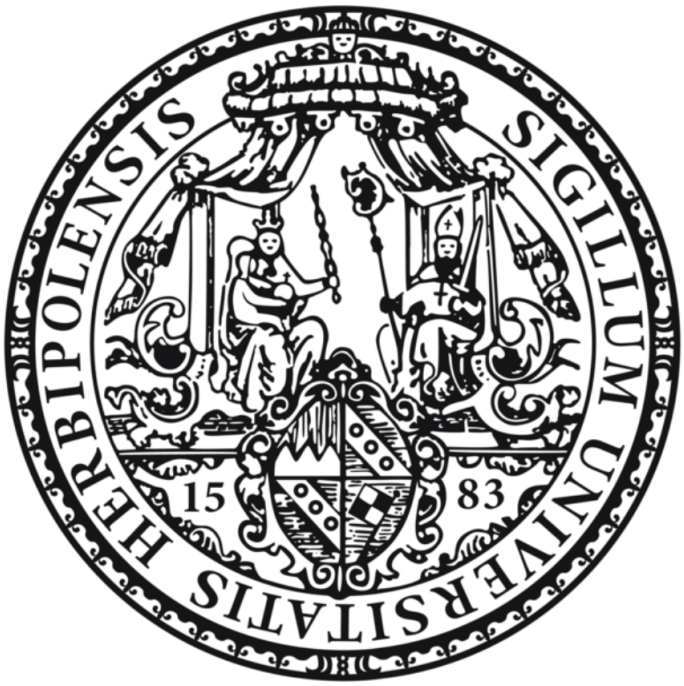
Julius-Maximilians-Universität Würzburg

Fakultät für Mathematik und Informatik

Institut für Mathematik

Simulating Water Waves in Networks Using a High-Order Accurate Entropy Stable Finite Volume Scheme



Masterthesis in Computational Mathematics

Veronika Mayerhofer

Supervisor:
Prof. Dr. Christian Klingenberg

Würzburg, Oktober 2023

Abstract

Mathematical modeling can be performed by solving a suitable system of partial differential equations. Since there is often no analytical solution available, numerical methods can be used to approximate the solution. This work focuses on modeling water waves in a network, i.e. a collection of straight channels connected by junctions. To compute solutions in this setting, the most straightforward treatment consists of employing a two-dimensional numerical solver. However, since this is computationally very expensive, we focus on exploiting the specific structure of this problem to ease the computation. The nature of this setting suggests to use a one-dimensional solver in each channel and update the states across the junction accordingly. The underlying hyperbolic system consists of the *shallow water equations*, a set of nonlinear conservation laws that can be used to model certain types of water waves. We solve them using the *finite volume method* which approximates the average value of the conserved quantities in each grid cell in the computational domain.

To improve the quality of the numerical solution, we use the *TeCNO scheme* developed by Fjordholm et al. [1] in the channels. It combines entropy conservative fluxes with diffusion operators using *ENO reconstructions* to obtain a high-order accurate, entropy stable method. At the junctions, we update the states by solving a nonlinear system of equations derived by Briani et al. [2]. Hence, in summary, we use the one-dimensional TeCNO scheme in the channels and the junction solver across the junction instead of applying a two-dimensional solver. Our numerical experiments verify the effectiveness of this method.

In the last part of this thesis, we account for uncertainties in the initial data which commonly appear in practical applications. We apply the *stochastic collocation method* to different problems on intervals and networks with one- or two-dimensional random variables in order to assess the impact of uncertain initial data on the solution.

Zusammenfassung

Mathematische Modellierung erfolgt häufig durch das Lösen partieller Differentialgleichungen. Da in den meisten Fällen keine analytische Lösung exitiert, werden numerische Methoden verwendet um die Lösung zu approximieren. Diese Arbeit beschäftigt sich mit der Modellierung von Wasser in Netzwerken, d.h. gerader Kanäle die durch Kreuzungen miteinander verbunden sind. Hierbei ist die naheliegendste Herangehensweise die Verwendung eines zweidimensionalen numerischen Lösers. Da dies in der Praxis jedoch sehr rechenintensiv ist, nutzen wir in dieser Arbeit die spezielle Struktur des Problems um die Berechnungen zu vereinfachen. Die zugrundeliegende Geometrie legt nahe, in den Kanälen einen eindimensionalen Löser zu verwenden und die Zustände über die Kreuzung hinweg zu aktualisieren. Das zugrundeliegende hyperbolische System sind die *Flachwassergleichungen*, nichtlineare Erhaltungsgleichungen die zur Simulation bestimmter Arten von Wasserwellen verwendet werden können. Diese werden mithilfe des *Finite-Volumen-Verfahrens* gelöst, welches den Durchschnittswert der Erhaltungsgröße in jeder Zelle des Berechnungsgebiets approximiert.
Um die Qualität der numerischen Berechnungen zu verbessern, verwenden wir das von Fjordholm et al. entwickelte *TeCNO Verfahren* [1] in den Kanälen. Es verknüpft entropie-erhaltende Flüsse mit Diffusionsoperatoren, welche auf *ENO Rekonstruktionsverfahren* beruhen um entropie-stabile Verfahren höherer Ordnung zu erhalten. An den Kreuzungen zwischen den Kanälen aktualisieren wir die Zustände durch das Lösen eines nicht-linearen Systems basierend auf der Arbeit von Briani et al. [2]. Die numerischen Experimente in dieser Arbeit belegen die Effektivität dieser Methode.
Im letzten Teil dieser Arbeit widmen wir uns Ungenauigkeiten in den Ausgangsdaten, wie sie in praktischen Anwendungen häufig auftreten. Hierbei wenden wir die *stochastische Kollokationsmethode* auf verschiedene Probleme auf Intervalen und Netzwerken unter der Verwendung ein-oder zweidimensionaler Zufallsvariablen an, um die Auswirkung ungenauer initialer Daten auf die Lösung zu betrachten.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Christian Klingenberg for his support and supervision during my writing process. I really enjoyed my time in his work group and I am grateful for the work environment he provided for his students.
Secondly, I would like to thank Ulrik Fjordholm for his support and professional advice during my time in Oslo and for giving me the opportunity to spend these incredible months abroad to work on my thesis.
Finally, I would like to thank my family and friends for all their (moral and financial) support during my studies.

