }
71    }
72  }
```

Based on the `BcValues` interface (Section 4.3.5), several simple boundary conditions were implemented. Using the implemented `Morphology` layer definition, the `HMorphologyValues`, the `HMorphologyRiemannScheme`, the `MorphologySettings`, and the boundary conditions a new application can be setup according to Section 4.4.2. In the following a test case shows the correctness and accuracy of the implemented scheme.

5.6.2 1D dam break over mobile bed

The considered domain was $50\,000\,\mathrm{m} \times 200\,\mathrm{m}$ and the dam break position was at $x = 25\,000\,\mathrm{m}$. The upstream water depth was 40 m and downstream was is 2 m. The initial flow velocity was set to $0\,\mathrm{m\,s^{-1}}$. The Manning's roughness coefficient was set to $0.03\,\mathrm{s\,m^{-1/3}}$. A non-cohesive material (Equation 2.42) with a bed sediment porosity of $\phi = 0.4$, a grain diameter of $d_\mathrm{G} = 8 \cdot 10^{-3}\,\mathrm{m}$, a sediment density of $\rho_\mathrm{s} = 2650\,\mathrm{kg\,m^{-3}}$ and a water density of $\rho_\mathrm{w} = 1000\,\mathrm{kg\,m^{-3}}$ were assumed. The critical Shields parameter θ_c was set to 0.045. The kinematic viscosity of water was $\nu = 1.2 \cdot 10^{-6}\,\mathrm{m^2\,s^{-1}}$. The domain was discretized using a regular grid with a cell size of 10 m. The CFL criterion was set to 0.3 and the time step was computed adaptively using Equation 3.15. The same time step was used for the shallow water equations and the sediment transport equations. For the numerical solution of the governing equations, the same schemes were applied for all equations. The results of the first- and second-order accurate HLLC schemes (Section 3.2.4) were compared. Second-order accuracy was achieved by using the minmod TVD limiter (Equation 3.82). The bottom elevation was updated using the bed-load mass balance equation (Equation 2.36).

5.6.3 Results

In Figure 5.12 the free surface and bed profiles at $t = 60\,\mathrm{s}$ are given. The results of the first- and second-order accurate HLLC schemes were compared with the numerical results presented by Cao et al. (2004). Both schemes showed good qualitative agreement of all simulation results with the results given by Cao et al. (2004). The hydraulic jump in the free surface profile near the origin of the dam break was captured well by both schemes. In comparison to the one-dimensional dam break on wet bed (Section 5.2.2), it was seen, that the mobile bed had a distinct effect on the shape of the free surface profile. Cao et al. gave a detailed comparison of the dam break on fixed and mobile bed (2004).

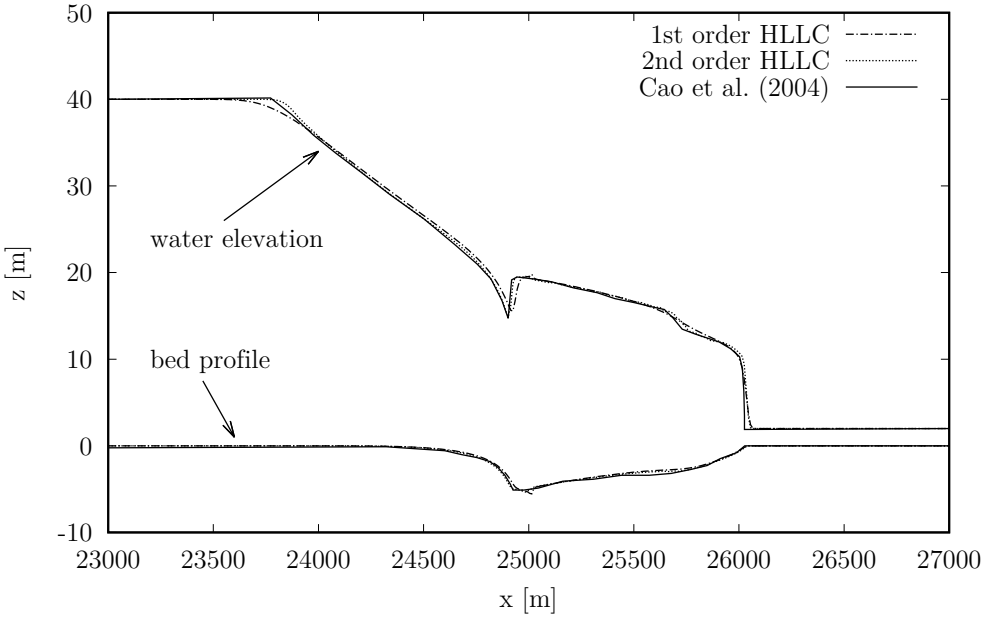


Figure 5.12: Free surface and bed profiles at $t = 60\,\mathrm{s}$

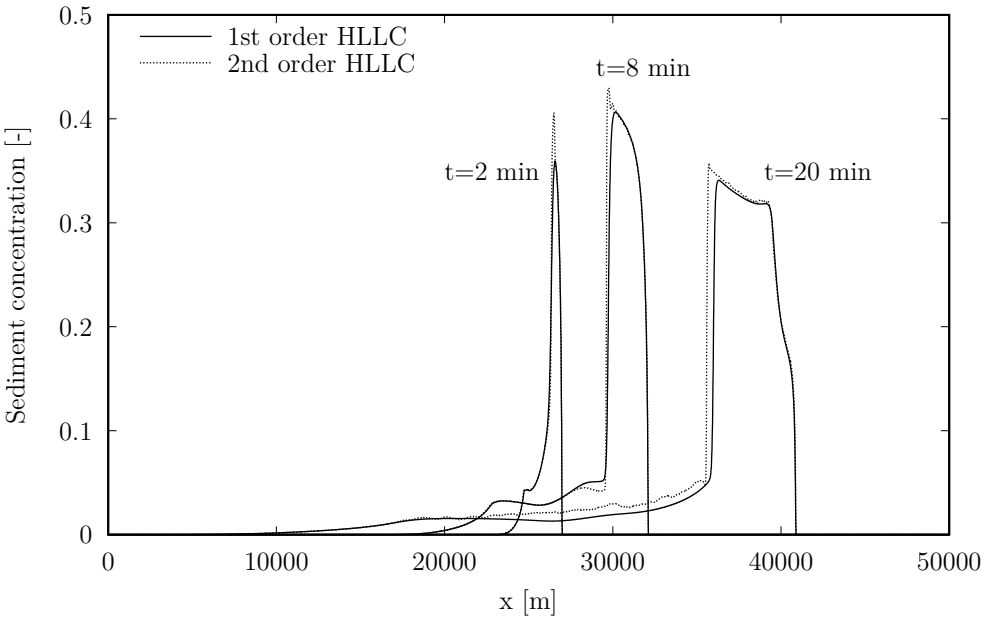


Figure 5.13: Sediment concentration profiles after 2, 8 and 20 minutes

Figure 5.13 shows the sediment concentration profiles after 2, 8 and 20 minutes. A heavily concentrated wavefront is followed by a long tailing with lower concentration. In comparison to the second-order accurate scheme, the first-order accurate HLLC scheme tends to smooth sharp gradients.

5.6.4 Discussion

With the proof of concept it could be shown, that the implementation of new physical transport processes reduces to the actual implementation of the discretized terms in Equation 4.1. It is not necessary to deal with the spatial and temporal discretization, the solution sequence and the code parallelization. The new generic finite volume solver simplifies the implementation of new physical transport processes and to examine different numerical solution approaches. This allows creating complex applications for endusers who use `hms` to simulate water and environment related problems.

The implemented scheme for sediment transport and morphology showed stable results for a complicated test case involving sharp fronts. To evaluate its ability to accurately simulate real sediment transport problems further tests and comparisons with measurements will be necessary. This, however, is beyond the scope of this thesis.

6 Case studies

In this chapter three case studies are used to demonstrate the application of the presented numerical schemes in the framework of `hms`. The first case study is about robustness and accuracy of the presented numerical implementations. An initially dry domain is flooded by a shockwave approaching a simplified urban district. The numerical results are compared to experimental data acquired in a laboratory model. The second case study is a numerical simulation of artificial rainfall experiments on plot-scale carried out in the city Thies, Senegal. In the experiment the hydrograph at the domain outlet was measured. At steady-state conditions the flow velocities in the domain were measured. In addition, tracer experiments were carried out at steady-state rainfall. By simulating all three processes, several aspects of the implemented numerical schemes are demonstrated: surface runoff, tracer transport and infiltration into the soil. The last case study is a real-world application. An unsteady rainfall event in a small alpine catchment was simulated. The measured and simulated hydrographs are compared. In addition the computational performance of the implemented algorithms is evaluated.

In summary, the three case studies together highlight all aspects of the presented numerical schemes. They allow showing the advantages of the approach followed in this thesis. At the same time, they are suitable to point out potential disadvantages and further development possibilities.

Parts of the case studies were extracted from Simons et al. (2012; 2014).

6.1 Flash flood in a simplified urban district

In this case study, simulation results are compared to experimental data of a test case that includes varying bottom topography, wetting and drying, and complex flow conditions. As part of the joint European project Investigation of Extreme Flood Processes & Uncertainty (IMPACT), Testa et al. (2007) acquired experimental data for a test case of a flash flood in a simplified urban district. The domain of consideration

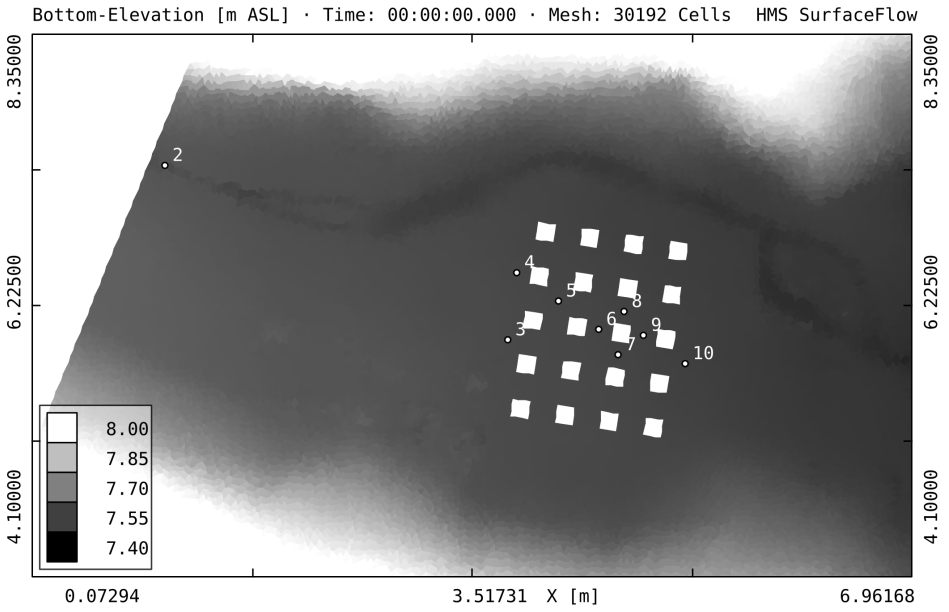


Figure 6.1: Domain topography and locations of water depth gauges (Simons et al., 2014)

was a 1:100 scaled physical model of the Toce River valley (Italy). The topography data was obtained by measurements of the physical model with a spacing of 0.05 m. The idealized city in the considered domain consisted of 20 houses with the positions shown in Figure 6.1. The complete setup of the experiment can be found in Testa et al. (2007). The experimental data was obtained through ten electrical conductivity gauges placed on different spots in the domain under consideration, measuring the water depth every 0.2 s. The locations of the considered gauges are also shown in Figure 6.1. Gauge 1 was located inside the feeding tank of the laboratory model.

The domain was discretized with a total amount of 30 192 triangular cells with an average size of 0.05 m. On the upstream, an inflow boundary condition computed from the measured discharge into the feeding tank and the measured water depths at gauge 1 and 2 was imposed, where the water depth at the boundary was computed by nearest neighbor interpolation. The reason for this were the critical flow conditions at the inflow boundary. Thus, both the flow velocity and water depth had to be given. Downstream, a free outflow boundary condition was chosen. All other boundaries were closed. The data used for this boundary condition is denoted as case “Low” by Testa et al. (2007), which refers to a low peak inflow. The initial water

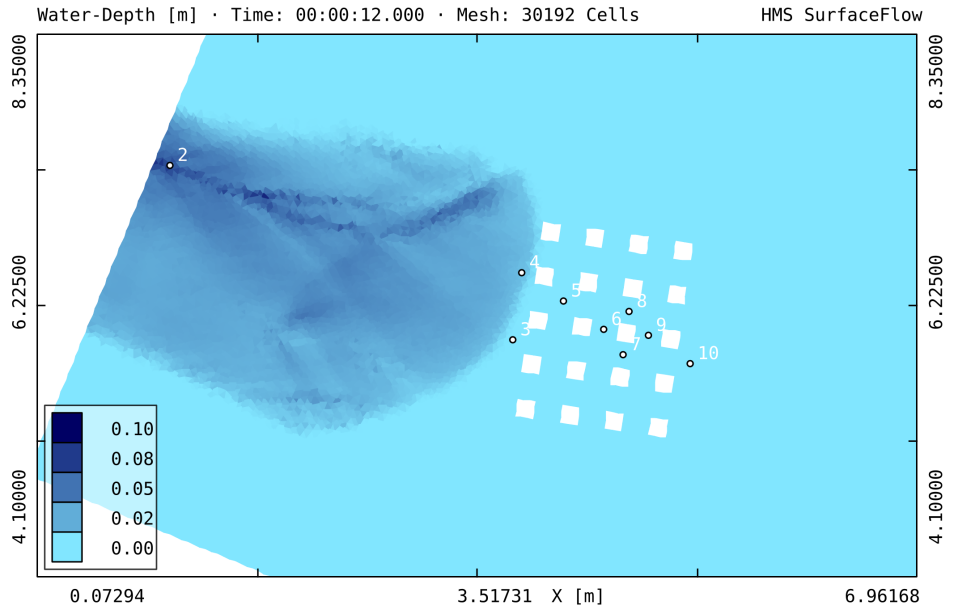


Figure 6.2: Simulated flood propagation after t = 12 s (m) (Simons et al., 2014)

depth and flow velocity was set to zero in the whole domain. Friction was calculated with a Manning coefficient of 0.0162 s $m^{-1/3}$ (Testa et al., 2007), and turbulence was neglected. The time step is calculated adaptively using Equation 3.15 with a CFL criterion set to 0.5. For the simulation the second-order accurate HLLC scheme with the minmod TVD limiter was used.

6.1.1 Results

The simulated flood propagations after 12, 14 and 18 s are plotted in Figures 6.2, 6.3 and 6.4, respectively. To evaluate the flow characteristics, the Froude numbers after 18 s are shown in Figure 6.5. Supercritical inflow was observed (Fr $>$ 1, light grey color). In front of the buildings a hydraulic jump occurred and the flow became subcritical (Fr $<$ 1). Between the buildings and at the downstream side the flow was supercritical again.