Contents

1 Introduction

Conservation laws play a crucial role when modeling natural phenomena mathematically. They can be used in a variety of different situations to simulate the behavior of conserved quantities. For instance, the *shallow water equations* can be used to model certain kinds of water waves where the wavelength is of at least the same magnitude as the water depth with a relatively small wave height. Examples of such waves are tsunamis or waves propagating in shallow water. The equations enable us to predict wave heights as tsunamis approach the shore and can hence play an important role in hazard prevention and constructing appropriate precautions in endangered areas [3, 4]. Another application of conservation laws in real-world scenarios is modeling traffic flow, i.e. the interaction between travelers such as cars or pedestrians and infrastructure which can be taken into consideration for city planing. There are different possibilities to choose the governing equations, depending on external circumstances such as the density of cars [5]. However, even particular equations can be used in a variety of different applications that do not seem to have much in common at first glance. The *compressible Euler system* is used to model density, velocity and pressure of inviscid compressible fluids. For industrial purposes, they can be applied to model fluid flow around turbines, vehicles or airplane wings. By adding a gravitational source term, they can also be used in atmospheric physics and astrophysics to model stellar structures [6].
However, even though solutions to these equations can be used to model physical phenomena, in most cases there is no analytical solution available. Hence, in order to use the theoretical findings, we require numerical methods to approximate the exact solution to a certain degree of accuracy. Even though the underlying partial differential equations may look very different and can be applied in diverse contexts, they can be solved numerically in the same manner. Two popular methods divide the computational domain into grid cells and try to approximate either the average value or the function value at a particular point in each cell which results in *finite volume methods* or *finite difference methods*, respectively. Due to conservation, the quantities can neither be created nor destroyed in the process. Therefore, the respective values only change due to fluxes through the boundaries of the cells in each time step. When using finite volume or finite difference methods, it is crucial to correctly define the numerical flux functions as they have a significant impact on the quality of the resulting solutions.
What makes the numerical computation even more complex is that some real-world applications cannot be modeled by a one-dimensional problem setting. When we model water waves in a network, i.e. a collection of straight channels separated by junctions, one usually needs a two-dimensional solver. However, this is computationally very expensive and difficult to implement. Hence, we exploit the specific structure of this setting and consider the solution throughout channels and across the junction as two distinct, but coupled problems. Since we fix the width for each channel, they can each be interpreted as one-dimensional problems. The states across the junction are then updated by considering the solutions to a nonlinear system in the numerical fluxes between the cells. This reduces the computational complexity from a two-dimensional problem to a one-dimensional one coupled with a nonlinear system. This problem setting can be applied in various scenarios when one seeks to model water in networks, e.g. in the construction of harbours or sewers. We find, for instance, that the angle of the outgoing channels has an impact on the water height as a wave passes through the junction which is important to consider in real-world networks.
In addition to the general need for numerical methods to compute solutions to conservation laws due to the lack of analytical solutions and the diverse complexity of the circumstances, there arises a third problem in mathematical simulations. To model the evolution of a conserved quantity, we require an

initial state that is passed on to the numerical method. In real-world scenarios, however, the initial data is generally affected by measurement errors, so we have to account for a certain level of inaccuracy. By applying stochastic methods, we attempt to derive information regarding the influence of these errors on the solution. The simplest method for treating stochastic equations is the *Monte Carlo method* which computes realizations of the deterministic solver at random points in the random parameter space. Even though the implementation is straightforward once the underlying deterministic solver is available, the convergence rate is too slow to make the method applicable in practice. Hence, we decided to employ *stochastic collocation methods* instead. Similar to the Monte Carlo method, it uses the fact that fixing the random parameter leads to a deterministic problem that can be solved by a deterministic solver. However, instead of using random points, it works with predetermined collocation points and interpolates in the random variable space afterwards. The main difficulty is the choice of the collocation points in multi-dimensional random spaces. We will show that tensor products of one-dimensional quadrature points are not suitable, since these sets grow quickly with increasing dimensions. Instead, we derive a sparse grid using the *Smolyak algorithm*.

In this thesis, we use finite volume methods to solve the partial differential equations numerically. We equip them with the *TeCNO scheme* developed by Fjordholm et al. [1] to compute the numerical flux functions between cells. We show that this method is high-order accurate and entropy stable. The first seven chapters are dedicated to deriving the theoretical background necessary to define the scheme.

In the first chapter, we derive hyperbolic conservation laws and discuss some important examples that will frequently reappear throughout this thesis. Since we mainly focus on the shallow water system, this set of equations is considered in detail. Furthermore, we consider exact solutions to Riemann problems for this system in Chapter 3. They play an important role in the numerical examples since they enable us to compute an exact solution that serves as a reference solution. Furthermore, they are used in the construction of the nonlinear junction solver. In Chapter 4, we formally introduce finite volume methods. These first three introductory chapters are mainly based on [1, 5, 7–9].

Afterwards, we start investigating the construction of first-order entropy conservative and entropy stable methods for scalar equations as well as systems based on [1, 7]. The main idea to obtain entropy stable fluxes is to combine an entropy conservative flux with a diffusion term depending on entropy variables. However, since the TeCNO scheme is supposed to be high-order accurate, we need to extend them to higher orders. For this purpose, we introduce general reconstruction procedures in Chapter 6 based on [6]. In Chapter 7, we first construct high-order entropy conservative fluxes by using linear combinations of second-order accurate ones. However, we will see that the quality of the numerical solution in the vicinity of shocks is highly dependent on the diffusion term. We can smooth out spurious oscillations by using specifically structured diffusion matrices combined with reconstructed entropy variables. The reconstruction method needs to fulfill the sign property, i.e. the jump in the reconstructed values must have the same sign as the jump in the original values. For this purpose, we use the *ENO reconstruction* investigated in Chapter 8. Afterwards, we are finally able to formally introduce the TeCNO scheme in Chapter 9. The last three chapters are all based on [1, 7]. In Chapter 10, we present numerical examples verifying the formal rate of convergence and the non-oscillatory property of the method. Up to this point, however, only considered the TeCNO scheme on intervals. Hence, the following chapters are dedicated to modeling water waves in networks.

In Chapter 11, we formally define networks and introduce the required notation. Following the procedure in [2], we compute solutions in the channels with a one-dimensional solver and handle the junction separately. To this end, we derive a nonlinear system that has to be solved in order to update the cell

states across the junction in the finite volume method. In this system, three equations arise from Riemann problems across the interfaces of the junction while three more follow from conservation of mass and momentum in the "junction triangle". However, since there are no general existence and uniqueness results for nonlinear systems available, we can only guarantee a solution under specific circumstances. In the numerical experiments in Chapter 12, we consider a variety of different channel angles and initial data to investigate their influence on the solution throughout the network.

In the last part of this work, we account for certain inaccuracies in the initial data and discuss two stochastic methods to solve stochastic partial differential equations. First, we introduce the Monte Carlo method and discuss its advantages and drawbacks. Afterwards, we consider the stochastic collocation method that applies the deterministic solver at predetermined, data-independent points and uses Lagrange interpolation to derive a function that can be evaluated at all points in the random parameter space. To reduce the number of collocation points, we use the Smolyak algorithm that returns a sparse grid and therefore significantly lowers the computational complexity. This last part of the thesis is based on [10, 11]. Finally, we demonstrate the stochastic collocation method for one- and two-dimensional random variables and visualize the resulting sample mean and standard deviation.

Notation

In this section, we introduce the notation that is used throughout this thesis.
We denote the set of non-negative real numbers by $\mathbb{R}^+ := [0, \infty)$. The canonical inner product on $\mathbb{R}^n$ is $x \cdot y = x^T y = \sum_{k=1}^n x_k y_k$ with the associated norm $|x| = \sqrt{x \cdot x}$.
The characteristic function of a set U is denoted by

$$\mathbb{1}_U(x) = \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{if } x \notin U \end{cases} . \tag{1.1}$$

We denote the set of k–times differentiable functions $f : \mathbb{R}^m \to \mathbb{R}^n$ by $C^k(\mathbb{R}^m, \mathbb{R}^n)$. Furthermore, the space of smooth functions with compact support $U \subset \mathbb{R}^m$ is denoted by $\mathcal{D}(U) := C_c^\infty(U, \mathbb{R})$. Its dual space $\mathcal{D}'(U)$ is the space of distributions on U.
Partial derivatives of a function are denoted by subscripts, i.e. $f_x = \frac{\partial f}{\partial x}$ and the Jacobian of a function $f \in C^1(\mathbb{R}^m, \mathbb{R}^n)$ by $f'(x) \in \mathbb{R}^{n \times m}$.
For piecewise constant functions on a grid with grid cells $\mathcal{C}_i$, i.e. $u(x) = \sum_{i \in \mathbb{Z}} u_i \mathbb{1}_{\mathcal{C}_i}(x)$, we denote the jump and the average value across the interface at $x_{i+1/2}$ by $[\![u]\!]_{i+1/2} := u_{i+1} - u_i$ and $\overline{u}_{i+1/2} := \frac{u_{i+1}+u_i}{2}$, respectively. Note that the identity

$$[\![ab]\!]_{i+1/2} = \overline{a}_{i+1/2}[\![b]\!]_{i+1/2} + [\![a]\!]_{i+1/2}\overline{b}_{i+1/2} \tag{1.2}$$

holds across all interfaces.
Furthermore, for a reconstruction method $v_i(x) = \mathcal{R}_i(\{v_j\}_{j \in \mathbb{Z}})$, we denote the jump at the interface at $x_{i+1/2}$ by $\langle\!\langle v \rangle\!\rangle_{i+1/2} := v_{i+1}^- - v_i^+ = v_{i+1}(x_{i+1/2}) - v_i(x_{i+1/2})$.
We call a function $f \in C^2(\mathbb{R}^n, \mathbb{R})$ *convex*, if $f''(x) \geq 0,\ \forall x \in \mathbb{R}^n$ and *strictly convex* if $f''(x) > 0,\ \forall x \in \mathbb{R}^n$.
We denote the identity matrix in $\mathbb{R}^{n \times n}$ by I_n.
For two symmetric matrices $A, B \in \mathbb{R}^{n \times n}$, we say $A \leq B$ if $B - A$ is a positive semi-definite matrix and $A < B$ if $B - A$ is a positive definite matrix. We call a matrix *positive* or *non-negative*, if $0 < A$ or $0 \leq A$, respectively.
For two functions $f, g : \mathbb{R} \to \mathbb{R}$, we say $f \in O(g)$ if there exists a constant $c > 0$ and $x_0 \in \mathbb{R}$ such that

$$|f(x)| \leq cg(x) \quad \forall x \geq x_0. \tag{1.3}$$

We denote the *dirac delta function* by

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & \text{else} \end{cases} . \tag{1.4}$$

Code

This work is accompanied by Python code that can be used to reproduce the numerical examples in Chapter 10, 12 and 13. This chapter gives a brief introduction to the structure of the code and how it can be used.