In Figure 6.6, the measured water depths at gauges 2 to 10 are compared to the numerical results. A relative L^1-error was computed from the deviation of the observed and simulated values normalized by the average observed value using Equation 5.4 on page 130. The results for the simulated water depths are given in

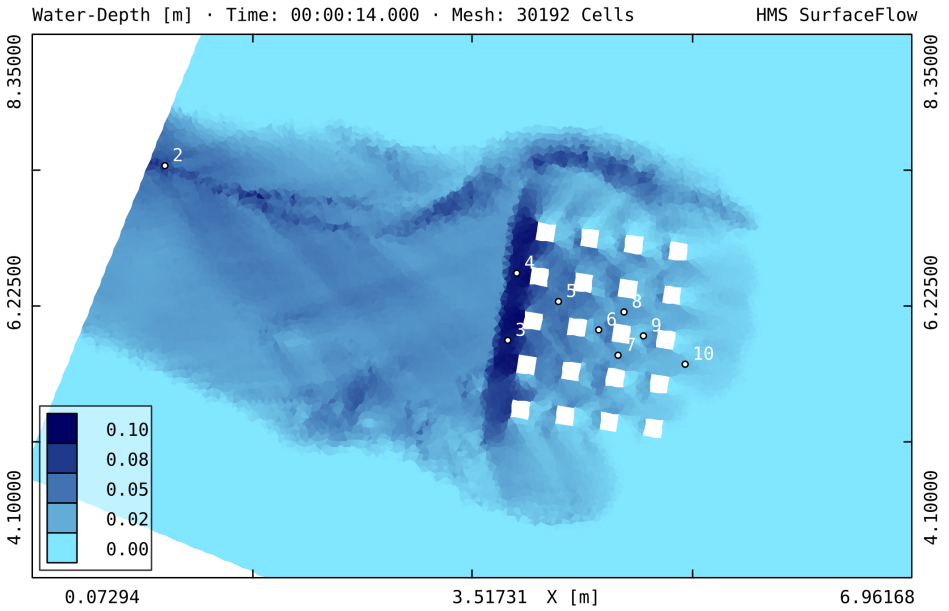


Figure 6.3: Simulated flood propagation after t = 14 s (m) (Simons et al., 2014)

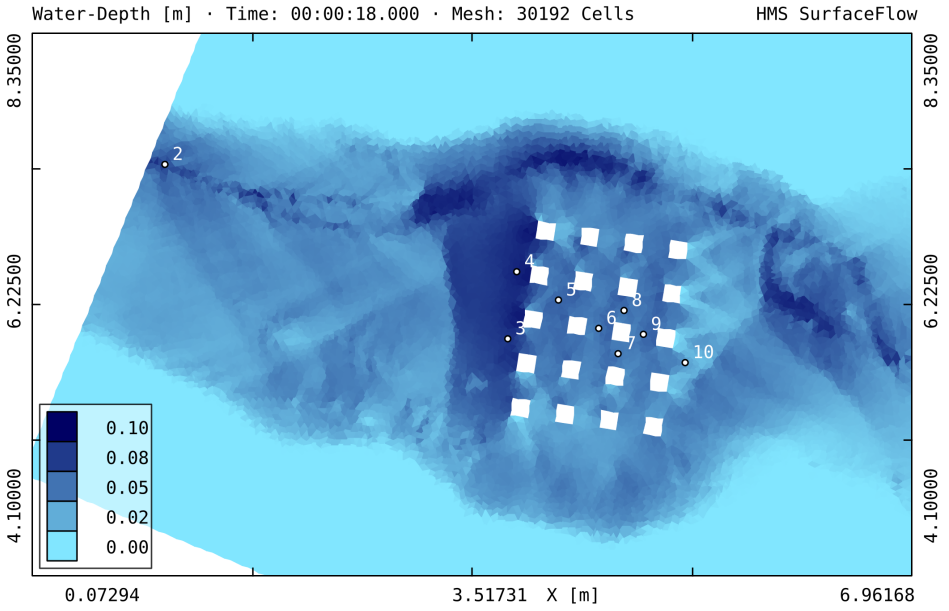


Figure 6.4: Simulated flood propagation after t = 18 s (m) (Simons et al., 2014)

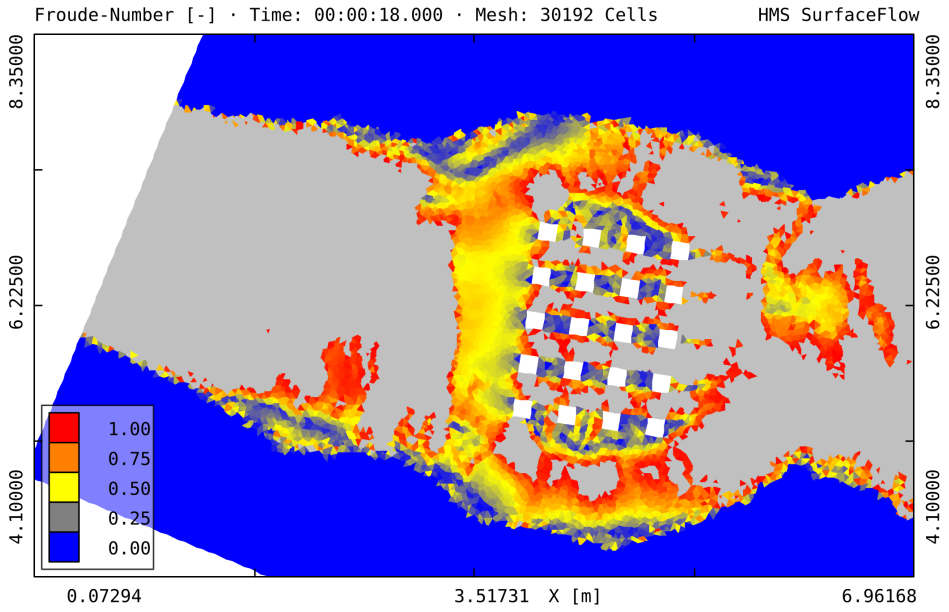


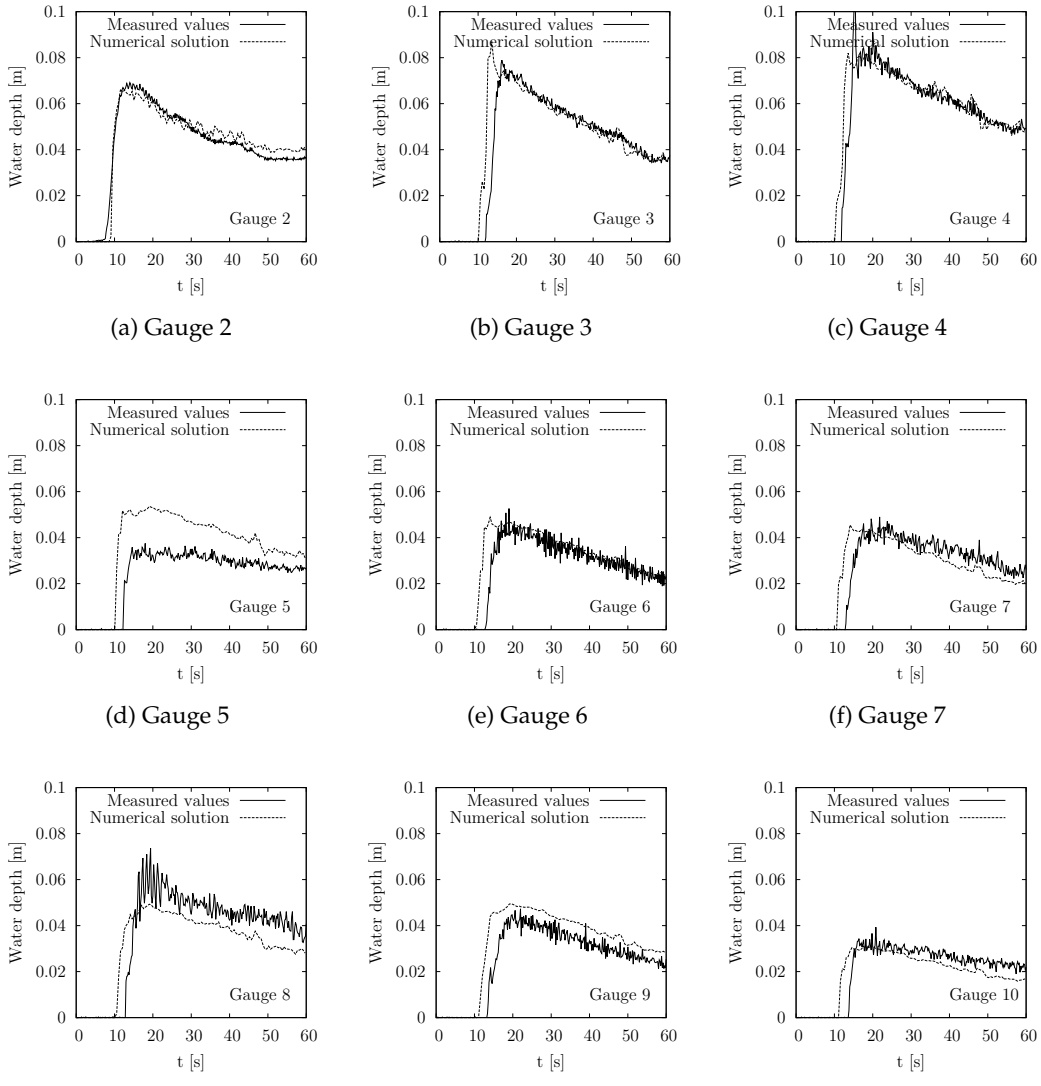
Figure 6.5: Froude number after t = 18 s (values > 1.0 are shown in light grey color)

Table 6.1. Overall, the simulation showed good agreement with the measurements, with a relative error between 7.3 % and 25.5 %. The largest deviation was observed at gauge 5 (Figure 6.6d), with a maximum error of 0.02 m and a relative error of 49.8 %. A similar deviation can also be found in the results of Sanders et al. (2008) and Soares-Frazão et al. (2008), which also could be an indication for an error in the measurements.

The arrival time of the wave showed a systematic error for all gauges. An explanation for this can be the temporal lag between the discharge, which was measured in the inflow pipe to the feeding tank, and the real inflow directly at the border of the laboratory model. To show the influence of such temporal lag, in an additional study the discharge time series has been shifted by 2 seconds. Figure 6.7 shows the

Table 6.1: Relative L^1-errors of the simulated water depth at gauges 2 to 10

Gauge	2	3	4	5	6	7	8	9	10
Relative L^1-error (%)	7.3	10.5	8.2	49.8	15.3	20.0	23.0	25.5	19.7



(a) Gauge 2 (b) Gauge 3 (c) Gauge 4

(d) Gauge 5 (e) Gauge 6 (f) Gauge 7

(g) Gauge 8 (h) Gauge 9 (i) Gauge 10

Figure 6.6: Comparison of numerical and experimental results of the water depth at the gauges 2 to 10

comparison of the experimental and the numerical results obtained from the shifted discharge time series. An improved agreement of the simulated and observed time series is recognized. In Table 6.2 the relative errors are given. All gauges beside 5 and 9 show a reduced error.

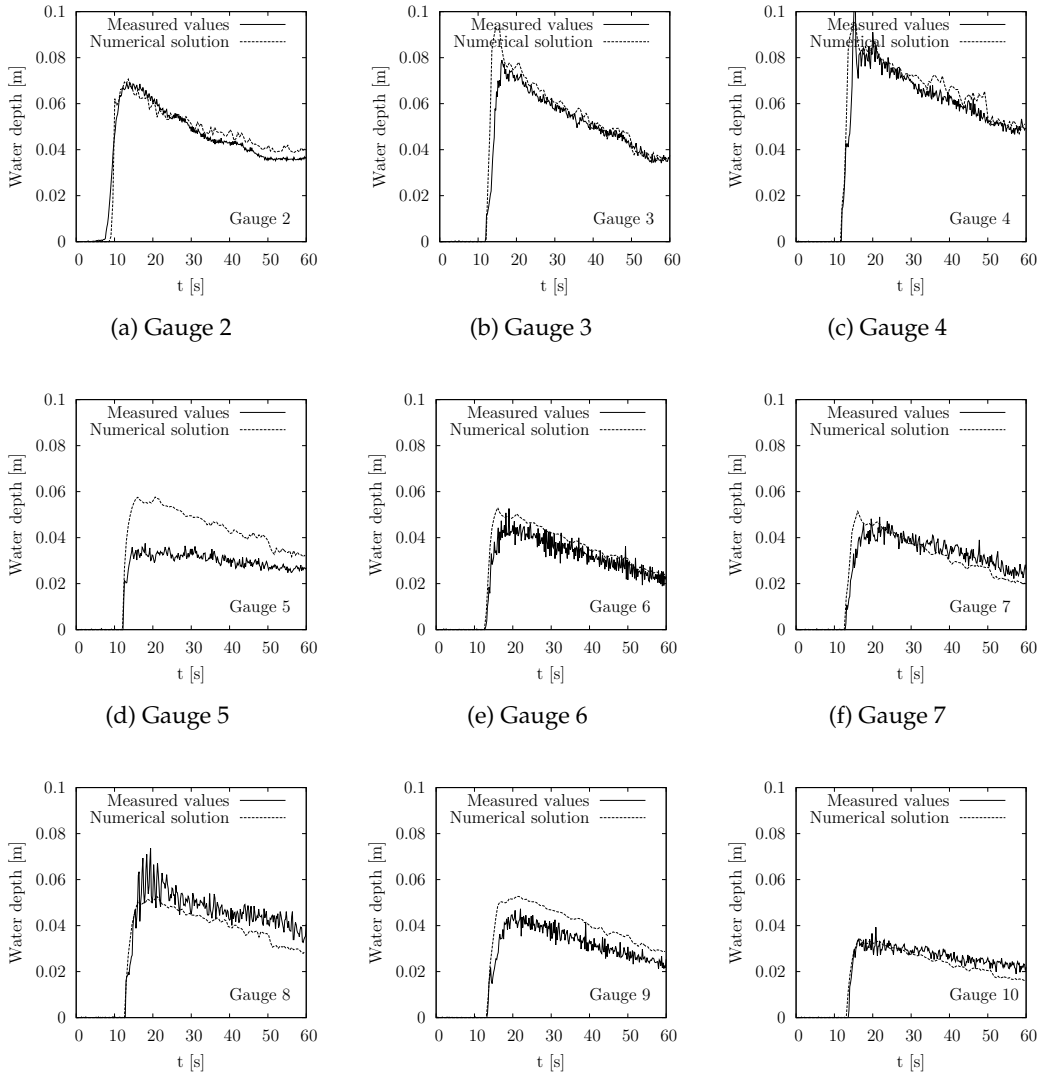
Table 6.2: Relative L^1-errors of the simulated water depth using a time series shifted by 2 seconds for the inflow boundary condition

Gauge	2	3	4	5	6	7	8	9	10
Relative L^1-error (%)	7.5	9.3	7.6	49.4	14.4	13.4	15.2	26.0	12.2

6.1.2 Discussion

The presented case study involved a complex topography and flow field, with sub- and supercritical flow conditions and wetting and drying processes. The numerical scheme showed reasonable and robust results.

Overall, the comparison with the experimental data showed good qualitative agreement. A systematic error in the arrival times for all considered gauges was observed. A possible explanation can be the fact, that the inflow was not directly measured at the inflow boundary, but in the pipe to the feeding tank. The spatial distance can explain a temporal lag of measured and actual inflow.



(a) Gauge 2 (b) Gauge 3 (c) Gauge 4

(d) Gauge 5 (e) Gauge 6 (f) Gauge 7

(g) Gauge 8 (h) Gauge 9 (i) Gauge 10

Figure 6.7: Comparison of experimental and numerical results using a time series shifted by 2 seconds for the inflow boundary condition

6.2 Rainfall simulation experiment (Thies, Senegal)

This case study is about a rainfall simulation experiment which was carried out in Thies, Senegal, and was presented by Tatard et al. (2008) and Mügler et al. (2011). The experimental data is available at UMR LISAH (2004). It is a unique experimental setup which provides valuable data for numerical simulations demonstrating several aspects of the implemented numerical schemes: surface runoff, tracer transport and infiltration into the soil. The example was chosen as a benchmark test because not only the total runoff was measured, but also local flow velocities inside the domain at stationary flow conditions. By comparing measured and simulated velocities, it was possible to evaluate the numerical model's ability to compute both accurate flow field and accurate integral results. Furthermore, tracer experiments were carried out at stationary rainfall and the breakthrough curves at the plot outlet were determined.

Numerical simulations were done by Mügler et al. (2011) for steady-state conditions and by Weill (2007), who used a fully coupled numerical model based on the diffusive wave approximation and Richard's equation on the entire hydrograph.

Here, first the results of the steady-state case are shown to demonstrate the accuracy of the presented numerical model by comparing its results with measured flow velocities. For steady-state conditions, infiltration was neglected and the effective rainfall was used as source in the simulation. Next, the entire hydrograph was simulated. The Green-Ampt equation presented in Section 2.3.3 was used to consider infiltration. The total surface runoff was compared with the measurements. At last, the numerical results for the tracer experiments are shown. As tracer transport is strongly influenced by the accuracy of the computed flow velocities and water depth along the flow path of the tracer, the measured concentration over time at the outlet of the domain was a good measure for the overall accuracy and performance of the numerical model.

6.2.1 Numerical model setup

The experimental setup and the plot preparation has been described in full detail by Mügler et al. (2011). The plot was 10 m long and 4 m wide with a 1 % slope and sandy soil (1 % clay, 7 % silt, 43 % fine and 49 % coarse sand). To reduce the influence of erosion on the experimental results, a long steady rainfall was applied prior to the experiments to allow the development of a stable surface structure. Even so, the topography of the plot was measured twice. Once after the initial rainfall and

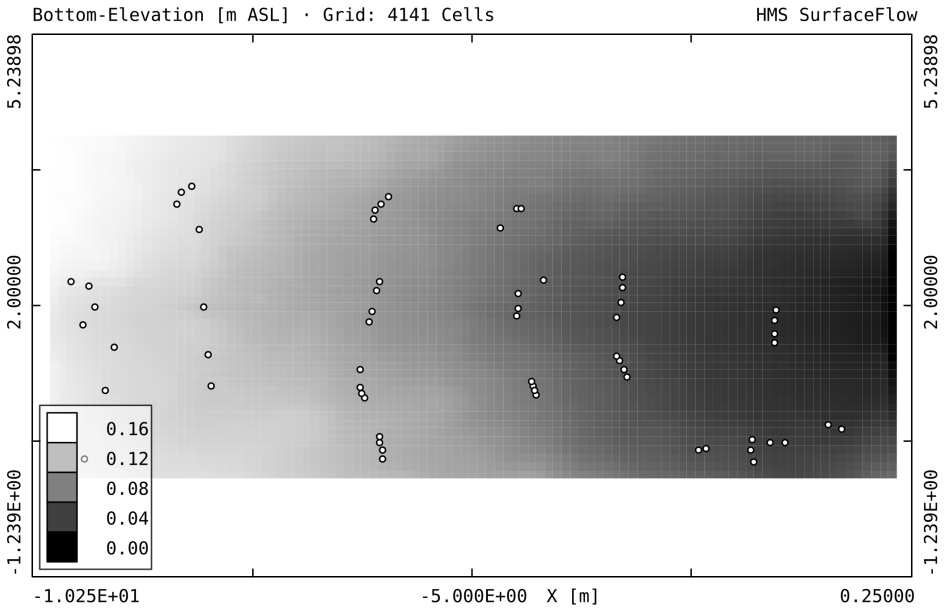


Figure 6.8: Domain topography and positions of velocity measurements for the surface runoff experiments (m) (Simons et al., 2014)

once after the tracer experiments, which were carried out first. For the numerical simulation, the domain topography of the surface runoff experiments and the tracer experiments was given by a 0.1 m $\times$ 0.1 m and a 0.025 m $\times$ 0.025 m digital elevation model (DEM), respectively (Figure 6.8).

For the numerical simulation the domain was discretized by uniform rectangular meshes. For the surface runoff experiments a cell size of 0.1 m was used, resulting in 4 141 cells. For the tracer experiment not the full resolution of the provided DEM was used, but a cell size of 0.05 m, resulting in 15 678 cells.

The upstream boundary condition was set as free outflow; all other boundaries were set as closed. The rainfall was defined as mass source uniformly for all cells (m_w = const. in Equation 2.2). Here, the effective rainfall was used for steady-state conditions. For unsteady conditions an additional sink term for infiltration was used (Equation 2.19). In all simulations the initial water depths and flow velocities were set to zero.

In all simulations, the water-depth-dependent friction formulation of Jain et al. (2004) was used (Equation 2.8). The influence of turbulence was not taken into account in the surface runoff and transport simulations.

All simulations were carried out using the second-order accurate HLLC scheme. Second-order accuracy was achieved using the minmod TVD limiter. The CFL criterion was set to 0.3.