The implementation itself is contained in the folder *numerics*. There, it is split into different subfolders. The folder *basics* contains all files that compute basic functions necessary for the remaining implementations. For instance, it incorporates the code that computes the eigenvalues, eigenvectors or flux functions of the conservation laws of interest. Furthermore, the file *compute.py* contains basic computations such as *cfl_condition()*, *diffusion_scalar()* or *diffusion_system()*.

The folder *solver* contains different possibilities to determine a solution to a given conservation law. The implementations of exact solutions for different equations can be found in *exact_solution.py*. Furthermore, in the finite volume method implemented in *finite_volume_method.py*, one can choose between different numerical solvers such as the TeCNO scheme (*tecno.py*), Lax-Friedrichs scheme (*lax_friedrichs.py*) or Godunov's scheme (*godunov.py*). The desired method can be set in the *main.py* file.

The folder *reconstruction* contains the implementations of the ENO reconstruction that is necessary for the construction of high-order accurate, entropy stable methods and the minmod reconstruction used for computing reference solutions with the Lax-Friedrichs method.

The folder *stochastics* incorporates the stochastic collocation method and the folder *network* the code necessary for computing solutions on a network.

The implemented finite volume method and the numerical solver on networks can plot the current solutions in the folder *plots* during the computation. The frequency in which they are plotted can be adapted in the respective method.

The specific functions contained in the code are explained in the respective chapter the theoretical background is discussed in.

2 Hyperbolic Conservation Laws

In the context of this thesis, we study numerical and exact solutions to conservation laws. They are partial differential equations of the form

$$u_t + f(u)_x = 0, \quad (x,t) \in \mathbb{R} \times \mathbb{R}^+, \tag{2.1}$$

$$u(\cdot, 0) = u_0 \tag{2.2}$$

where $u : \mathbb{R} \times \mathbb{R}^+ \to \mathbf{U}$ is the vector of *conserved variables* that maps into a connected, non-empty set $\mathbf{U} \subset \mathbb{R}^m$ and $f : \mathbf{U} \to \mathbb{R}^m$ is the *flux function*. We call (2.1) a *scalar conservation law* if $m = 1$, i.e. if it consists of only one equation, and a *system of conservation laws* if $m > 1$.
These kinds of equations appear frequently in physical sciences. Some important examples are listed below, but we are especially interested in the shallow water equations, a nonlinear system of partial differential equations commonly used to model certain kinds of water waves and predict their behavior. Not only can they be applied to shallow waves in lakes or rivers, but also to waves with long wavelengths traversing oceans, such as tsunamis. However, we mainly focus on modeling water in a network of channels, where the computation of solutions across junctions is of particular interest.
Note that even though we mostly consider the one-dimensional case, (2.1) can be extended to multiple dimensions where we obtain

$$u_t + div(f(u)) = 0, \quad (x,t) \in \mathbb{R}^d \times \mathbb{R} \tag{2.3}$$

$$u(\cdot, 0) = u_0 \tag{2.4}$$

with $u : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbf{U}$, $\mathbf{U} \subset \mathbb{R}^m$, and $f = (f_1, \dots, f_d) : \mathbf{U} \to \mathbb{R}^{m \times d}$.
A drawback of using conservation laws for mathematical modeling is that an analytical solution exists only in very specific cases. Hence, in order to determine a solution, we have to employ numerical methods that should approximate the exact solution. However, the quality of such solutions highly depends on the underlying numerical scheme.
Nonetheless, before we investigate numerical methods in detail, we want to derive the integral and differential form of conservation laws. Afterwards, we introduce the concept of *hyperbolicity* and discuss examples of hyperbolic conservation laws, especially the shallow water system. Finally, we investigate weak solutions, a specific kind of solution that satisfies the conservation in a "weak sense". Since they lead to multiple spurious solutions, we employ entropy conditions to determine the "physically correct" one. In the last section, we discuss existence and uniqueness of entropy solutions. This chapter is mainly based on [1, 5, 7, 8].

2.1 Derivation of Hyperbolic Conservation Laws

We consider fluid in a channel with unit width and a known velocity $w(x,t)$ depending on x which refers to the distance along the channel and t which denotes the time that has passed since the initial time t_0 that is usually set to zero. Furthermore, we denote the density of a chemical tracer in the fluid by $u(x,t)$ and assume that its concentration is too small to affect fluid dynamics. Later, the conserved quantities will play the role of the chemical tracer.

The total mass of the tracer at a specific time t in the interval $[x_1, x_2]$ along the channel is given by

$$\int_{x_1}^{x_2} u(x,t)dx. \tag{2.5}$$

Here, we dropped the dependence on the width of the channel since we assume it to be constant.
If we now assume that channel walls are impermeable and the tracer is neither created nor destroyed within the process, the mass in one section of the tube changes only due to fluxes through the endpoints x_1 and x_2 of the section. Note that this forms the basis for *conservation*, an underlying concept in this work.
Let $F_i(t)$ denote the rate of particle flow through the fixed point x_i for $i = 1, 2$. We assume that $F_i(t)$ refers to a flux from the left to the right side and $-F_i(t)$ to a flux in the opposite direction. Therefore, $F_1(t)$ and $-F_2(t)$ both represent fluxes into the channel section $[x_1, x_2]$.
Hence, the mass of the chemical tracer in one section changes according to

$$\frac{d}{dt}\int_{x_1}^{x_2} u(x,t)dx = F_1(t) - F_2(t). \tag{2.6}$$

This is called the *integral form of conservation laws.*
Since we want to obtain an equation solvable for u, we have to determine the relation between the flux functions $F_i(t)$ and $u(x,t)$.
The velocity $w(x,t)$ determines how fast particles move past a certain point, the density u how many grams of the tracer one meter of fluid contains. Hence, the product of both refers to the rate of chemical particles passing at point x and therefore to the flux at this point, i.e. the flux at the point (x,t) in space-time is given by $w(x,t)u(x,t)$. Since the velocity is a known function, we can rewrite the flux as $f(u,x,t) = w(x,t)u$.
By assuming that $w(x,t) = \overline{w}$ is constant, we receive the *autonomous* system

$$f(u) = \overline{w}u. \tag{2.7}$$

That is, the value of the flux function at any point in space-time can be determined directly from the value of the conserved quantity in this point and does not depend on the location of that point itself.
If we consider an autonomous equation, we can write (2.6) in the following way:

$$\frac{d}{dt}\int_{x_1}^{x_2} u(x,t)dx = f(u(x_1,t)) - f(u(x_2,t)) \tag{2.8}$$

where $f(u(x,t))$ denotes the flux at position x at time t.
With

$$f(u(x_1,t)) - f(u(x_2,t)) = -f(u(x,t))\Big|_{x=x_1}^{x_2} = -\int_{x_1}^{x_2} \frac{\partial}{\partial x} f(u(x,t))dx \tag{2.9}$$

we can rewrite (2.8) by assuming that u and f are continuously differentiable and therefore using Leibniz integral rule:

$$\begin{aligned} &\frac{d}{dt}\int_{x_1}^{x_2} u(x,t)dx + \int_{x_1}^{x_2} \frac{\partial}{\partial x} f(u(x,t))dx = 0 \\ \iff &\int_{x_1}^{x_2} \frac{\partial}{\partial t} u(x,t)dx + \int_{x_1}^{x_2} \frac{\partial}{\partial x} f(u(x,t))dx = 0 \\ \iff &\int_{x_1}^{x_2} \left[\frac{\partial}{\partial t} u(x,t) + \frac{\partial}{\partial x} f(u(x,t))\right] dx = 0 \end{aligned} \tag{2.10}$$

Since (2.10) holds for every interval $[x_1, x_2]$ along the channel, we get

$$u(x,t)_t + f(u(x,t))_x = 0. \tag{2.11}$$

where the subscripts denote the partial derivatives with respect to t and x. We will refer to this equation as the *differential form of conservation laws* or simply as *conservation law*.

2.2 Hyperbolicity

Hyperbolicity is a crucial property of the equations we want to study in this thesis. We start by rewriting the problem (2.11) in *quasilinear form*

$$u(x,t)_t + f'(u(x,t))u(x,t)_x = 0. \tag{2.12}$$

In order for this problem to be *hyperbolic*, the $m \times m$ flux Jacobian matrix $f'(u)$ must have real eigenvalues at every point (x,t) in space-time and a corresponding set of m linearly independent eigenvectors. Therefore, every vector in $\mathbb{R}^m$ can be uniquely decomposed into a linear combination of these eigenvectors. This property is used later on to decompose the solution vector into distinct waves.
Note, that the hyperbolicity of (2.12) is equivalent to $f'(u(x,t))$ being diagonalizable with real eigenvalues.
We call the system (2.12) *strictly hyperbolic* if $f'(u(x,t))$ has m distinct real eigenvalues $\lambda^1 < \lambda^2 < ... < \lambda^m$.

2.3 Examples of Hyperbolic Conservation Laws

In this section, we give a few examples of hyperbolic conservation laws in ascending order of complexity. We start by introducing two scalar conservation laws before we continue to systems. In this thesis, we mainly focus on the shallow water equations, a nonlinear system of equations that is considered in detail in Section 2.4.

Example 2.1 (Linear Advection Equation)**.** *A basic example of a hyperbolic conservation law is the advection equation, a linear, scalar equation. It describes a substance being carried along by the motion of the fluid. The density $u = u(x,t)$ of the substance is described by*

$$u_t + au_x = 0. \tag{2.13}$$

The initial wave form $u(x,0) = u_0(x)$ simply propagates with unchanged shape at advection speed $a \in \mathbb{R}$, i.e. the solution is given by

$$u(x,t) = u(x - at). \tag{2.14}$$

Example 2.2 (Burger's Equation)**.** *In the next step, we still consider a scalar equation, but omit the linearity, which makes the construction of solutions slightly more complicated. An example of a nonlinear, scalar conservation law is Burger's equation*

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0. \tag{2.15}$$

This problem has been used for developing new theory and numerical methods for hyperbolic equations [5].

So far, we only considered scalar equations. However, most physically relevant hyperbolic conservation laws are systems of equations such as the shallow water or the Euler equations.