6.2.2 Steady-state surface runoff

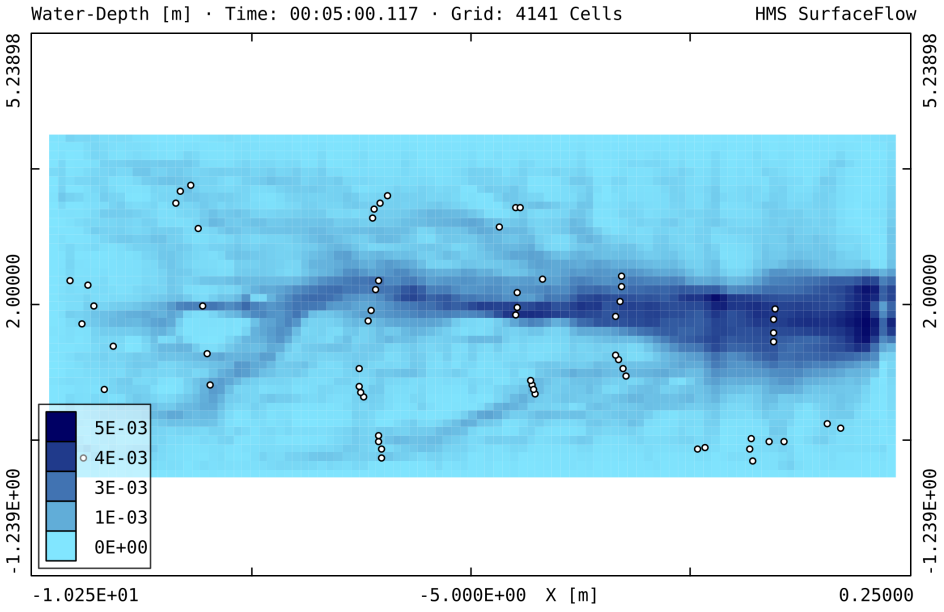
For the steady-state simulation, infiltration was assumed as constant and, corresponding to the steady-state discharge, an effective rainfall of $51.5\,\mathrm{mm\,h^{-1}}$ was applied. The simulation was run until steady water depth was reached with a remaining variation of $1 \cdot 10^{-3}\,\mathrm{m}$. The simulated flow field was compared with mean local flow velocities at 62 probes within the entire plot. Due to missing information on the friction properties, the parameters n_0, d_0 and ε (Equation 2.8) were calibrated by minimizing the L^2-error (Equation 5.12) of simulated and measured flow velocities. Here, calibration was done manually. In a student research project, Adamczak (2012) got comparable results using optimization methods. A minimum $L^2 = 2.55 \cdot 10^{-2}\,\mathrm{m\,s^{-1}}$ was obtained by the parameters $n_0 = 0.014\,\mathrm{s\,m^{-1/3}}$, $d_0 = 0.0045\,\mathrm{m}$ and $\varepsilon = 0.1$. Figure 6.9 shows the spatial distribution of the water depth and flow velocity at steady state and the locations of the measurements. The water depths were in the range of 1 to 5 mm and the flow velocities were in the range of 0.08 to $0.3\,\mathrm{m\,s^{-1}}$. In Figure 6.10 the simulated and measured flow velocity are compared. An overall good agreement was observed. High flow velocities were slightly underestimated by the numerical scheme.

6.2.3 Unsteady surface runoff

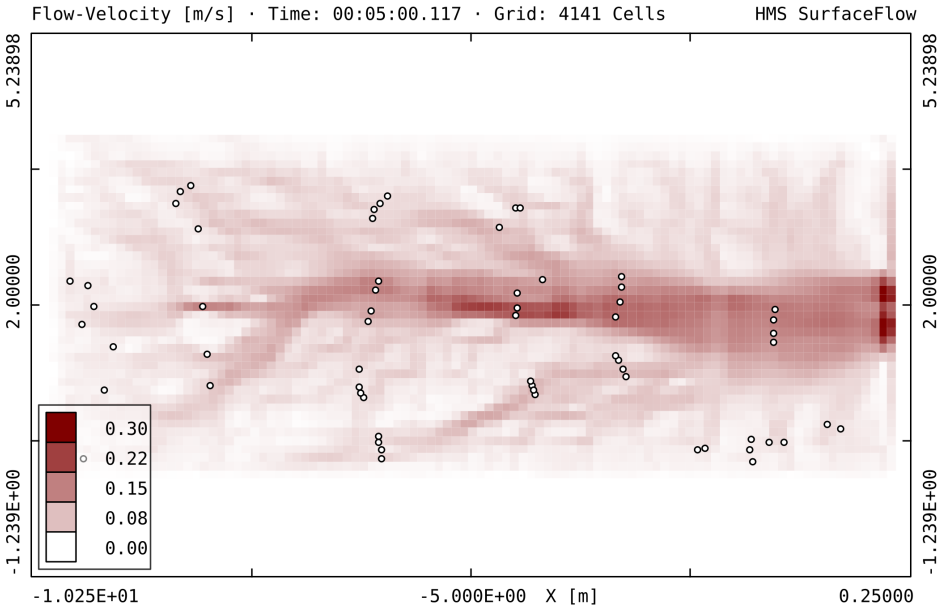
In the second part an unsteady simulation for the entire hydrograph was carried out. During the experiment the rainfall intensity and the total discharge were recorded. For the simulation a rainfall time series with a duration of 5 100 s was used as input. The parameters of the Green-Ampt equation were approximated by the soil type and

Table 6.3: Calibrated parameters for Green-Ampt equation

θ_i	θ_0	$K(h_0)$	h_f
0	0.43	$4.5 \cdot 10^{-6}\,\mathrm{m\,s^{-1}}$	$-0.09\,\mathrm{m}$



(a) Simulated water depth



(b) Simulated flow velocity

Figure 6.9: Simulated water depth (m) and velocity magnitude $(\mathrm{m\,s^{-1}})$ at steady state for the surface runoff experiment (Simons et al., 2014)

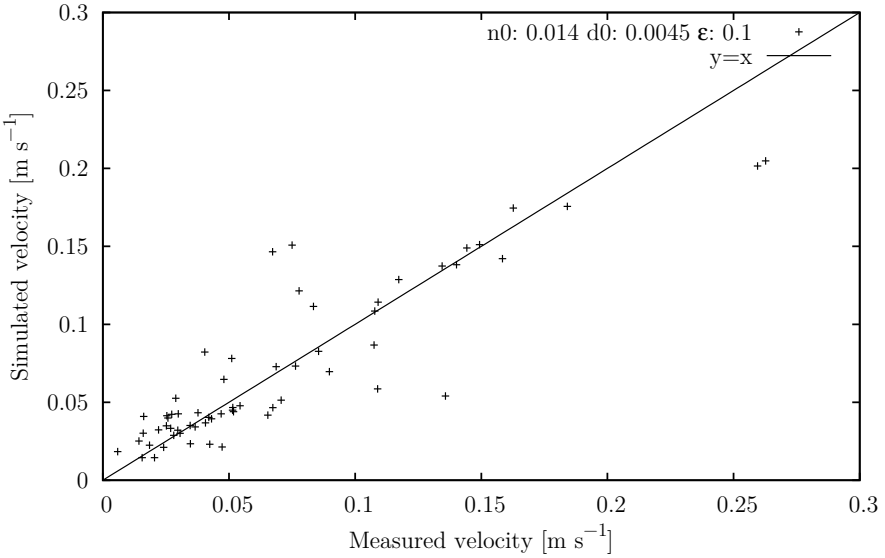


Figure 6.10: Simulated versus measured flow velocities with $L^2 = 2.55 \cdot 10^{-2}\,\mathrm{m\,s^{-1}}$ (Simons et al., 2014)

calibrated from the steady-state infiltration and the total discharge. The calibrated parameters are given in Table 6.3.

In Figure 6.11 the simulated infiltration intensity is compared with the observed intensity, which was obtained by calculating the difference between measured rain intensity and total runoff. The overall trend was reproduced well and the relative error according to Equation 5.4 for the infiltration intensity was 17.6 %. In the beginning of the rainfall, all water infiltrated into the soil. After ~700 s the rainfall intensity exceeded the infiltration rate and water was ponding on the surface.

Figure 6.12 shows the comparison of the measured and simulated total discharge at the outlet of the domain. A good qualitative agreement was observed with a relative error of 13.7 %. However, the measured discharge in the beginning of the event was not captured. Possible explanations are surface coating effects which are not captured by the Green-Ampt equation.

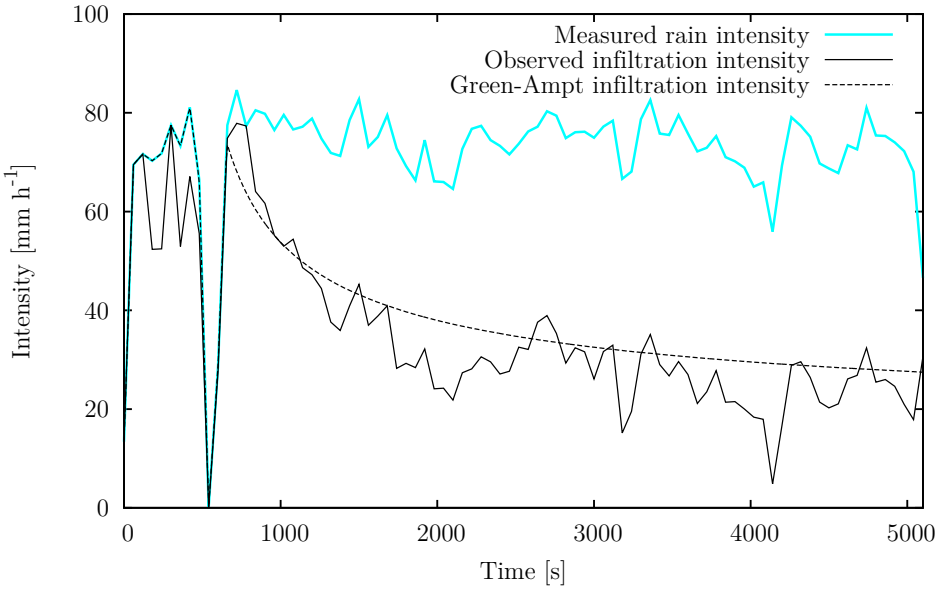


Figure 6.11: Comparison of observed and simulated infiltration intensities (Simons et al., 2014)

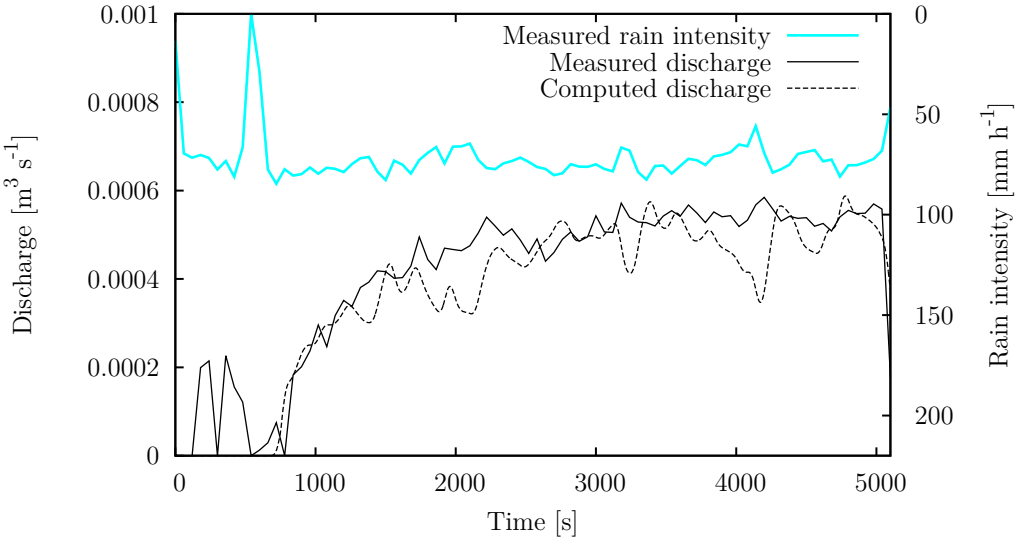


Figure 6.12: Comparison of measured and simulated hydrographs (Simons et al., 2014)

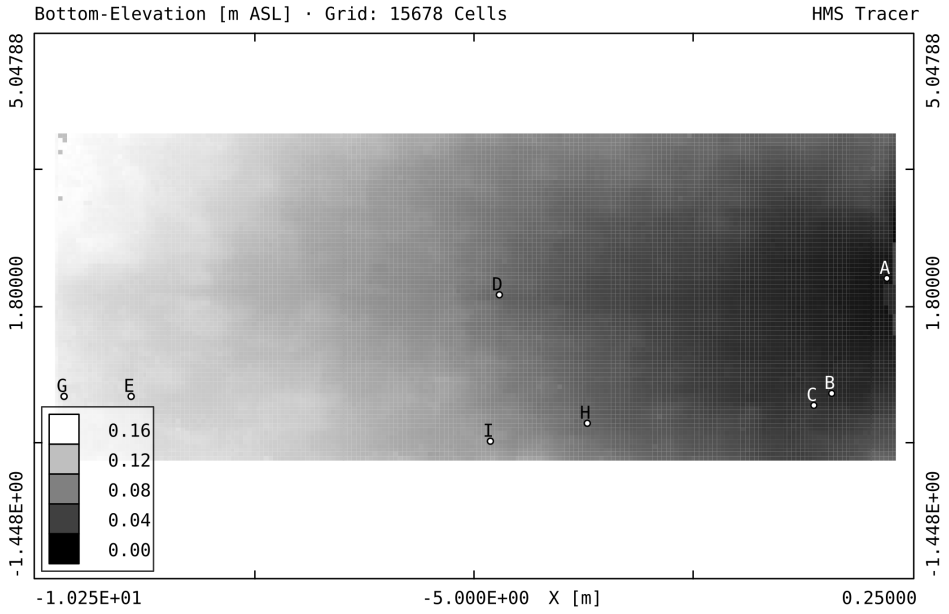


Figure 6.13: Domain topography and injection points for the tracer experiment (m)

6.2.4 Tracer experiments

In the tracer experiment, a chemical mixture was injected consecutively at eight locations throughout the plot after a steady runoff could be observed. The locations are given in Figure 6.13. Injection point A was placed at the domain outlet and used for calibration of the concentration measurement. The injection point G, H and I were placed at the origins of the main channel, while the injections point D and E were directly placed in the main channel. The measurements of injection points B and C appeared to be not useful and were disregarded (Mügler et al., 2011). To determine the tracer concentration over time, the electrical conductivity of the discharge at the outlet of the plot was measured.

Over 30 seconds $1 \cdot 10^{-3}\,\mathrm{kg\,s^{-1}}$ of the tracer mixture was injected. With a given density of $1437\,\mathrm{kg\,m^{-3}}$ for the mixture and a cell size of 0.05 m, this equals a specific mass flux of $1.4 \cdot 10^{-5}\,\mathrm{m\,s^{-1}}$, which was assigned as source term m_c in the advection-diffusion equation (Equation 2.33) for the cell containing the injection point. To not only consider the increasing tracer concentration, but the increased fluid volume, this value was added to the mass source term of the shallow water equations, too. In the steady-state simulation, infiltration was assumed to be constant and and

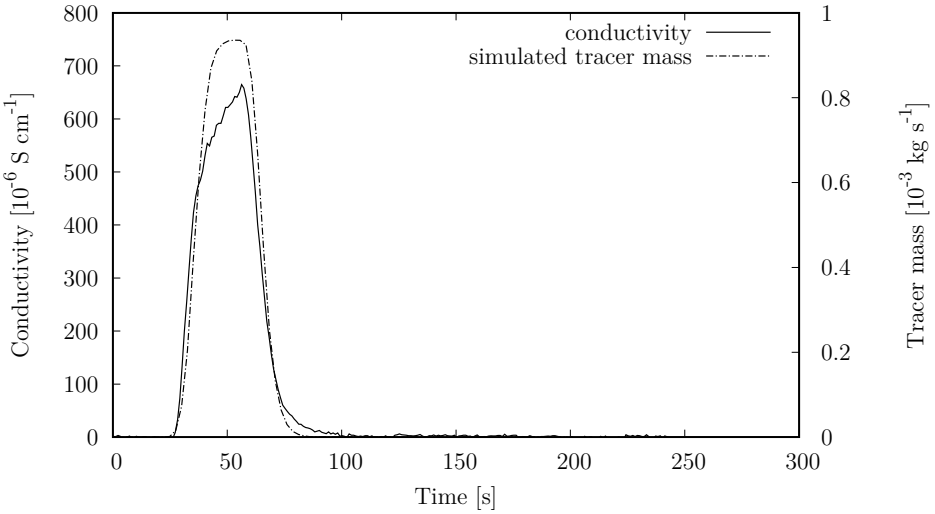


Figure 6.14: Tracer breakthrough curve for tracer injection at point D

effective rainfall of $48.5\,\mathrm{mm\,h^{-1}}$ was applied in all cells. With a total rainfall of $75\,\mathrm{mm\,h^{-1}}$, this resulted in a constant infiltration of $26.5\,\mathrm{mm\,h^{-1}}$. As the effective rainfall was applied as source term in the mass balance equation of the shallow water equations, the infiltration was not considered explicitly. However, as the tracer concentration decreases when the polluted water infiltrates the soil, a specific mass flux of $-26.5\,\mathrm{mm\,h^{-1}}$ or rather $-7.4 \cdot 10^{-6}\,\mathrm{m\,s^{-1}}$ was considered in all cells as sink in the advection-diffusion equation (Equation 2.33), if the tracer volume cd was bigger zero.

As the DEM of the tracer experiment differs from the one of the surface runoff experiments, the friction parameters were calibrated by minimizing the error of simulated and measured tracer breakthrough curves. Best overall agreement was achieved with the parameters $n_0 = 0.0165\,\mathrm{s\,m^{-1/3}}$, $d_0 = 0.003\,\mathrm{m}$ and $\varepsilon = 0.8$.

In the Figures 6.14, 6.15, 6.16, 6.17 and 6.18 the measured and simulated breakthrough curves at the plot outlet for the tracer injections at the points D, E, G, H and I are given. Using information about the calibration process given by the authors, a conductivity of 0 to $800 \cdot 10^{-6}\,\mathrm{S\,cm^{-1}}$ was hereby mapped to a tracer mass flux of 0 to $1 \cdot 10^{-3}\,\mathrm{kg\,s^{-1}}$ (Mügler et al., 2011, via E-mail).

The tracer arrival time was in good agreement with the observed breakthrough curves. At point E and I perfect agreement of numerical and experimental data was

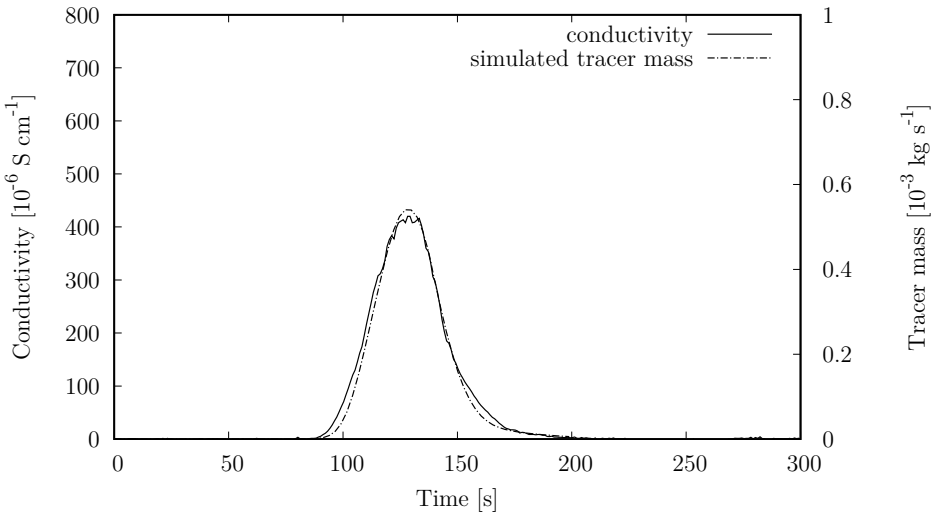


Figure 6.15: Tracer breakthrough curve for tracer injection at point E

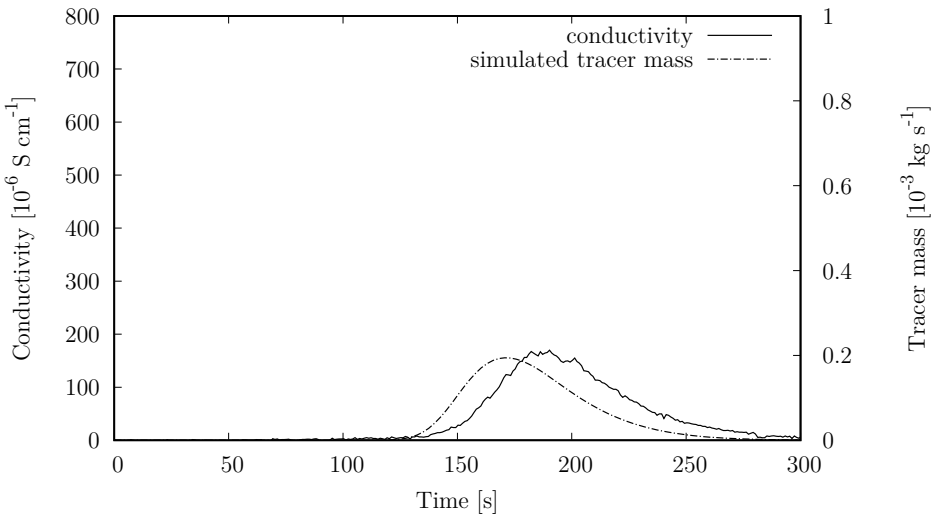


Figure 6.16: Tracer breakthrough curve for tracer injection at point G

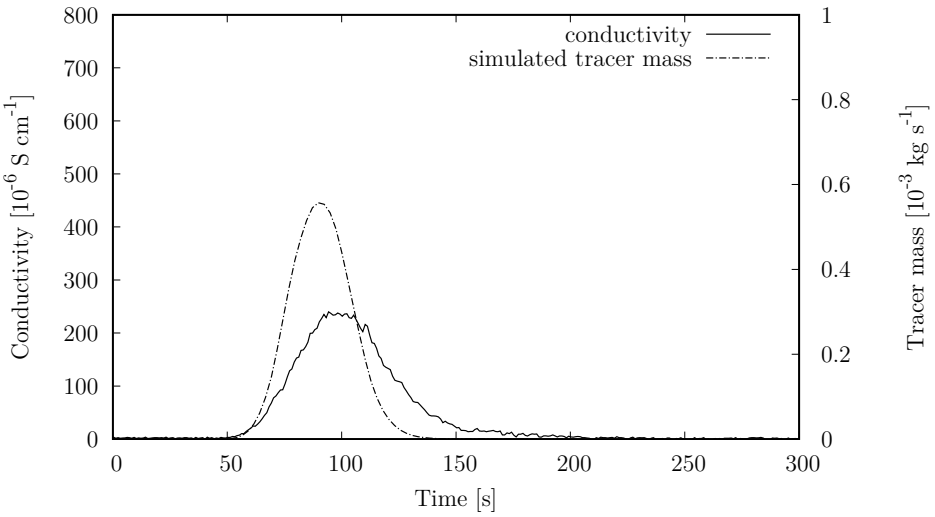


Figure 6.17: Tracer breakthrough curve for tracer injection at point H

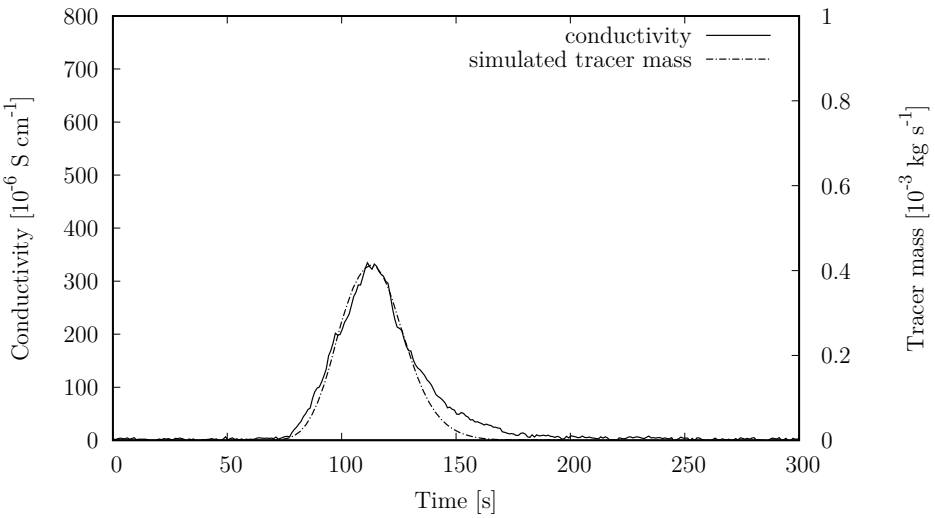


Figure 6.18: Tracer breakthrough curve for tracer injection at point I

Table 6.4: Mass recovery percentage at the outlet from experimental and simulated data

Injection point	Experiment (%)	Simulation (%)
D	82	94
E	57	64
G	44	39
H	53	63
I	50	49

observed. At point G the peak of the breakthrough curve was shifted to the left, which means the tracer in the simulation travelled faster than in the experimental setup. Injection point G was located at the very top of the experimental plot where very small water depth occur ($d \ll 1\,\text{mm}$). The calibration process aimed for an overall good agreement. As the other injection points were located in regions with water depth greater than one millimeter, the roughness parameters did not perfectly capture the conditions at injection point G. However, a more sophisticated parameter optimization approach could improve this result.

The simulated breakthrough curve at injection point H had a much higher peak value. As this difference was also shown by Mügler et al. (2011), it could be either strong dissipation or infiltration of the tracer not captured in the numerical model or errors in the conductivity measurements.

In Table 6.4, the mass recovery percentage at the outlet from experimental and simulated data is given. The error between experiment and simulation was around 10 % for all injections.

In the Figures 6.19, 6.20, 6.21, 6.22 and 6.23 the tracer flow paths for the five injection points are given at different times.

6.2.5 Discussion

In this case study an artificial rainfall experiment on plot scale was simulated. In the experiments the total runoff and local flow velocities inside the domain were measured. In addition, tracer experiments were carried out at steady-state

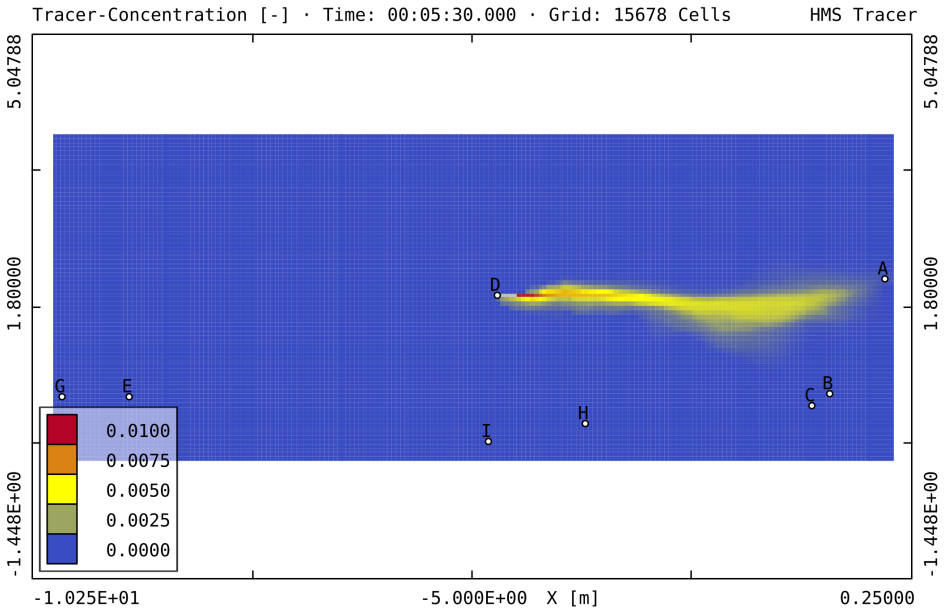


Figure 6.19: Tracer concentration 30 s after start of injection at point D

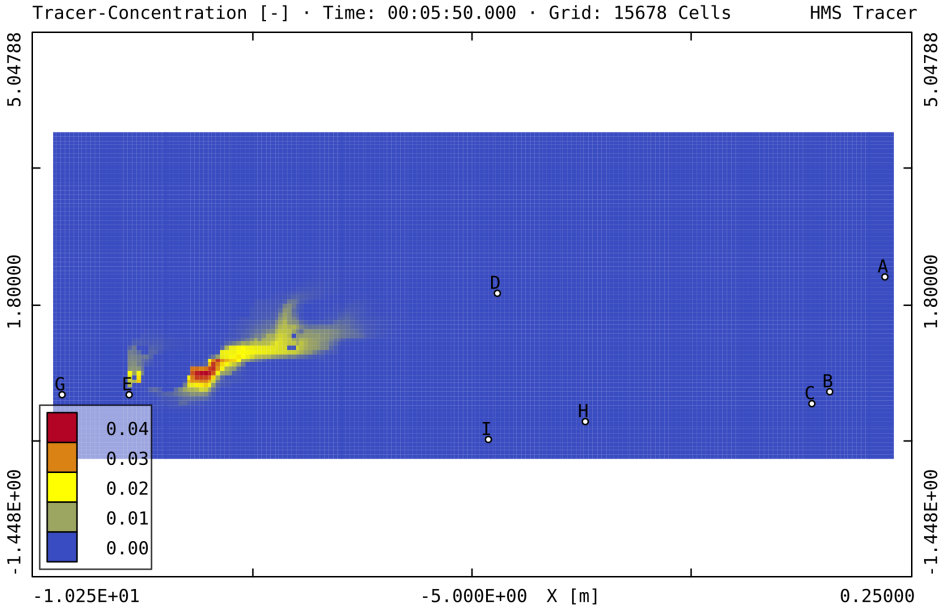


Figure 6.20: Tracer concentration 50 s after start of injection at point E

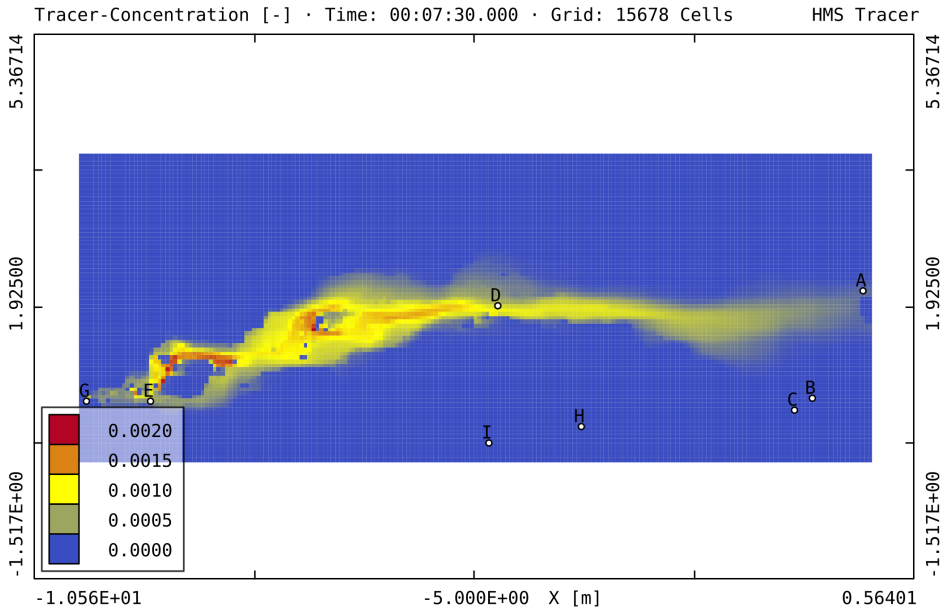


Figure 6.21: Tracer concentration 150 s after start of injection at point G

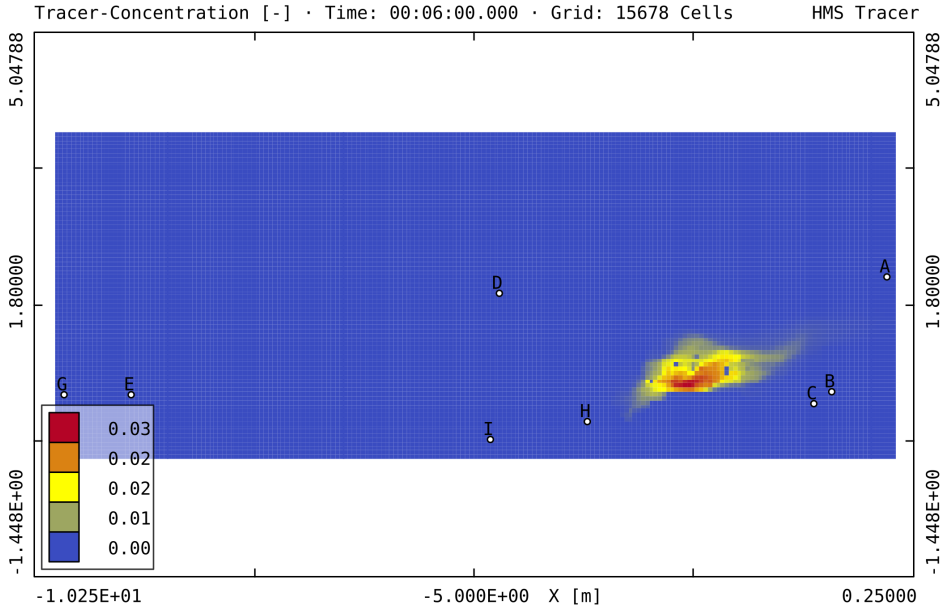


Figure 6.22: Tracer concentration 60 s after start of injection at point H

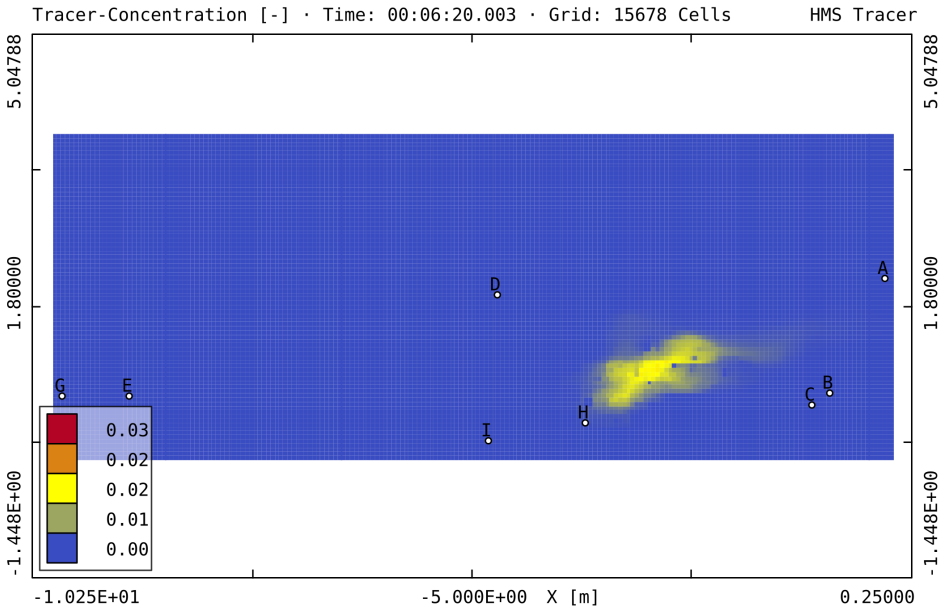


Figure 6.23: Tracer concentration 80 s after start of injection at point I

flow conditions. The available data allowed investigating the physical processes interacting in a rainfall-runoff event and their reproduction in the numerical model.

In the steady-state simulations it was shown that the presented numerical model accurately simulated the surface runoff process. In the unsteady simulations, the simplifications of the Green-Ampt equation infiltration model could be seen. A good qualitative agreement of the simulated and measured hydrograph could be achieved. However, the runoff generation is of course a crucial part and should be improved in the future.

In the last section, the numerical tracer transport model was included. The shape of the breakthrough curves was strongly influenced by the flow field. The results showed good agreement for most of the injections points and errors in the mass recovery were within 10 %.

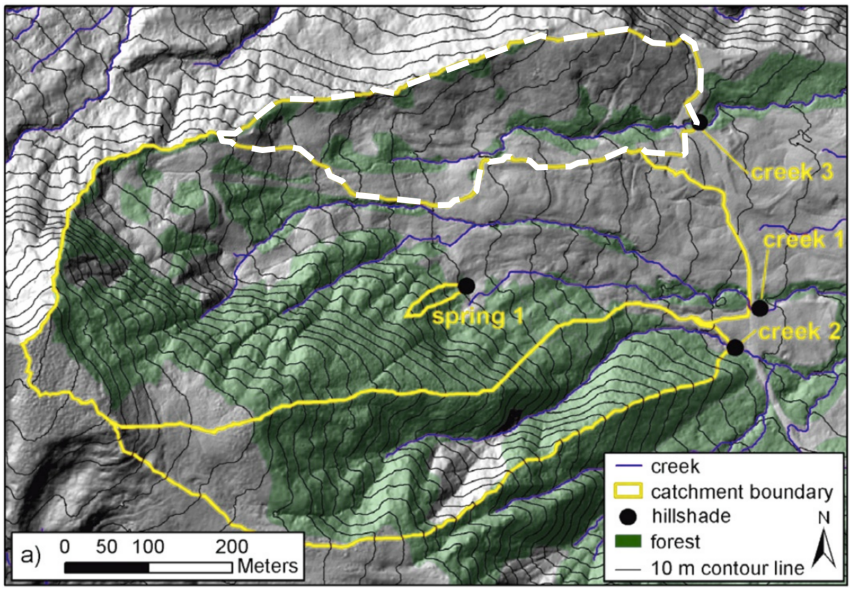


Figure 6.24: Sub-catchment boundaries at the Heumöser slope (Lindenmaier, 2008)

6.3 Rainfall-runoff simulation at Heumöser

The aim of the DFG Research Unit "Coupling of flow and deformation processes for modeling the movement of natural slopes" was to understand and predict the failure of natural slopes (Hinkelmann and Zehe, 2007). Rainfall, surface runoff and infiltration are possible triggers for landslides and have been investigated in this project. The study area Heumöser is located in the Vorarlberg Alps (Austria), 10 km south-east of the city of Dornbirn. Further information on the project can be found in Hinkelmann and Zehe (2013), Hinkelmann and Zehe (2007), Lindenmaier (2008), Wienhöfer et al. (2009), Stadler et al. (2012) and Stadler (2016). In this case study the surface runoff in the sub-catchment area of creek 3 with an area of ~ 100 000 m² (Figure 6.24, dashed outline) was simulated for a rain event in July 2008 (Simons et al., 2011). In total, a rainfall of ~ 160 mm in ~ 100 h was measured. The peak discharge at the outlet of the sub-catchment was 0.2 $m^3 s^{-1}$. The bottom elevation was defined by a digital elevation model provided by the "Wildbach- und Lawinenverbauung", Austria, with a resolution of 1 m × 1 m. All simulations were carried out on a 1 m² uniform rectangular mesh with 147 400 cells. At all boundaries free outflow conditions were defined. The initial water depths and flow velocities were set to zero. The rainfall was considered as a source term in Equation

Figure 6.25: Discharge measurement at creek 3

2.2 and infiltration was considered by a constant runoff-coefficient, i. e. a constant amount was subtracted from the measured rainfall. The influence of turbulence was neglected. To compare measured and simulated discharge, the discharge in the cells located in the cross section of creek 3 at the outlet of the sub-catchment was monitored (Figure 6.25). Figure 6.26 shows a photograph of surface runoff after a heavy rainfall at Heumöser.