Example 2.3 (Linear Wave Equation)**.** *The simplest system of equations is an extension of the linear advection equation. The linear wave equation is given by*

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}_t + \begin{pmatrix} au_2 \\ au_1 \end{pmatrix} = 0 \tag{2.16}$$

where again $a \in \mathbb{R}$*. The system can be rewritten in the form*

$$u_t + Au_x = 0 \tag{2.17}$$

where $A \in \mathbb{R}$ *is a diagonalizable matrix, i.e.* $A = R\Lambda R^{-1}$ *for a diagonal matrix* Λ*. Multiplying (2.17) by* R^{-1} *and defining* $\hat{u} = R^{-1}u$*, the system can be rewritten as*

$$\hat{u}_t + \Lambda\hat{u}_x = 0. \tag{2.18}$$

Since Λ *is a diagonal matrix, each equation in this system is a linear advection equation* $(u_i)_t + \lambda_i(u_i)_x = 0$ *and has therefore a unique solution. Hence, the system itself has a unique solution that can be computed using the insights we gained for the advection equation in Example 2.1.*

Example 2.4 (Euler Equations)**.** *An important, physically relevant example are the Euler equations that are commonly used to model gas dynamics. They result from conservation of mass, momentum and energy and yield the system*

$$\begin{pmatrix} \rho \\ \rho w \\ E \end{pmatrix} + \begin{pmatrix} \rho w \\ \rho w^2 + p \\ w(E+p) \end{pmatrix} \tag{2.19}$$

where ρ *denotes the density,* w *the velocity,* p *the pressure and* E *the energy. This is an example of three equations that form a coupled system.*

2.4 Shallow Water Equations

The equations we mainly consider in the context of this thesis are the shallow water equations. As mentioned earlier, they can be used to model water waves where the horizontal length scale is much greater than the vertical length scale, e.g. tsunamis or waves in shallow water. The fluid flow is governed by conservation of mass and momentum.

To begin with, we consider fluid in a channel with unit width and assume the vertical velocity of the fluid is negligible and the horizontal velocity $w(x,t)$ is constant through any vertical cross-section. This holds for small amplitude waves in a fluid that is shallow relative to the wavelength. Furthermore, we assume that the fluid is incompressible, i.e. the density ρ does not depend on pressure and is therefore constant.

Denote the water height at time t and point x along the channel by $h = h(x,t)$. The total mass in an

interval $[x_1, x_2]$ at time t is given by

$$\int_{x_1}^{x_2} \rho h(x,t)dx. \tag{2.20}$$

The density of momentum at a certain point x and at time t is given by $\rho w(x,t)$. Integrating this vertically yields

$$\int_0^{h(x,t)} \rho w(x,t)dy = \rho w(x,t)y\Big|_{y=0}^{h(x,t)} = \rho h(x,t)w(x,t). \tag{2.21}$$

This is called the mass flux and denotes the mass of fluid passing through the cross-section at point x at time t.

Due to conservation of mass, the total mass in $[x_1, x_2]$ can only change due to fluxes through the endpoints of the interval. Hence, we can insert the mass (2.20) and the mass flux (2.21) in the differential form of conservation laws (2.11) to obtain the equation

$$h_t + (hw)_x = 0 \tag{2.22}$$

resulting from conservation of mass.

The second equation in the shallow water system is derived from conservation of momentum. The total momentum in $[x_1, x_2]$ is given by

$$\int_{x_1}^{x_2} \rho h(x,t)w(x,t)dx. \tag{2.23}$$

Analogously to our considerations above, the momentum in an interval only changes due to fluxes through the boundaries.

However, we still need to derive the momentum flux to apply (2.11). It is composed of two parts, the first one refers to the change of momentum caused by the movement of the fluid. For every quantity function $\overline{\rho}$ advected with the flow, the contribution to the flux for $\overline{\rho}$ will be of the form $\overline{\rho}w$. Since we consider the case $\overline{\rho} = \rho w$, we get a flux of the form ρw^2. Integrating this vertically leads to $\rho h w^2$.

However, the momentum is not only affected by advection but also by other forces that cause acceleration due to Newton's law. Since we assume there are no outside forces, the only force occurs in the fluid itself and is proportional to the pressure gradient which is simply p_x in one dimension. If we combine this with the advective flux and insert this and the total momentum in (2.11), we receive

$$(\rho hw)_t + (\rho hw^2 + p)_x = 0. \tag{2.24}$$

We now wish to determine the pressure p to receive an equation that depends only on the unknown functions $h(x,t)$ and $w(x,t)$. We exploit the *hydrostatic law* stating that the pressure at depth y is determined by ρgy and integrate this vertically to obtain the total pressure at (x,t)

$$\int_0^{h(x,t)} \rho gy \; dy = \frac{1}{2}\rho g y^2\Big|_{y=0}^{h(x,t)} = \frac{1}{2}\rho g h^2(x,t) \tag{2.25}$$

where g denotes the gravitational constant. Using this in (2.24) and dividing by ρ yields the second shallow water equation

$$(hw)_t + (hw^2 + \frac{1}{2}gh^2)_x = 0. \tag{2.26}$$

Combining the equations (2.22) and (2.26) results in the *one-dimensional system of shallow water equations*

$$u_t + f(u)_x = 0 \tag{2.27a}$$

where

$$u(x,t) = \begin{bmatrix} h \\ hw \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad f(u) = \begin{bmatrix} hw \\ hw^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} u_2 \\ (u_2)^2/u_1 + \frac{1}{2}g(u_1)^2 \end{bmatrix}. \tag{2.27b}$$

Sometimes, we denote the discharge by $q := hw$.
By adding a *source term* on the right-hand side, we can account for the change in the bathymetry. This results in the *shallow water system with bathymetry*

$$\begin{bmatrix} h \\ hu \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}_x = \begin{bmatrix} 0 \\ -ghb_x \end{bmatrix}. \tag{2.28}$$

However, since we consider artificially built channels in this work, we assume that the bathymetry is constant and, therefore, we do not require a source term.
Now, assume that h and u are smooth and $h > 0$. Using differentiation rules and (2.22), we can simplify (2.26):

$$\begin{aligned} &(hw)_t + (hw^2 + \frac{1}{2}gh^2)_x \\ &= h_t w + hw_t + h_x w^2 + 2hww_x + ghh_x \\ &= -(hw)_x w + hw_t + h_x w^2 + 2hww_x + ghh_x \\ &= -h_x w^2 - hww_x + hw_t + h_x w^2 + 2hww_x + ghh_x \\ &= hw_t + hww_x + ghh_x \end{aligned} \tag{2.29}$$

After dividing by h, we can write the conservation of momentum equation in the form

$$w_t + (\frac{1}{2}w^2 + gh)_x = 0. \tag{2.30}$$

If we now define $\varphi = gh$, we obtain the shallow water system for smooth solutions:

$$\begin{bmatrix} \varphi \\ w \end{bmatrix}_t + \begin{bmatrix} \varphi w \\ w^2/2 + \varphi \end{bmatrix}_x = 0 \iff u_t + f'(u)u_x = 0 \tag{2.31}$$

Here, the Jacobian matrix $f'(u)$ is of the form

$$f'(u) = \begin{bmatrix} 0 & 1 \\ -(u_2/u_1)^2 + gu_1 & 2u_2/u_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -w^2 + gh & 2w \end{bmatrix} \tag{2.32}$$

and has the eigenvalues

$$\lambda_1 = w - \sqrt{gh}, \quad \lambda_2 = w + \sqrt{gh} \tag{2.33}$$

with corresponding eigenvectors

$$r_1 = \begin{bmatrix} 1 \\ w - \sqrt{gh} \end{bmatrix}, \quad r_2 = \begin{bmatrix} 1 \\ w + \sqrt{gh} \end{bmatrix}. \tag{2.34}$$

We refer to the pair $\{\lambda_j, r_j\}$ as the *j-th eigenpair*. Note that the velocities of the waves in the solution are given by the eigenvalues, i.e. they move at the characteristic velocities $\lambda_1 = w - \sqrt{gh}$, $\lambda_2 = w + \sqrt{gh}$. Since w refers to the fluid velocity, the velocity of the waves relative to the fluid is given by $c = \pm\sqrt{gh}$. To compute the gradient of the eigenvalues, we first rewrite the eigenvalues in terms of u_1, u_2:

$$\lambda_1 = \frac{u_2}{u_1} - \sqrt{gu_1}, \quad \lambda_2 = \frac{u_2}{u_1} + \sqrt{gu_1} \tag{2.35}$$

Then, this yields

$$\nabla\lambda_j(u) = \begin{bmatrix} -\frac{u_2}{(u_1)^2} \mp \frac{1}{2}\sqrt{g/u_1} \\ \frac{1}{u_1} \end{bmatrix} = \begin{bmatrix} -\frac{w}{h} \mp \frac{1}{2}\sqrt{g/h} \\ \frac{1}{h} \end{bmatrix}. \tag{2.36}$$

In general, if

$$\nabla\lambda_j(u) \cdot r_j(u) \neq 0, \ \forall u, \tag{2.37}$$

we call the wave corresponding to the j-th eigenpair *genuinely nonlinear*. On the other hand, if

$$\nabla\lambda^j(u) \cdot r^j(u) \equiv 0, \quad \forall u, \tag{2.38}$$

we call it *linearly degenerate*. In both cases, the associated waves are discontinuities between the states u_* and u satisfying the *Rankine-Hugoniot jump conditions*

$$s(u_* - u) = f(u_*) - f(u) \tag{2.39}$$

where s denotes the *shock speed*. In contrast, smooth differentiable transitions arising in the solution are called *rarefaction waves*.
For the shallow water equations, we obtain

$$\nabla\lambda_j(u) \cdot r_j(u) = \mp\frac{3}{2}\sqrt{g/h} \neq 0, \ \forall u \tag{2.40}$$

and hence both resulting waves are genuinely nonlinear. This property will be important if we investigate solutions to Riemann problems for the shallow water system in Chapter 3.
Note that so far, we only considered the one-dimensional shallow water equations. However, they can easily be extended to more dimensions. It then takes the form

$$u_t + f(u)_x + g(u)_y = 0 \tag{2.41a}$$

with

$$u(x,y,t) = \begin{bmatrix} h \\ hw_x \\ hw_y \end{bmatrix}, \quad f(u) = \begin{bmatrix} hw_x \\ hw_x^2 + \frac{1}{2}gh^2 \\ hw_xw_y \end{bmatrix}, \quad g(u) = \begin{bmatrix} hw_y \\ hw_xw_y \\ hw_y^2 + \frac{1}{2}gh^2 \end{bmatrix} \tag{2.41b}$$

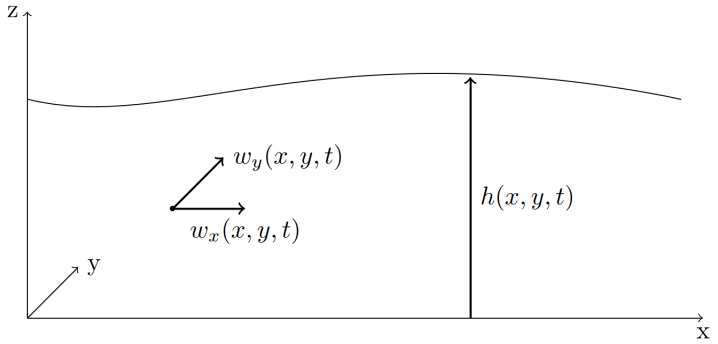


Figure 2.1: Two-dimensional Shallow Water System

Here, $h(x,y,t)$ denotes the two-dimensional water height and $w_x(x,y,t)$, $w_y(x,y,t)$ the fluid velocities in the x- and y-direction, respectively.

where $h = h(x,y,t)$ denotes the two-dimensional water height, g the gravitational constant and $w_x = w_x(x,y,t)$, $w_y = w_y(x,y,t)$ the horizontal velocities in the $x-$ and $y-$direction, respectively.