6.3.1 Sensitivity analysis

First, the influence of the friction- and the runoff-coefficient was studied. The runoff-coefficient is defined as:

$$\Psi = \frac{\text{surface runoff}}{\text{total rainfall}} \tag{6.1}$$

In Figure 6.27a the comparison of measured and simulated discharges for two different runoff-coefficients Ψ and a constant Manning coefficient of $0.067\,\mathrm{s\,m^{-1/3}}$ is given. The results showed only qualitative agreement and very high relative errors (Equation 5.4) of 78.0 % for $\Psi = 0.3$ and 64.7 % for $\Psi = 0.6$. In Figure 6.27b the hydrographs for Manning coefficients $0.143\,\mathrm{s\,m^{-1/3}}$, $0.067\,\mathrm{s\,m^{-1/3}}$ and $0.033\,\mathrm{s\,m^{-1/3}}$ and a constant runoff-coefficient of $\Psi = 0.6$ are given. The results showed small

Figure 6.26: Surface runoff after heavy rainfall

sensitivity and only a slight damping of the peak discharges was observed.

The poor agreement of the overall results can be explained by the fact that the numerical surface runoff model does not take into account any slower discharge component in the subsurface such as interflow or groundwater flow.

6.3.2 Simple consideration of interflow

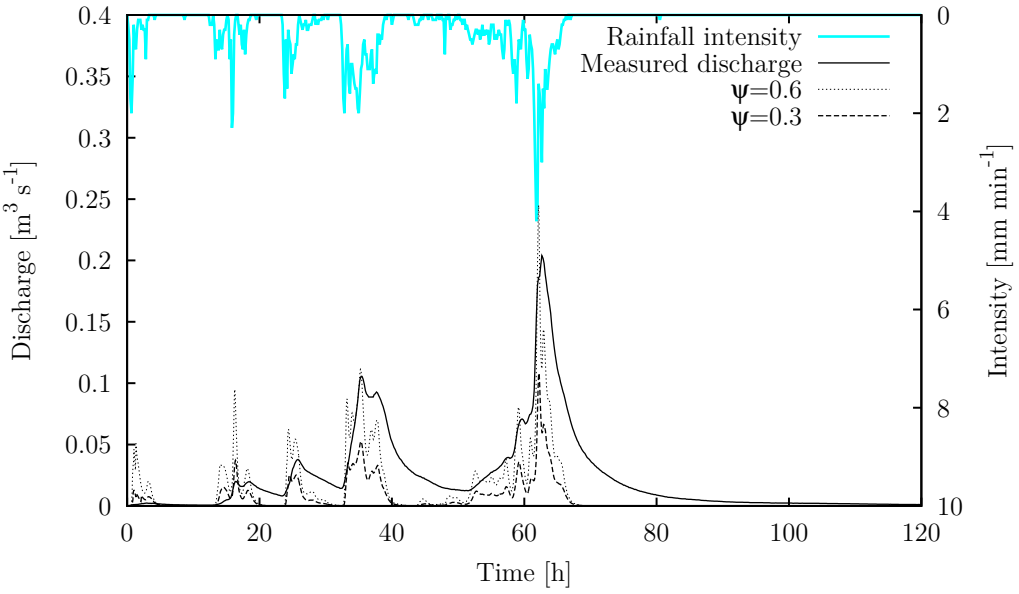
As the focus of this thesis is on surface water flow, a simple linear reservoir model was added to account for an interflow component. It is described by the continuity equation for the reservoir:

$$\frac{\mathrm{d}S\left(t\right)}{\mathrm{d}t} = I\left(t\right) - Q\left(t\right) \tag{6.2}$$

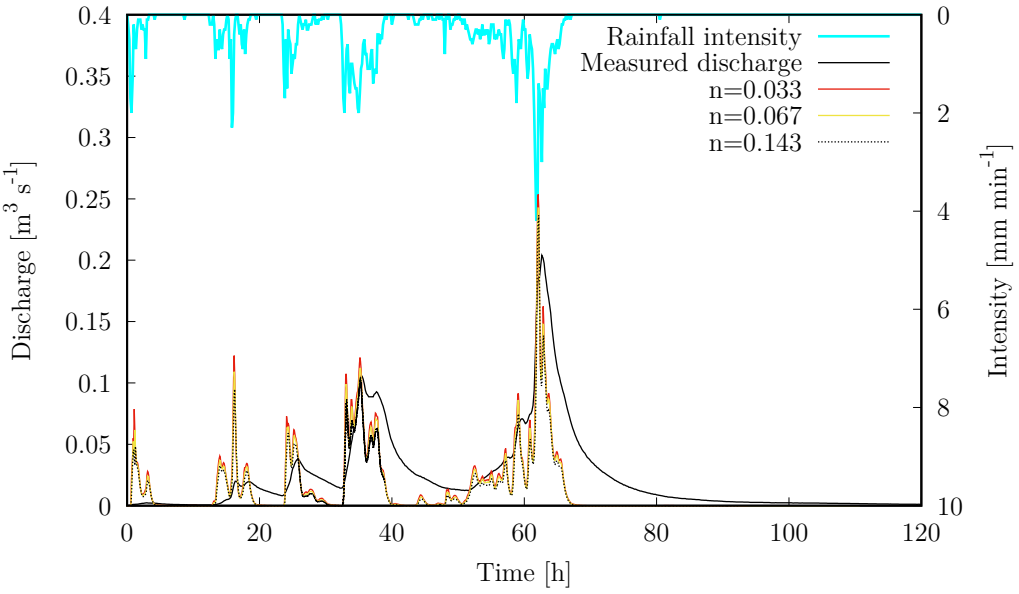
and a storage-discharge relation:

$$S\left(t\right) = K\,Q\left(t\right) \tag{6.3}$$

where $S\left(t\right)$ is the storage at time t, $I\left(t\right)$ is the inflow and $Q\left(t\right)$ is the outflow of the reservoir (e. g. Duggal and Soni, 1996). The constant of proportionality K is the average residence time and was obtained by calibration. The infiltrated



(a) Comparison of different runoff coefficients



(b) Comparison of different Manning coefficients n [$\mathrm{s\,m^{-1/3}}$]

Figure 6.27: Simulated hydrographs of the sub-catchment “creek 3”

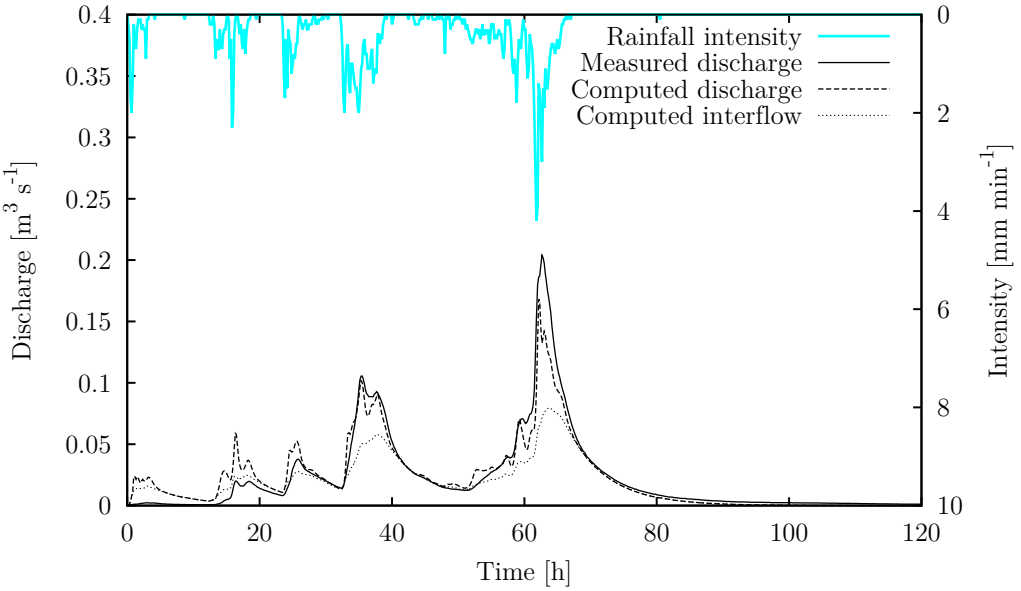


Figure 6.28: Simulated hydrograph of the sub-catchment "creek 3" with consideration of interflow via linear reservoir model

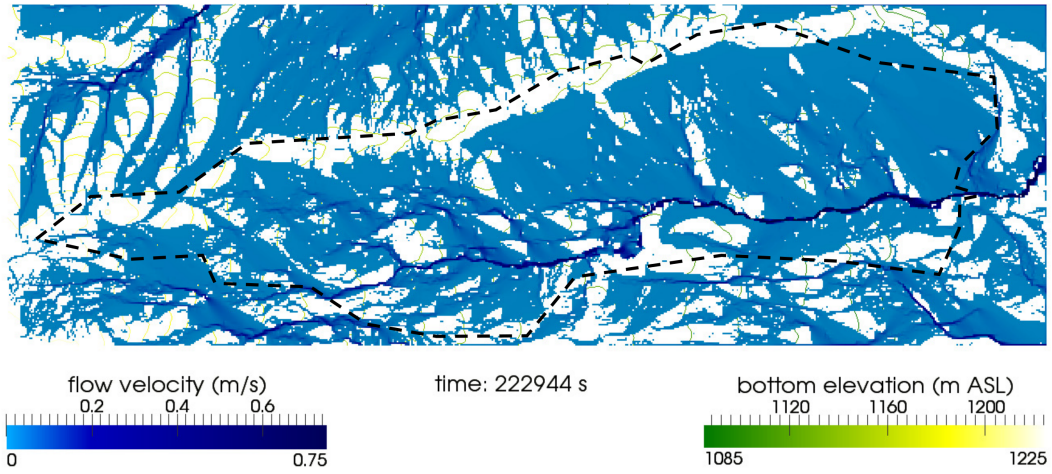


Figure 6.29: Simulated flow field at t = 62 h (flow velocities are shown for water depth > 1 mm)

amount of water which is determined by the runoff-coefficient was used as inflow and the computed outflow was superimposed with the surface runoff to obtain the total discharge of the domain. For the runoff coefficient $\Psi = 0.3$ and the Manning coefficient $0.067\,\mathrm{s\,m^{-1/3}}$, an average residence time of $K = 6\,\mathrm{h}$ led to the best agreement of computed and measured discharges and a relative error of 26.5 % which corresponds to mean absolute error of $0.006\,\mathrm{m^3\,s^{-1}}$. Figure 6.28 shows a comparison of measured and computed discharges. A reason for the remaining error could be due to neglecting exfiltrating water, which is an important process at steep slopes.

Figure 6.29 presents the flow field in the domain at $t = 62\,\mathrm{h}$ when the highest discharge was measured for water depth larger than 1 mm. Small creeks with high flow velocities ($> 0.6\,\mathrm{m\,s^{-1}}$, dark blue color) have developed in the domain, while the flow velocities on the surrounding surfaces are below $0.1\,\mathrm{m\,s^{-1}}$ (light blue color). The numerical model was able to handle very small water depth and alternations between sub- and supercritical flow

6.3.3 Parallel run-time behavior

In this section the parallel run-time behavior of the numerical algorithms is evaluated. Hinkelmann (2005) introduced two performance numbers to evaluate the performance quality. The *parallel speedup* is defined as:

$$s_p = \frac{\text{run time on 1 core}}{\text{run time on } p \text{ cores}} \tag{6.4}$$

The theoretical speedup increases linear with an increasing number of cores. In real applications it is always below the number of cores, i. e. $s_p\,(p) < p$.

The *parallel efficiency* can be seen as a scaling of the parallel speedup (Hinkelmann, 2005). It is defined as:

$$e_p = \frac{\text{run time on 1 core}}{p \cdot \text{run time on } p \text{ cores}} = \frac{s_p}{p} \tag{6.5}$$

Analogous to the parallel speedup, the parallel efficiency is always below the theoretical value for real applications, i. e. $e_p\,(p) < 1$ (or 100 %).

Computer scientist Gene Amdahl stated, that the speedup due to parallelization is limited by the sequential part of the the program (Amdahl, 1967). Amdahl's law can

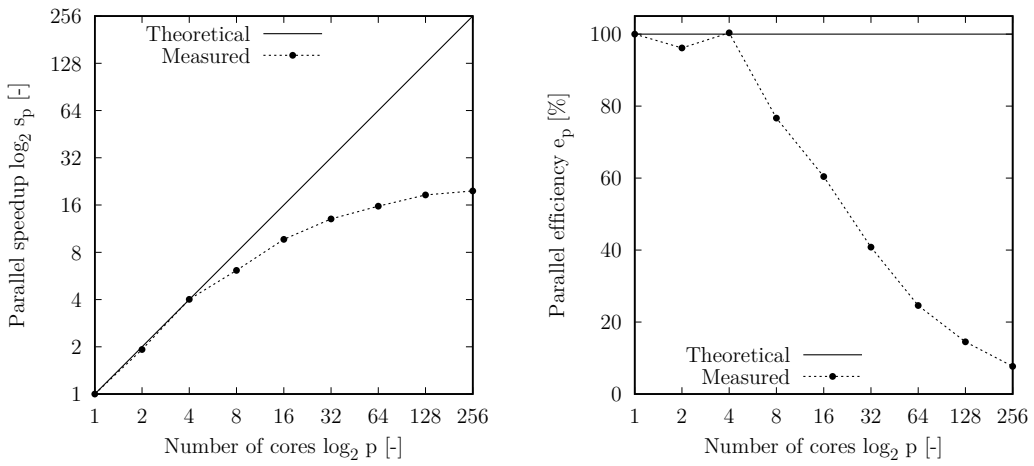


Figure 6.30: Parallel speedup and efficiency on the SMP1 system

be used to predict the *theoretical speedup*:

$$s_t(p) = \frac{1}{(1-\eta_p) + \frac{\eta_p}{p}} \tag{6.6}$$

where η_p is the proportion of the program code which can benefit from the parallelization. For an infinite number of cores p, the theoretical speedup is limited by the amount of sequential code in the program:

$$\lim_{p\to\infty} s_l(p) = \frac{1}{1-\eta_p} \tag{6.7}$$

The parallel computations were carried out on the supercomputing system of the North-German Supercomputing Alliance (HLRN, 2016). As `hms` uses shared-memory parallelization, the SMP (symmetric multiprocessing) cluster at the Gottfried complex in Hannover was used. This cluster consists of 64 MEGWARE compute blades. Each of the 32 nodes in the SMP1 system is equipped with four Intel Xeon 8-core SandyBridge (E5-4640 @2.7GHz) processors and 256 GiB RAM. Thus, 32 cores are available on each node and in total 1 024 cores and 8 192 GiB RAM are available on the SMP1 system and can be used for shared-memory computations.

In Figure 6.30 the parallel speedup and efficiency on 1 to 256 cores are shown. For up to 4 cores an almost theoretical speedup and a consistent efficiency was observed.

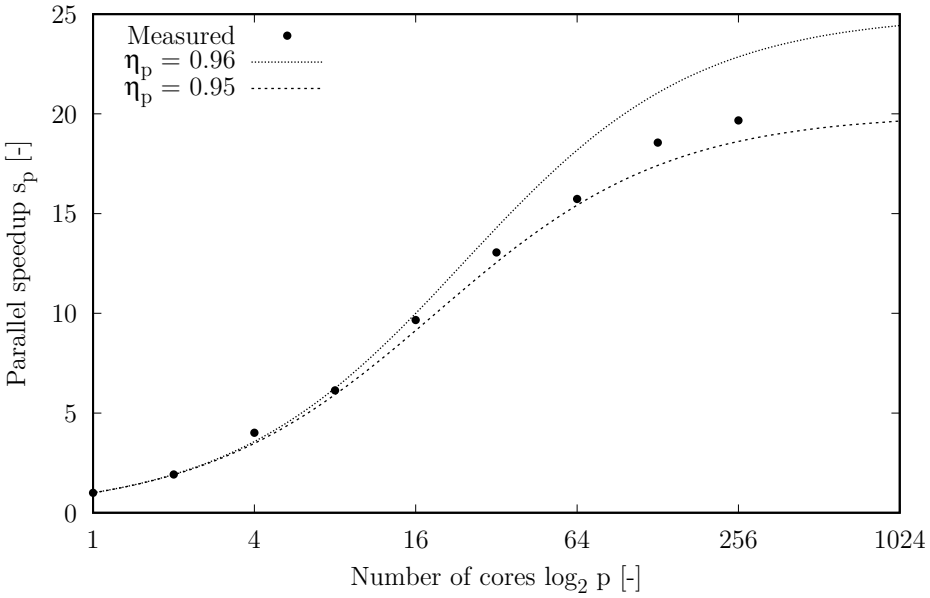


Figure 6.31: Comparison of measured parallel speedup and the theoretical speedup according to Amdahl's law

For more than 4 cores, the parallel overhead caused a significant drop in the parallel efficiency. In the same time the inclination of the parallel speedup decreased, however, the speedup itself does still increased. The results showed that for an application with just 147 400 cells more than 8 cores (<18 425 cells/core) are not suitable as the parallel efficiency decreases significantly. Overall, an efficient parallelization was achieved for up to 8 cores and an increased speedup was achieved for up to 256 cores.

In Figure 6.31 the measured speedup is compared with the theoretical speedup according to Amdahl's law for a parallel portion of 95% and 96% (Equation 6.6). It can be seen, that the parallel speedup for up to 16 cores corresponded to the theoretical speedup expected of program with 96% of parallel code. For a higher core count, the measured speedup never drops below the theoretical speedup for 95% parallel code.

6.3.4 Discussion

In this application, a simulation of a real rainfall event in a small alpine catchment was investigated. As in the previous application (Section 5.3), infiltration in the soil strongly influenced the amount of surface runoff. In addition, the discharge of a steep natural catchment area is strongly influenced by interflow processes.

The following simplifications were made: Infiltration was considered by a constant runoff-coefficient. Rainfall intensity and friction were assumed to be uniform in the whole catchment. An interflow component was added in the form of a linear reservoir model. All parameters were found by manual calibration. Here, automatic optimization methods could improve the overall results.

In a next step the numerical surface water flow model should be coupled with a numerical model for flow in the unsaturated zone for a complete physically based description. This would not only allow an improved description of infiltration and interflow, but also the consideration of exfiltration. Significant research has been carried out on simulating flow processes in the unsaturated zone of the Heumöser area (Stadler et al., 2012; Stadler, 2016) and model coupling was already investigated by Mieth (2008).

Using the simplified approaches to consider infiltration and interflow, good agreement of the simulated and measured hydrograph could be obtained. The numerical model was able to handle very small water depth and alternations between sub- and supercritical flow. Finally, the parallel run-time behavior was evaluated. It could be shown, that there is a good parallel speedup for core counts up to 16. This means, the parallel behavior is suitable for modern multi-core desktop computers and small clusters. As `hms` aims to be a development framework for prototyping of new numerical model concepts, this result is very satisfying. However, for large-scale applications, larger clusters are necessary and the parallel run-time behavior of `hms` has to be further improved. Here, a parallel implementation using the Message Passing Interface (MPI) standard is most common for distributed parallel computations, which is, due to language restrictions, a very difficult task in the Java programming language (Wikipedia, 2018). A reimplementation of the presented software architecture in a programming language which supports the MPI standard, e. g. C++, would allow distributed memory parallelization and using the computational power of modern graphics processing units (GPU).

7 Summary and conclusions

7.1 Summary

The expected changes in the hydrologic water cycle due to climate change and the demands of the EU water framework and floods directives put more complex requirements on water resources management. Climate change leads to new and rapidly changing conditions and the EU directives require an integrated view on environmental processes. Nowadays, not only balances of water resources and discharge of pollutants are of interest anymore, but also the origin and the spatial and temporal distribution of water or transported matter. Computer simulations and numerical methods have become a more and more important tool for supporting decisions. They simplify the understanding of complex relationships in the hydrological cycle and enable predictions about their future development.

This thesis describes the development of a numerical modeling software which allows the integrated simulation of surface water flow and transport processes as well as their interactions. In the following, the content of the thesis is summarized:

In Chapter 2 the mathematical model concepts for all processes regarded in this thesis were given. All equations were given in the form of a general conservation law. In the description of the runoff generation and sediment transport model, the coupling of the models by source/sink terms was shown.

In Chapter 3 the spatial and temporal discretization of the partial differential equations was shown. Applying Godunov's method the computation of numerical fluxes was regarded as solution to a Riemann problem. The HLLC approximate Riemann solver with hydrostatic reconstruction was presented, which can handle small water depths, varying flow conditions and wetting/drying processes. The spatial accuracy of this scheme was increased to second-order using total variation diminishing methods which guarantee the monotonicity of the solution.

Chapter 4 gave an introduction to the infrastructure and concepts of `hms`. Then, the software architecture of the new numerical framework was described. Three base

components were described: `Engine`, `Solver` and `Scheme`. The `Engine` component encapsulates all numerical methods and handles their parallelization. The `Solver` component is an explicit cell-centered finite volume solver, based on the mathematical concepts presented in Chapter 2 and the numerical methods presented in Chapter 3. The actual solution for the terms of the balance equations for a physical process is encapsulated in the `Scheme` component. A prototype implementation of the conceptual software architecture was presented. Numerical models for surface water flow, tracer and sediment transport were implemented in the framework of `hms` using either the plain FVM method or Godunov's method. In the same concept, the numerical runoff generation model was implemented.

In Chapter 5, the accuracy of the numerics and the implemented code was verified. In five test cases the numerical results were compared to analytical solutions. The test cases involved different flow conditions i. e. sub-, trans- and supercritical flow, discontinuities, very low water depths and wetting/drying processes. In addition, to prove the software concept presented in Chapter 4, the implementation of the mathematical model of sediment transport and morphological evolution was explained.

Finally, in Chapter 6, three applications of the presented software and the implemented numerical models were presented. First, the simulation of a flash flood in a simplified urban district involving strong discontinuities and an initially dry domain was shown. Secondly, the data from an artificial rainfall experiment in Senegal was used for simulation of rainfall-runoff processes including tracer transport and infiltration. In the third application, rainfall-runoff in a small alpine catchment was simulated for a real rainfall event.

7.2 Evaluation and outlook

The objectives of this thesis were explained in Chapter 1. The central goal was to develop a numerical modeling software which allows an integrated view of *surface flow and transport processes* in natural and urban environments. The numerical models should make use of the improvements in the methods to survey *high-resolution* topography information such as LIDAR and photogrammetry to provide highly detailed results. This requires numerical methods which allow a *robust* and highly detailed simulation of surface water flow and transport processes. The developed software should allow *flexible* and easy extension by new processes and numerical methods.

With regard to these objectives, the thesis is evaluated based on the findings from Chapters 5 and 6 and an outlook on further possible developments is given.

To use the developed numerical model for a wide range of applications, it should not be necessary to split surface water flow into separate simulations of rainfall-runoff and channel flow, as it is done in classical hydrological models due to the different complexity of surface runoff and channel flow. The implemented numerical scheme based on the solution of the shallow water equations using the HLLC approximate Riemann solver and the hydrostatic reconstruction has proven to produce stable results for complex flow conditions, e. g. small water depths, wetting/drying and varying flow conditions including sub- and supercritical flows, hydraulic jumps and sharp water level gradients. This is even the case in combination with high-order reconstruction using total variation diminishing methods. The model was verified (Chapter 5) by simulating a hydraulic jump, a dam break, oscillatory flow in a parabolic bowl, and rainfall-runoff in a V-shaped catchment. In particular, the accuracy of the computed flow field was proven in the flash flood and rainfall experiment case studies (Sections 6.1 and 6.2). It can be concluded, that for surface runoff applications, where low water depths can occur, the second-order accurate scheme is superior and allows a detailed representation of the wetting/drying front. In cases with no or minor bottom topography or where water depths are much higher than the variations in bottom topography, the first-order accurate scheme can also provide robust and accurate solutions. Future development could deal with different types of topography discretization techniques, which could improve the results of the first-order accurate scheme for cases with arbitrary bottom topography.

For the simulation of rainfall-runoff in natural areas, an infiltration model based on the Green-Ampt equation and a simple consideration of interflow with a linear reservoir model were implemented. The infiltration model was successfully applied in the rainfall simulation experiment (Section 6.2). It allows a short-term event-based view on hydrological processes. The interflow component was used in the rainfall-runoff simulation at Heumöser (Section 6.3). However, both the Green-Ampt equation and the linear reservoir model are a strong simplification of the flow processes in the vadose zone. In future development, the Green-Ampt equation could be replaced by the one-dimensional Richards equation. Alternatively in order to consider the flow processes in the soil in their entirety, it could be replaced by a two- or three-dimensional numerical model. In addition, for long-term simulations, an implementation of evapotranspiration would be possible.

The shallow water equations were augmented by the transport of contaminants

and sediments. The contaminant and sediment transport models were validated in advection-diffusion simulations (Section 5.5) and dam break over mobile bed simulation (Section 5.6.2), respectively. The results of the tracer transport simulation in the rainfall simulation experiment (Section 6.2) showed good agreement with the measurements and the errors in the mass recovery were within 10 %. In future development, both the contaminant and sediment transport models have to be further investigated to prove their accuracy in surface runoff applications. In addition the extension by other processes, for example chemical reactions could be of interest.

In the case studies it was demonstrated, that the simulations can take advantage of improvements in the methods for surveying high-resolution topography information. In the setup of the rainfall simulation experiment (Section 6.2) and the rainfall-runoff simulation at Heumöser (Section 6.3), high-resolution topography information was used. This allowed a detailed simulation of the flow in small rills and gullies and an accurate reproduction of the flow velocities in the rainfall simulation experiment. A drawback was the amount of mesh cells needed to represent the high-resolution topography information. This limits the range of application in terms of temporal and spatial scales. However, the insights obtained by such simulations can be used to derive improved methods to consider sub-scale effects in large-scale applications. In a research project Özgen et al. developed coarse grid approaches for the numerical shallow water model, which consider the influence of small-scale topography on coarse grids (Özgen et al., 2015; Özgen, Liang and Hinkelmann, 2016; Özgen, Zhao, Liang and Hinkelmann, 2016; Özgen, 2017). The numerical model concepts were successfully implemented based on the software architecture presented in Chapter 4.

When implementing the numerical models for surface flow and transport processes in the framework of `hms`, the concept of the new numerical framework made the development more efficient. The flexible design simplified the implementation with different levels of abstraction. As a proof of concept, the implementation of sediment transport and morphology was given in the model verification (Section 5.6).

The investigation of the parallel run-time behavior showed, that the current implementation is suitable for modern multi-core desktop computers and small clusters (Section 6.3). As `hms` aims to be a development framework for rapid prototyping of new numerical model concepts, this result is very satisfying. For large-scale and long-term applications further improvements are necessary. Possible solutions are the use of graphics processing units (GPU) or distributed memory clusters. Here, a parallel implementation using the Message Passing Interface (MPI) standard is

most common for distributed parallel computations, which is, due to language restrictions, a very difficult task in the Java programming language (Wikipedia, 2018) and a reimplementation in a different language, as C++ is necessary. Overall, a continuous code refactoring by future developers will improve the readability and extendibility of the code.

Since the beginning of the research presented in this thesis, the numerical models implemented in `hms` were successfully used as a tool for analyzing rainfall-runoff and river flow problems in several bachelor and master theses, research projects and doctoral theses (e. g. Özgen, 2011*b*; Adamczak, 2013; Nasser, 2013; Matta et al., 2014; Busse, 2017; Schwettmann, 2016; Özgen, 2017; Tügel et al., 2018). The productive use of `hms` in applications, research and development can be seen as final proof of the suitability of the concept. In future, `hms` framework can be further improved to simplify the practical application for new users. In the moment applications are build up by coding them. This approach is easy for developers as it gives great flexibility as e. g. the possibility to create custom boundary conditions and to automate post-processing steps. For new users, it is difficult to get started. Therefore the possibility to read full case setups from files should be integrated. The foundation was laid with the possibility to write and read settings from and to XML files (cf. Section 4.3.6).

In conclusion, `hms` is heading towards a holistic integrated numerical model for water and environment related problems. `hms` provides the fundamental infrastructure for explicit high-order finite volume schemes, robust numerical methods and a flexible codebase which allows simple extension by new processes and numerical schemes. The framework is therefore a common base for research and development.

Nomenclature

The used notation is described in Section 1.7 on page 11. Page number of first occurence is given.

A	cell area, page 36	$[\mathrm{m}^2]$
C	Chézy coefficient, page 15	$[\mathrm{m}^{1/2}\,\mathrm{s}^{-1}]$
c	depth-averaged concentration, page 28	$[-]$
CCFVM	cell-centered finite volume method, page 34	
CFL	Courant-Friedrichs-Lewy condition, page 37	
Cr	Courant number, page 37	$[-]$
c_w	gravity wave speed, page 20	$[\mathrm{m}\,\mathrm{s}^{-1}]$
D	diffusion coefficient, page 28	$[\mathrm{m}^2\,\mathrm{s}^{-1}]$
D	sediment deposition flux, page 30	$[\mathrm{m}\,\mathrm{s}^{-1}]$
D_t	turbulent diffusion coefficient, page 28	$[\mathrm{m}^2\,\mathrm{s}^{-1}]$
d_G	grain diameter, page 31	$[\mathrm{m}]$
d_0	reference water depth for n_0, page 16	$[\mathrm{m}]$
d	water depth (cf. Figure 2.1), page 14	$[\mathrm{m}]$
E	sediment entrainment flux, page 30	$[\mathrm{m}\,\mathrm{s}^{-1}]$
ε	vegetation drag related coefficient, page 16	$[-]$
$\mathrm{erf}(x)$	error function, page 134	
$\mathbf{F}$	flux vector normal to cell surface, page 35	

$\mathbf{f}$	vector of advective and diffusive fluxes in x direction, page 13	
FOU	first order upwind method, page 39	
Fr	Froude number, page 116	$[-]$
f_x	mass specific external force in x direction, page 15	$[\mathrm{m\,s^{-2}}]$
f_y	mass specific external force in y direction, page 15	$[\mathrm{m\,s^{-2}}]$
$\mathbf{g}$	vector of advective and diffusive fluxes in y direction, page 13	
g	gravity constant, page 14	$[\mathrm{m\,s^{-2}}]$
h	water elevation (cf. Figure 2.1), page 14	$[\mathrm{m}]$
h_f	pressure head at wetting front, page 25	$[\mathrm{m}]$
HLLC	Harten-Lax-van Leer-Contact Riemann solver, page 54	
hms	Hydroinformatics Modelling System, page 6	
h_0	pressure head at soil surface, page 25	$[\mathrm{m}]$
I	cumulative infiltration, page 26	$[\mathrm{m}]$
i	infiltration rate, page 23	$[\mathrm{m\,s^{-1}}]$
K	unsaturated hydraulic conductivity, page 24	$[\mathrm{m\,s^{-1}}]$
κ	Von Kármán constant, page 18	$[-]$
k_s	equivalent sand grain roughness, page 15	$[\mathrm{m}]$
k_Str	Strickler roughness coefficient, page 16	$[\mathrm{m^{1/3}\,s^{-1}}]$
L^1	absolute error, page 116	
L^2	root mean square error, page 129	
λ	eigenvalue, page 20	
λ	Colebrook and White roughness coefficient, page 16	$[-]$
$\mathrm{lb}(x)$	binary logarithm, page 132	
$\lg(x)$	common logarithm, page 15	

l_k	cell edge length, page 35	[m]
$\ln(x)$	natural logarithm, page 26	
m_c	component source/sink term, page 28	$[m\,s^{-1}]$
MC	monotonized central-difference limiter function, page 65	
MUSCL	monotonic upwind-centered scheme for conservation laws, page 62	
m_w	mass source/sink term, page 14	$[m\,s^{-1}]$
$\mathbf{n}$	unit vector normal to cell surface, page 35	
n	Manning roughness coefficient, page 16	$[s\,m^{-1/3}]$
n_0	minimum Manning coefficient below d_0, page 16	$[s\,m^{-1/3}]$
ν	kinematic viscosity, page 15	$[m^2\,s^{-1}]$
ν_{mol}	kinematic molecular viscosity, page 17	$[m^2\,s^{-1}]$
ν_t	kinematic turbulent viscosity, page 17	$[m^2\,s^{-1}]$
OpenMI	Open Modeling Interface, page 83	
$\boldsymbol{\phi}$	vector of primitive unknowns, page 46	
ϕ	bed sediment porositiy, page 30	$[-]$
$\mathbf{\Psi}$	limiter function, page 63	$[-]$
Ψ	runoff-coeffcient, page 172	$[-]$
$\mathbf{q}$	vector of conserved state variables, page 13	
$\mathbf{r}$	eigenvector, page 20	
$\mathbf{r}$	smoothness monitor, page 63	$[-]$
r	total rainfall, page 23	$[m\,s^{-1}]$
RANS	Reynolds-averaged Navier-Stokes equations, page 13	
ρ	density, page 14	$[kg\,m^{-3}]$
$\mathbf{s}$	vector of sources and sinks, page 13	

S	wave speed, page 42	$[\mathrm{ms}^{-1}]$
$\mathrm{Sc_t}$	turbulent Schmidt number, page 27	$[-]$
SWE	shallow water equations, page 14	
$\boldsymbol{\tau}_\mathrm{B}$	bed shear stress, page 15	$[\mathrm{N\,m}^{-2}]$
θ	volumetric water content, page 24	$[-]$
θ_c	critical Shield's parameter, page 31	$[-]$
θ_i	initial water content, page 26	$[-]$
θ_0	water content at soil surface, page 26	$[-]$
TVD	total variation diminishing, page 63	
TV	total variation, page 62	
u	depth-averaged flow velocity in x direction, page 14	$[\mathrm{m\,s}^{-1}]$
UML	Unified Modeling Language, page 10	
u_*	friction velocity, page 18	$[\mathrm{m\,s}^{-1}]$
$\mathbf{v}$	depth-averaged flow velocity, page 15	$[\mathrm{m\,s}^{-1}]$
v	depth-averaged flow velocity in y direction, page 14	$[\mathrm{m\,s}^{-1}]$
V_w	specific soil water volume, page 23	$[\mathrm{m}]$
XML	Extensible Markup Languange, page 107	
z_B	bottom elevation (cf. Figure 2.1), page 14	$[\mathrm{m}]$

Bibliography

Adamczak, C. (2012), *Hochauflösende Strömungs- und Transportsimulation für ein Beregnungs- und Tracerexperiment*, Student research project, Technische Universität Berlin.

Adamczak, C. (2013), *Sensitivitätsanalyse und Parameteridentifikation bei Fragestellungen aus der Hydrosystemmodellierung*, Student research project, Technische Universität Berlin.

Amdahl, G. M. (1967), Validity of the single processor approach to achieving large scale computing capabilities, *in* 'Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (Spring)'.

Audusse, E., Bouchut, F., Bristeau, M.-O., Klein, R. and Perthame, B. (2004), 'A Fast and Stable Well-Balanced Scheme with Hydrostatic Reconstruction for Shallow Water Flows', *SIAM Journal on Scientific Computing* **25**(6), 2050–2065. doi: 10.1137/S1064827503431090.

Audusse, E. and Bristeau, M.-O. (2005), 'A well-balanced positivity preserving "second-order" scheme for shallow water flows on unstructured meshes', *Journal of Computational Physics* **206**(1), 311–333. doi: 10.1016/j.jcp.2004.12.016.

Bathurst, J. C. and O'Connell, P. E. (1992), 'Future of distributed modelling: The Systeme Hydrologique Europeen', *Hydrological Processes* **6**(3), 265–277. doi: 10.1002/hyp.3360060304.

Benkhaldoun, F., Elmahi, I. and Seaïd, M. (2007), 'Well-balanced finite volume schemes for pollutant transport by shallow water equations on unstructured meshes', *Journal of Computational Physics* . doi: 10.1016/j.jcp.2007.04.005.

Berthon, C. and Foucher, F. (2012), 'Efficient well-balanced hydrostatic upwind schemes for shallow-water equations', *Journal of Computational Physics* **231**(15), 4993–5015. doi: 10.1016/j.jcp.2012.02.031.

Beven, K. J. and Kirkby, M. J. (1979), 'A physically based, variable contributing area model of basin hydrology', *Hydrological Sciences Bulletin* **24**(1), 43–69. doi: 10.1080/02626667909491834.

Beven, K. J., Lamb, R., Quinn, P. F., Romanowicz, R. and Freer, J. E. (1995), TOPMODEL, *in* V. P. Singh, ed., 'Computer Models of Watershed Hydrology', Water Resource Publications, Colorado, pp. 627–668.

Borah, D. K. (2011), 'Hydrologic procedures of storm event watershed models: a comprehensive review and comparison', *Hydrological Processes* **25**(22), 3472–3489. doi: 10.1002/hyp.8075.

Box, G. E. P. (1979), Robustness in the strategy of scientific model building, *in* R. L. Launer and G. N. Wilkinson, eds, 'Robustness in statistics', Academic Press, pp. 201–236.

Buffard, T. and Clain, S. (2010), 'Monoslope and multislope MUSCL methods for unstructured meshes', *Journal of Computational Physics* **229**(10), 3745–3776. doi: 10.1016/j.jcp.2010.01.026.

Busse, T. (2017), *HMS - A component-based hydroinformatics modelling system for the engineering of customised and integrated software applications*, Vol. 23 of *Book Series of the Department of Civil Engineering, Technische Universität Berlin*, Doctoral thesis, Shaker Verlag GmbH, Aachen. ISBN 9783844051971.

Busse, T., Molkenthin, F. and Hinkelmann, R. (2007), A software concept of a numerical modelling system for an adaptive simulation of coupled hydrodynamic processes, *in* A. P. Merkel, R. Schütz and T. Wießflecker, eds, 'Proceedings Forum Bauinformatik 2007', Verlag der Technischen Universität Graz, Graz, Austria.

Busse, T., Simons, F., Mieth, S., Hinkelmann, R. and Molkenthin, F. (2012), HMS: A generalised software design to enhance the modelling of geospatial referenced flow and transport phenomena, *in* 'Proceedings of the 10th International Conference on Hydroinformatics', Hamburg, Germany.

Bussing, T. R. A. and Murman, E. M. (1988), 'Finite-volume method for the calculation of compressible chemically reacting flows', *AIAA Journal* **26**(9), 1070–1078. doi: 10.2514/3.10013.

Canestrelli, A., Siviglia, A., Dumbser, M. and Toro, E. F. (2009), 'Well-balanced high-order centred schemes for non-conservative hyperbolic systems. Applications to shallow water equations with fixed and mobile bed', *Advances in Water Resources* **32**(6), 834–844. doi: 10.1016/j.advwatres.2009.02.006.

Cao, Z. and Carling, P. A. (2002), Mathematical modelling of alluvial rivers: reality and myth. Part 2: Special issues, *in* 'Proceedings of the Institution of Civil Engineers-Water Maritime and Engineering', Vol. 154, London: Published for the Institution of Civil Engineers by Thomas Telford Ltd., c2000-c2003., pp. 297–308.

Cao, Z., Pender, G., Wallis, S. and Carling, P. (2004), 'Computational Dam-Break Hydraulics over Erodible Sediment Bed', *Journal of Hydraulic Engineering* **130**(7), 689. doi: 10.1061/(ASCE)0733-9429(2004)130:7(689).

Cea, L., Garrido, M. and Puertas, J. (2010), 'Experimental validation of two-dimensional depth-averaged models for forecasting rainfall-runoff from pre-

cipitation data in urban areas', *Journal of Hydrology* **382**(1-4), 88–102. doi: 10.1016/j.jhydrol.2009.12.020.

Cea, L., Puertas, J. and Vázquez-Cendón, M.-E. (2007), 'Depth Averaged Modelling of Turbulent Shallow Water Flow with Wet-Dry Fronts', *Archives of Computational Methods in Engineering* **14**(3), 303–341. doi: 10.1007/s11831-007-9009-3.

Cheng, N.-S. (1997), 'Simplified Settling Velocity Formula for Sediment Particle', *Journal of Hydraulic Engineering* **123**(2), 149. doi: 10.1061/(ASCE)0733-9429(1997)123:2(149).

Colebrook, C. F. (1939), 'Turbulent flow in pipes, with particular reference to the transition region between the smooth and rough pipe laws', *Journal of the ICE* **11**(4), 133–156. doi: 10.1680/ijoti.1939.13150.

Costabile, P., Costanzo, C. and Macchione, F. (2009), Two-dimensional numerical models for overland flow simulations, *in* 'Proceedings of the River Basin Management 2009 Fifth International Conference on River Basin Management', Vol. 124, Wessex Institute of Technology, UK, Malta, pp. 137–148.
URL: *http://library.witpress.com/viewpaper.asp?pcode=RM09-013-1*

Costabile, P., Costanzo, C. and Macchione, F. (2012), 'Comparative analysis of overland flow models using finite volume schemes', *Journal of Hydroinformatics* **14**(1), 122–135. doi: 10.2166/hydro.2011.077.

Delis, A. I. and Nikolos, I. K. (2013), 'A novel multidimensional solution reconstruction and edge-based limiting procedure for unstructured cell-centered finite volumes with application to shallow water dynamics', *International Journal for Numerical Methods in Fluids* . doi: 10.1002/fld.3674.

Delis, A. I., Nikolos, I. K. and Kazolea, M. (2011), 'Performance and Comparison of Cell-Centered and Node-Centered Unstructured Finite Volume Discretizations for Shallow Water Free Surface Flows', *Archives of Computational Methods in Engineering* . doi: 10.1007/s11831-011-9057-6.

DHI (2018), 'MIKE 21C'.
URL: *https://www.mikepoweredbydhi.com/products/mike-21c* [Accessed 2018-03-05]

Di Giammarco, P., Todini, E. and Lamberti, P. (1996), 'A conservative finite elements approach to overland flow: the control volume finite element formulation', *Journal of Hydrology* **175**(1-4), 267–291. doi: 10.1016/S0022-1694(96)80014-X.

Duggal, K. N. and Soni, J. P. (1996), *Elements Of Water Resources Engineering*, 1 edn, New Age International, New Delhi. ISBN 978-81-224-0807-2.

Elshorbagy, A. and Ormsbee, L. (2006), 'Object-oriented modeling approach to surface water quality management', *Environmental Modelling & Software* **21**(5), 689–698. doi: 10.1016/j.envsoft.2005.02.001.

Emmett, W. W. (1978), Overland flow, *in* M. J. Kirkby, ed., 'Hillslope hydrology', Wiley, pp. 145–176. ISBN 0-471-99510-X.

Esteves, M., Faucher, X., Galle, S. and Vauclin, M. (2000), 'Overland flow and infiltration modelling for small plots during unsteady rain: numerical results versus observed values', *Journal of Hydrology* **228**(3-4), 265–282. doi: 10.1016/S0022-1694(00)00155-4.

European Union (2000), 'Directive 2000/60/EC of the European Parliament and of the Council establishing a framework for the Community action in the field of water policy', *Official Journal* **OJ L 327**.

European Union (2007), 'Directive 2007/60/EC of the European Parliament and of the Council on the assessment and management of flood risks', *Official Journal* **OJ L 288/2**.

Exner, F. M. (1925), 'Über die Wechselwirkung zwischen Wasser und Geschiebe in Flüssen', *Akad. der Wiss in Wien, Math-Naturwissenschafliche Klasse, Sitzungsberichte, Abt IIa* **134**, 165–203.

Fiedler, F. R. and Ramirez, J. A. (2000), 'A numerical method for simulating discontinuous shallow flow over an infiltrating surface', *International Journal for Numerical Methods in Fluids* **32**(2), 219–240.

Freeze, R. A. and Harlan, R. (1969), 'Blueprint for a physically-based, digitally-simulated hydrologic response model', *Journal of Hydrology* **9**(3), 237–258. doi: 10.1016/0022-1694(69)90020-1.

Galland, J.-C., Goutal, N. and Hervouet, J.-M. (1991), 'TELEMAC: A new numerical model for solving shallow water equations', *Advances in Water Resources* **14**(3), 138–148. doi: 10.1016/0309-1708(91)90006-A.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns – Elements of Reusable Object-Oriented Software*. ISBN 9780201633610.

Gessner, M. O., Hinkelmann, R., Nützmann, G., Jekel, M., Singer, G., Lewandowski, J., Nehls, T. and Barjenbruch, M. (2014), 'Urban water interfaces', *Journal of Hydrology* **514**, 226–232.

Godunov, S. K. (1959), 'A Finite Difference Method for the Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics', *Matematicheskii Sbornik* **47**, 357–393.

Gottardi, G. and Venutelli, M. (1993), 'A control-volume finite-element model for two-dimensional overland flow', *Advances in Water Resources* **16**(5), 277–284. doi: 10.1016/0309-1708(93)90019-C.

Green, W. H. and Ampt, G. A. (1911), 'Studies on soil physics, 1, the flow of air and water through soils', *The Journal of Agricultural Science* **4**(1), 1–24. doi: 10.1017/S0021859600001441.

Gregersen, J. B., Gijsbers, P. J. a. and Westen, S. J. P. (2007), 'OpenMI: Open modelling interface', *Journal of Hydroinformatics* **9**(3), 175. doi: 10.2166/hydro.2007.023.

Gualtieri, C., Angeloudis, A., Bombardelli, F., Jha, S., Stoesser, T., Gualtieri, C., Angeloudis, A., Bombardelli, F., Jha, S. and Stoesser, T. (2017), 'On the Values for the Turbulent Schmidt Number in Environmental Flows', *Fluids* **2**(2), 17. doi: 10.3390/fluids2020017.

Harten, A., Lax, P. and van Leer, B. (1983), 'On upstream differencing and Godunov-type schemes for hyperbolic conservation laws', *SIAM review* **25**(1), 35–61.

He, Z., Weng, H., Mao, M. and Ran, Q. (2013), Experimental Study on Soil Erosion and Pollutant Transport During Rainfall-runoff Processes, *in* 'Proceedings of the 35th IAHR World Congress', Tsinghua University Press, Beijing, China, Chengdu, China, p. 12.

He, Z., Wu, W. and Wang, S. S. Y. (2008), 'Coupled Finite-Volume Model for 2D Surface and 3D Subsurface Flows', *Journal of Hydrologic Engineering* **13**(9), 835–845. doi: 10.1061/(ASCE)1084-0699(2008)13:9(835).

Heng, B. C. P., Sander, G. C. and Scott, C. F. (2009), 'Modeling overland flow and soil erosion on nonuniform hillslopes: A finite volume scheme', *Water Resources Research* **45**(5), 1–11. doi: 10.1029/2008WR007502.

Hervouet, J.-M. (2007), *Hydrodynamics of free surface flows: modelling with the finite element method*, John Wiley and Sons. ISBN 9780470035580.

Hillel, D. (1998), *Environmental Soil Physics*, 1 edn, Academic Press, San Diego. ISBN 0-12-348525-8.

Hinkelmann, R. (2005), *Efficient Numerical Methods and Information-Processing Techniques for Modeling Hydro- and Environmental Systems*, Springer-Verlag, Berlin, Heidelberg.

Hinkelmann, R., Liang, Q., Aizinger, V. and Dawson, C. (2015), 'Robust shallow water models', *Environmental Earth Sciences* **74**(11), 7273–7274. doi: 10.1007/s12665-015-4764-1.

Hinkelmann, R. and Zehe, E. (2007), 'Kopplung von Strömungs- und Transportprozessen für die Modellierung von Großhangbewegungen', *Hydrologie und Wasserbewirtschaftung* **1**, 51–54.

Hinkelmann, R. and Zehe, E. (2013), *DFG Forschergruppe 581 "Großhang": Kopplung von Strömungs- und Deformationsprozessen für die Modellierung von Großhangbewegungen*, Final report.

Hirsch, C. (2007), *Numerical computation of internal and external flows: fundamentals of computational fluid dynamics*, Elsevier/Butterworth-Heinemann. ISBN 9780750665940.

HLRN (2016), 'HLRN Supercomputing Service'.
URL: *https://www.hlrn.de/home/view/Service* [Accessed 2016-02-07]

Horton, R. E. (1941), 'An approach toward a physical interpretation of infiltration-capacity', *Soil Science Society of America Journal* **5**(C), 399–417.

Hou, J., Liang, Q., Simons, F. and Hinkelmann, R. (2013), 'A 2D well-balanced shallow flow model for unstructured grids with novel slope source term treatment', *Advances in Water Resources* **52**, 107–131. doi: 10.1016/j.advwatres.2012.08.003.

Hou, J., Simons, F., Liang, Q. and Hinkelmann, R. (2014), 'An improved hydrostatic reconstruction method for shallow water model', *Journal of Hydraulic Research* **52**(3), 432–439. doi: 10.1080/00221686.2013.858648.

Hou, J., Simons, F., Mahgoub, M. and Hinkelmann, R. (2013), 'A robust well-balanced model on unstructured grids for shallow water flows with wetting and drying over complex topography', *Computer Methods in Applied Mechanics and Engineering* **257**, 126–149. doi: 10.1016/j.cma.2013.01.015.

Hydrotec (2018), 'HYDRO_AS-2D'.
URL: *https://www.hydrotec.de/software/hydro-as-2d/* [Accessed 2018-03-05]

Izumi, N. and Parker, G. (2000), 'Linear stability analysis of channel inception: downstream-driven theory', *Journal of Fluid Mechanics* **419**, 239–262.

Jain, M. K., Kothyari, U. C. and Ranga Raju, K. G. (2004), 'A GIS based distributed rainfall-runoff model', *Journal of Hydrology* **299**(1-2), 107–135. doi: 10.1016/j.jhydrol.2004.04.024.

Jiwen, W. and Ruxun, L. (2001), 'The composite finite volume method on unstructured meshes for the two-dimensional shallow water equations', *International Journal for Numerical Methods in Fluids* **37**(8), 933–949. doi: 10.1002/fld.198.

Kampf, S. K. and Burges, S. J. (2007), 'A framework for classifying and comparing distributed hillslope and catchment hydrologic models', *Water Resources Research* **43**(5), 24. doi: 10.1029/2006WR005370.

Kelly, A. (2004), 'The Encapsulate Context Pattern', *ACCU Overload Journal* **63**(63), 1–21 pp.

Kesserwani, G. (2013), 'Topography discretization techniques for Godunov-type shallow water numerical models: a comparative study', *Journal of Hydraulic Research* **51**(4), 351–367. doi: 10.1080/00221686.2013.796574.

Kolditz, O., Delfs, J.-O., Bürger, C., Beinhorn, M. and Park, C.-H. (2008), 'Numerical analysis of coupled hydrosystems based on an object-oriented compartment approach', *Journal of Hydroinformatics* **10**(3), 227–244. doi: 10.2166/hydro.2008.003.

Krámer, T. and Józsa, J. (2007), 'Solution-adaptivity in modelling complex shallow flows', *Computers & Fluids* **36**(3), 562–577. doi: 10.1016/j.compfluid.2006.03.006.

Kutija, V. and Murray, M. G. (2007), 'An object-oriented approach to the modelling of free-surface flows', *Journal of Hydroinformatics* **9**(2), 81–94. doi: 10.2166/hydro.2007.101.

Lacasta, A., Morales-Hernández, M., Murillo, J. and García-Navarro, P. (2015), 'GPU implementation of the 2D shallow water equations for the simulation of rainfall/runoff events', *Environmental Earth Sciences* **74**(11), 7295–7305. doi: 10.1007/s12665-015-4215-z.

Leveque, R. J. (2002), *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, New York.

Liang, Q. (2010), 'Flood Simulation Using a Well-Balanced Shallow Flow Model', *Journal of Hydraulic Engineering* **136**(9), 669–675. doi: 10.1061/(ASCE)HY.1943-7900.0000219.

Liang, Q. (2011), 'A coupled morphodynamic model for applications involving wetting and drying', *Journal of Hydrodynamics* **23**(3), 273–281. doi: 10.1016/S1001-6058(10)60113-8.

Liang, Q. and Borthwick, A. G. L. (2009), 'Adaptive quadtree simulation of shallow flows with wet-dry fronts over complex topography', *Computers & Fluids* **38**(2), 221–234. doi: 10.1016/j.compfluid.2008.02.008.

Liang, Q., Borthwick, A. G. L. and Stelling, G. (2004), 'Simulation of dam- and dyke-break hydrodynamics on dynamically adaptive quadtree grids', *International Journal for Numerical Methods in Fluids* **46**(2), 127–162. doi: 10.1002/fld.748.

Liang, Q. and Marche, F. (2009), 'Numerical resolution of well-balanced shallow water equations with complex source terms', *Advances in Water Resources* **32**(6), 873–884. doi: 10.1016/j.advwatres.2009.02.010.

Liang, Q., Xia, X. and Hou, J. (2015), 'Efficient urban flood simulation using a GPU-accelerated SPH model', *Environmental Earth Sciences* **74**(11), 7285–7294. doi: 10.1007/s12665-015-4753-4.

Lindenmaier, F. (2008), *Hydrology of a large unstable hillslope at Ebnit, Vorarlberg: Identifying dominating processes and structures*, Doctoral thesis, Universität Potsdam.

Luke, E. (2007), On Robust and Accurate Arbitrary Polytope CFD Solvers, *in* 'Proceedings of the 18th AIAA Computational Fluid Dynamics Conference', AIAA, Miami, FL.

MacDonald, I., Baines, M. J., Nichols, N. K. and Samuels, P. G. (1997), 'Analytic Benchmark Solutions for Open-Channel Flows', *Journal of Hydraulic Engineering* **123**(11), 1041–1045. doi: 10.1061/(ASCE)0733-9429(1997)123:11(1041).

Malcherek, A. (2003), *Hydromechanik der Oberflächengewässer*, Technical report, Bundesanstalt für Wasserbau, Außenstelle Küste, Hamburg.

Massoudieh, A. and Kayhanian, M. (2009), An Evolutionary-Based Stepwise Approach for Process Identification and Modeling of Highway Stormwater Contaminant Pollutographs, *in* 'Proceedings of the 33rd IAHR Congress', Vancouver BC, Canada.

Matta, E., Özgen, I., Cabral, J., Candeias, A. L. and Hinkelmann, R. (2014), Simulation of Wind-Induced Flow and Transport in a Brazilian Bay, *in* R. Lehfeldt and R. Kopmann, eds, 'International Conference on Hydroscience & Engineering (ICHE) 2014', Hamburg.

Meyer-Peter, E. and Müller, R. (1948), Formulas for Bed-Load Transport, *in* 'Proceedings of the 2nd Meeting of the International Association of Hydraulic Research', Stockholm, Sweden, pp. 39–64.
URL: *https://www.mendeley.com/catalog/formulas-bedload-transport/*

Mieth, S. (2008), *Kopplung von numerischen Modellierungssystemen unter Verwendung von OpenMI und XML-RPC*, Diploma thesis, Technische Universität Berlin.

Moramarco, T. and Singh, V. P. (2002), 'Accuracy of kinematic wave and diffusion wave for spatial-varying rainfall excess over a plane', *Hydrological Processes* **16**(17), 3419–3435. doi: 10.1002/hyp.1108.

Mügler, C., Planchon, O., Patin, J., Weill, S., Silvera, N., Richard, P. and Mouche, E. (2011), 'Comparison of roughness models to simulate overland flow and tracer transport experiments under simulated rainfall at plot scale', *Journal of Hydrology* **402**(1-2), 25–40. doi: 10.1016/j.jhydrol.2011.02.032.

Murillo, J., García-Navarro, P., Burguete, J. and Brufau, P. (2006), 'A conservative 2D model of inundation flow with solute transport over dry bed', *International Journal for Numerical Methods in Fluids* . doi: 10.1002/fld.1216.

Nasser, S. (2013), *Numerische Simulation der Strömungsprozesse im Reservoir des Luiz Gonzaga Staudammes, Brasilien*, Bachelor thesis, Technische Universität Berlin.

Nehls, T., Menzel, M. and Wessolek, G. (2015), 'Depression storage capacities of different ideal pavements as quantified by a terrestrial laser scanning-based method', *Water Science and Technology* **71**(6), 862–869.

Nikolos, I. and Delis, A. (2009), 'An unstructured node-centered finite volume scheme for shallow water flows with wet/dry fronts over complex topography',

Computer Methods in Applied Mechanics and Engineering **198**(47-48), 3723–3750. doi: 10.1016/j.cma.2009.08.006.

Nikuradse, J. (1933), *Strömungsgesetze in rauhen Rohren*, VDI-Forschungsheft, VDI-Verlag, Berlin.

Nujic, M. (1998), *Praktischer Einsatz eines hochgenauen Verfahrens für die Berechnung von tiefengemittelten Strömungen*, Mitteilungen, Institut für Wasserwesen. Universität der Bundeswehr, München.

OpenMI Association (2018), 'OpenMI'.
URL: *https://www.openmi.org* [Accessed 2018-09-23]

Oracle (2018), 'Java SE Development Kit'.
URL: *https://www.oracle.com/technetwork/java/javase/* [Accessed 2018-09-23]

Overton, D. and Brakensiek, D. (1970), A kinematic model of surface runoff response, *in* 'Proceedings of the Wellington Symposium', Unesco/IAHS, Paris, pp. 100–112.

Özgen, I. (2011*a*), *An HLLC Riemann solver based second order scheme for the shallow water equations*, Diploma thesis. Technische Universität Berlin.

Özgen, I. (2011*b*), *Simulation of natural surface runoff in consideration of infiltration in the unsaturated zone*, Student research project, Technische Universität Berlin.

Özgen, I. (2017), *Coarse Grid Approaches for the Shallow Water Model*, Doctoral thesis, Technische Universität Berlin. doi: 10.14279/depositonce-6269.

Özgen, I., Liang, D. and Hinkelmann, R. (2016), 'Shallow water equations with depth-dependent anisotropic porosity for subgrid-scale topography', *Applied Mathematical Modelling* **40**(17-18), 7447–7473. doi: 10.1016/J.APM.2015.12.012.

Özgen, I., Teuber, K., Simons, F., Liang, D. and Hinkelmann, R. (2015), 'Upscaling the shallow water model with a novel roughness formulation', *Environmental Earth Sciences* **74**(11), 7371–7386. doi: 10.1007/s12665-015-4726-7.

Özgen, I., Zhao, J., Liang, D. and Hinkelmann, R. (2016), 'Urban flood modeling using shallow water equations with depth-dependent anisotropic porosity', *Journal of Hydrology* **541**, 1165–1184. doi: 10.1016/j.jhydrol.2016.08.025.

Parlange, J.-Y., Haverkamp, R. and Touma, J. (1985), 'Infiltration under ponded conditions. I: Optimal analytical solution and comparison with experimental observations', *Soil science* **139**(4), 305–311.

Pavan, S., Ata, R. and Hervouet, J. M. (2015), 'Finite volume schemes and residual distribution schemes for pollutant transport on unstructured grids', *Environmental Earth Sciences* **74**(11), 7337–7356. doi: 10.1007/s12665-015-4760-5.

Philip, J. R. (1957), 'The Theory of Infiltration: 1. The Infiltration Equation and its Solution', *Soil Science* **83**(5), 345–358.

Ponce, V., Li, R. and Simons, D. (1978), 'Applicability of kinematic and diffusion models', *Journal of the Hydraulics Division* **104**(3), 353–360.

Prandtl, L. (1925), 'Bericht über Untersuchungen zur ausgebildeten Turbulenz', *Zeitschrift fur angewandte Mathematik und Mechanik* **5**(1925), 136–139.

Radcliffe, D. E. and Simunek, J. (2010), *Soil Physics with Hydrus: Modeling and Applications*, Crc Pr Inc. ISBN 142007380X.

Refsgaard, J. C. and Storm, B. (1995), MIKE SHE, *in* V. P. Singh, ed., 'Computer Models of Watershed Hydrology', Water Resources Publications, Colorado, USA, pp. 806–846. ISBN 0918334918.

Richards, L. A. (1931), 'Capillary conduction of liquids through porous mediums', *Journal of Applied Physics* **1**(5), 318–333.

Roe, P. (1985), 'Some contributions to the modelling of discontinuous flows', *Lectures Notes in Applied Mathematics* **22**, 163–193.

Sampson, J., Easton, A. and Singh, M. (2006), Moving boundary shallow water flow above parabolic bottom topography, *in* A. Stacey, B. Blyth, J. Shepherd and A. J. Roberts, eds, 'Proceedings of the 7th Biennial Engineering Mathematics and Applications Conference, EMAC-2005', Vol. 47 of *ANZIAM J.*, pp. C373–C387.

Sanders, B. F., Schubert, J. E. and Gallegos, H. A. (2008), 'Integral formulation of shallow-water equations with anisotropic porosity for urban flood modeling', *Journal of Hydrology* **362**(1-2), 19–38. doi: 10.1016/j.jhydrol.2008.08.009.

Schwettmann, K. (2016), *Hochauflösende Strömungssimulation für ein Dammbruch- und ein Beregnungsexperiment*, Bachelor thesis, Technische Universität Berlin.

Shen, C. and Phanikumar, M. S. (2010), 'A process-based, distributed hydrologic model based on a large-scale method for surface-subsurface coupling', *Advances in Water Resources* **33**(12), 1524–1541. doi: 10.1016/j.advwatres.2010.09.002.

Sherman, L. K. (1932), 'Streamflow from rainfall by the unit graph method', *Engineering News Record* **108**, 501–505.

Simons, F. (2008), *Weiterentwicklung einer Finite-Volumen-Methode für die Flachwassergleichungen zur Simulation des Oberflächenabflusses im Gewässer und im Gelände*, Diploma thesis, Technische Universität Berlin.

Simons, F., Busse, T., Hou, J., Notay, K. V. and Hinkelmann, R. (2011), A Robust and Efficient Solver for the Shallow Water Equations and its Application to a Complex Natural Hydrosystem, *in* 'Proceedings of the 34th IAHR Congress', Engineers Australia, Brisbane, Australia, pp. 4276–4283.

Simons, F., Busse, T., Hou, J., Özgen, I. and Hinkelmann, R. (2012), HMS: Model Concepts and Numerical Methods around Shallow Water Flow within an Extendable Modeling Framework, *in* 'Proceedings of the 10th International Conference on Hydroinformatics', Hamburg, Germany.

Simons, F., Busse, T., Hou, J., Özgen, I. and Hinkelmann, R. (2014), 'A model for overland flow and associated processes within the Hydroinformatics Modelling System', *Journal of Hydroinformatics* **16**(2), 375–391. doi: 10.2166/hydro.2013.173.

Simpson, G. and Castelltort, S. (2006), 'Coupled model of surface water flow, sediment transport and morphological evolution', *Computers & Geosciences* **32**(10), 1600–1614. doi: 10.1016/j.cageo.2006.02.020.

Singh, V. P. and Woolhiser, D. a. (2002), 'Mathematical Modeling of Watershed Hydrology', *Journal of Hydrologic Engineering* **7**(4), 270–292. doi: 10.1061/(ASCE)1084-0699(2002)7:4(270).

Smith, R. E. and Woolhiser, D. A. (1971), 'Overland Flow on an Infiltrating Surface', *Water Resources Research* **7**(4), 899–913. doi: 10.1029/WR007i004p00899.

Soares-Frazão, S., Lhomme, J., Guinot, V. and Zech, Y. (2008), 'Two-dimensional shallow-water model with porosity for urban flood modelling', *Journal of Hydraulic Research* **46**(1), 45–64. doi: 10.1080/00221686.2008.9521842.

Song, L., Zhou, J., Guo, J., Zou, Q. and Liu, Y. (2011), 'A robust well-balanced finite volume model for shallow water flows with wetting and drying over irregular terrain', *Advances in Water Resources* **34**(7), 915–932. doi: 10.1016/j.advwatres.2011.04.017.

Stadler, L. (2016), *Entwicklung von Modellkonzepten für die Simulation von Zweiphasenströmungen in makroporösen Böden*, Doctoral thesis, Technische Universtität Berlin. doi: 10.14279/depositonce-4972.

Stadler, L., Hinkelmann, R. and Helmig, R. (2012), 'Modeling Macroporous Soils with a Two-Phase Dual-Permeability Model', *Transport in Porous Media* **95**(3), 585–601. doi: 10.1007/s11242-012-0064-3.

Stelling, G. S. and Duinmeijer, S. P. A. (2003), 'A staggered conservative scheme for every Froude number in rapidly varied shallow water flows', *International Journal for Numerical Methods in Fluids* **43**(12), 1329–1354. doi: 10.1002/fld.537.

Stephenson, D. and Meadows, M. E. (1986), *Kinematic Hydrology and Modelling*, Elsevier, Amsterdam. ISBN 0444426167.

Sweby, P. K. (1984), 'High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws', *SIAM Journal on Numerical Analysis* . doi: 10.1137/0721062.

Tamás, K. (2006), Solution-adaptive 2D modelling of wind-induced lake circulation, PhD thesis.

Tatard, L., Planchon, O., Wainwright, J., Nord, G., Favismortlock, D., Silvera, N., Ribolzi, O., Esteves, M. and Huang, C. (2008), 'Measurement and modelling of high-resolution flow-velocity data under simulated rainfall on a low-slope sandy soil', *Journal of Hydrology* **348**(1-2), 1–12. doi: 10.1016/j.jhydrol.2007.07.016.

Tayfur, G., Kavvas, M. L., Govindaraju, R. S. and Storm, D. E. (1993), 'Applicability of St. Venant Equations for Two-Dimensional Overland Flows over Rough Infiltrating Surfaces', *Journal of Hydraulic Engineering* **119**(1), 51–63. doi: 10.1061/(ASCE)0733-9429(1993)119:1(51).

Testa, G., Zuccalà, D., Alcrudo, F., Mulet, J. and Soares-Frazão, S. (2007), 'Flash flood flow experiment in a simplified urban district', *Journal of Hydraulic Research* **45**(Extra Issue), 37–44.

Thacker, W. C. (1981), 'Some exact solutions to the nonlinear shallow-water wave equations', *J. Fluid Mech* **107**, 499–508. doi: 10.1017/S0022112081001882.

Toro, E. F. (1992), 'Riemann problems and the WAF method for solving two-dimensional shallow water equations', *Philosophical Transactions: Physical Sciences and Engineering* **338**(1649), 43–68.

Toro, E. F. (2001), *Shock-Capturing Methods for Free-Surface Shallow Flows*, 1. edn, John Wiley & Sons. ISBN 0471987662.

Toro, E. F. and Garcia-Navarro, P. (2007), 'Godunov-type methods for free-surface shallow flows: A review', *Journal of Hydraulic Research* **45**(6), 736–751. doi: 10.1080/00221686.2007.9521812.

Toro, E. F., Spruce, M. and Speares, W. (1994), 'Restoration of the contact surface in the HLL-Riemann solver', *Shock Waves* **4**(1), 25–34. doi: 10.1007/BF01414629.

Tsai, C. W. (2003), 'Applicability of Kinematic, Noninertia, and Quasi-Steady Dynamic Wave Models to Unsteady Flow Routing', *Journal of Hydraulic Engineering* **129**(8), 613–627. doi: 10.1061/(ASCE)0733-9429(2003)129:8(613).

TU Berlin (2017), 'Hydroinformatics Modeling System'.
URL: *http://www.wahyd.tu-berlin.de/menue/research/research_projects/modellentwicklung/hms/parameter/en/* [Accessed 2017-01-05]

Tügel, F., Özgen, I., Hinkelmann, R., Hadidi, A. and Tröger, U. (2018), Modeling of Flash Floods in Wadi Systems Using a Robust Shallow Water Model–Case Study El Gouna, Egypt, *in* 'Advances in Hydroinformatics', Springer, Singapore, pp. 579–593.
URL: *http://link.springer.com/10.1007/978-981-10-7218-5_41*

UMR LISAH (2004), 'Data Thies'.
URL: *https://www.umr-lisah.fr/Thies_2004/* [Accessed 2017-05-27]

U.S. Army Corps of Engineers (2000), *Hydrologic Modeling System HEC-HMS Technical Reference Manual*, Hydrologic Engineering Center, Davis, CA.

van Leer, B. (1973), 'Towards the ultimate conservative difference scheme I. The quest of monotonicity', *Springer Lecture Notes Phys.* **18**, 163—-168.

van Leer, B. (1974), 'Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second order scheme', *Journal of Computational Physics* **14**(4), 361–370.

van Leer, B. (1977*a*), 'Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow', *Journal of Computational Physics* **23**(3), 263–275.

van Leer, B. (1977*b*), 'Towards the ultimate conservative difference scheme IV. A new approach to numerical convection', *Journal of Computational Physics* **23**, 276—-299. doi: 10.1016/0021-9991(77)90095-X.

van Leer, B. (1979), 'Towards the ultimate conservative difference scheme V. A second order sequel to Godunov's method', *Journal of Computational Physics* **32**, 101–136.

VanderKwaak, J. E. and Loague, K. (2001), 'Hydrologic-Response simulations for the R-5 catchment with a comprehensive physics-based model', *Water Resources Research* **37**(4), 999–1013. doi: 10.1029/2000WR900272.

Vanzo, D., Siviglia, A. and Toro, E. F. (2016), 'Pollutant transport by shallow water equations on unstructured meshes: Hyperbolization of the model and numerical solution via a novel flux splitting scheme', *Journal of Computational Physics* **321**. doi: 10.1016/j.jcp.2016.05.023.

Wang, J., Endreny, T. A. and Hassett, J. M. (2005), 'A flexible modeling package for topographically based watershed hydrology', *Journal of Hydrology* **314**(1-4), 78–91. doi: 10.1016/j.jhydrol.2005.03.030.

Wang, Y., Liang, Q., Kesserwani, G. and Hall, J. W. (2011), 'A 2D shallow flow model for practical dam-break simulations', *Journal of Hydraulic Research* **49**(3), 307–316. doi: 10.1080/00221686.2011.566248.

Weill, S. (2007), Modélisation des échanges surface/subsurface à l'échelle de la parcelle par une approche darcéenne multidomaine, PhD thesis, École des Mines de Paris.
URL: *http://bib.rilk.com/3411/*

Weill, S., Mazzia, A., Putti, M. and Paniconi, C. (2011), 'Coupling water flow and solute transport into a physically-based surface-subsurface hydrological model', *Advances in Water Resources* **34**(1), 128–136. doi: 10.1016/j.advwatres.2010.10.001.

Weill, S., Mouche, E. and Patin, J. (2009), 'A generalized Richards equation for surface/subsurface flow modelling', *Journal of Hydrology* **366**(1-4), 9–20. doi: 10.1016/j.jhydrol.2008.12.007.

Wienhöfer, J., Germer, K., Lindenmaier, F., Färber, A. and Zehe, E. (2009), 'Applied tracers for the observation of subsurface stormflow at the hillslope scale', *Hydrol. Earth Syst. Sci.* **13**(7), 1145–1161.

Wikipedia (2017*a*), 'Java (programming language)'.
URL: *https://en.wikipedia.org/wiki/Java_(programming_language)* [Accessed 2017-10-31]

Wikipedia (2017*b*), 'Unified Modeling Language'.
URL: *https://en.wikipedia.org/wiki/Unified_Modeling_Language* [Accessed 2017-11-19]

Wikipedia (2018), 'Message Passing Interface'.
URL: *https://en.wikipedia.org/wiki/Message_Passing_Interface* [Accessed 2018-04-15]

Wu, W. (2007), *Computational river dynamics*, CRC Press.

Yeh, G.-T., Shih, D.-S. and Cheng, J.-R. C. (2011), 'An integrated media, integrated processes watershed model', *Computers & Fluids* **45**(1), 2–13. doi: 10.1016/j.compfluid.2010.11.018.

Yoon, T. H. and Kang, S.-K. (2004), 'Finite Volume Model for Two-Dimensional Shallow Water Flows on Unstructured Grids', *Journal of Hydraulic Engineering* **130**(7), 678–688. doi: 10.1061/(ASCE)0733-9429(2004)130:7(678).

Zhang, W. and Cundy, T. W. (1989), 'Modeling of two-dimensional overland flow', *Water Resources Research* **25**(9), 2019–2035. doi: 10.1029/WR025i009p02019.

Zhao, J., Özgen, I., Liang, D. and Hinkelmann, R. (2017), 'Comparison of depth-averaged concentration and bed load flux sediment transport models of dam-break flow', *Water Science and Engineering* **10**(4), 287–294. doi: 10.1016/j.wse.2017.12.006.

Zhou, J. G., Causon, D. M., Mingham, C. G. and Ingram, D. M. (2001), 'The Surface Gradient Method for the Treatment of Source Terms in the Shallow-Water Equations', *Journal of Computational Physics* **168**(1), 1–25. doi: 10.1006/jcph.2000.6670.