2.5 Weak Solutions and Entropy Conditions

First, recall that Equation (2.11) is only valid for smooth solutions. However, the integral form (2.6) still holds for non-smooth solutions. Hence, we introduce *weak solutions* to account for discontinuities in the solution.
Integrating (2.10) along $[t_1,t_2]$ yields

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2}\left[u_t + f(u)_x\right]dx\,dt = 0. \tag{2.42}$$

Now, instead of particular limits t_1,t_2,x_1,x_2 we consider

$$\int_0^\infty\int_{-\infty}^\infty\left[u_t + f(u)_x\right]\phi(x,t)dx\,dt = 0 \tag{2.43}$$

where $\phi(x,t)$ is taken from a set of certain functions we will specify later. Note that the particular choice

$$\phi(x,t) = \begin{cases} 1, & x\in[x_1,x_2],\ t\in[t_1,t_2] \\ 0, & \text{else} \end{cases} \tag{2.44}$$

leads to (2.42).
If we now assume $\phi \in C_c^1$, that is, ϕ is continuously differentiable and has compact support, we can integrate (2.43) by parts:

$$\begin{aligned}
&\int_0^\infty\int_{-\infty}^\infty\left[\nabla\cdot(u,f(u))\right]\phi(x,t)dx\,dt \\
=&\int_\Gamma\left[(u,f(u))\,\phi(x,t)\right]\cdot\hat{n}\,dx - \int_0^\infty\int_{-\infty}^\infty\left[u\phi_t + f(u)\phi_x\right]dx\,dt = 0 \\
\iff& -\int_{-\infty}^\infty u(x,0)\phi(x,0)dx + \lim_{t\to\infty}\int_{-\infty}^\infty u(x,t)\phi(x,t)dx \\
&-\lim_{x\to-\infty}\int_0^\infty f(u(x,t)\phi(x,t)dt + \lim_{x\to\infty}\int_0^\infty f(u(x,t)\phi(x,t)dt \\
&-\int_0^\infty\int_{-\infty}^\infty\left[u\phi_t + f(u)\phi_x\right]dx\,dt = 0
\end{aligned} \tag{2.45}$$

where $\hat{n}$ is the outward-pointing unit normal vector and Γ is the boundary of our domain.
Note that the boundary terms at $x, t = \pm\infty$ vanish, since ϕ is compactly supported and is therefore identically zero outside of a bounded region in the $x - t$ plane. Hence, we finally obtain

$$\int_0^\infty \int_{-\infty}^\infty [u\phi_t + f(u)\phi_x]\, dx\, dt = -\int_{-\infty}^\infty u(x,0)\phi(x,0)dx. \tag{2.46}$$

Now, we can refer to a function $u(x,t)$ as *weak solution of the conservation law* (2.11) if (2.46) holds for all functions $\phi \in C_c^1$ with given initial data $u(x,0)$.
Since we only compute derivatives of ϕ in (2.46), this equation also holds for a discontinuous function u.
In conclusion, we are now able to consider solutions to the conservation law that might contain discontinuities. However, this generalization leads to multiple numerical solutions. We employ *entropy conditions* to characterize the "physically correct" one.
First, we introduce *entropy pairs*.

Definition 2.1. A pair of functions (η, q) with $\eta \in C^2(\mathbb{R}^m, \mathbb{R})$ and $q = (q^1, \dots, q^d) \in C^2(\mathbb{R}^m, \mathbb{R}^d)$ is called an *entropy pair* for the conservation law (2.3) if

$$q'(u)^T = \eta'(u) \cdot f'(u) \quad \forall u \in \mathbf{U},$$

i.e.

$$(q^k)'(u)^T = \eta'(u) \cdot (f^k)'(u) \quad \text{for } k = 1, \dots, d. \tag{2.47}$$

An entropy pair is called convex if η is convex. Furthermore, we define the *entropy variable*

$$v(u) := \eta'(u) \tag{2.48}$$

and the *entropy potential*

$$\psi(v) := v \cdot f(u(v)) - q(u(v)). \tag{2.49}$$

They will both play an important role when we introduce entropy stable and entropy conservative methods.

Note that $u \mapsto v(u)$ is invertible if η is strongly convex, i.e. $\eta'' > 0$, and we denote the inverse by $u(v)$. Then,

$$\frac{du}{dv} = (\eta''(u(v)))^{-1} \tag{2.50}$$

is a symmetric positive definite matrix and

$$\frac{d}{dv} f(u(v)) \tag{2.51}$$

is symmetric.
Now, by taking the inner product of $\eta'(u)$ with (2.3), we see that smooth solutions u satisfy the

additional conservation law

$$0 = \eta'(u) \cdot (u_t + div(f(u)) = \eta(u)_t + div(q(u)). \tag{2.52}$$

This can easily be verified for the one-dimensional, scalar case. There, we first apply the chain rule for $\eta'(u) \cdot u_t = \eta(u)_t$. Then, we derive

$$\begin{aligned} \eta'(u) \cdot f(u)_x &= v \cdot f(u)_x \\ &= (v \cdot f(u))_x - v_x \cdot f(u) \\ &= (v \cdot f(u) - \psi(v))_x - v_x \cdot f(u) + v_x \cdot \psi'(v) \\ &= q(u)_x. \end{aligned} \tag{2.53}$$

Note that we used the product rule and the definitions of the entropy variable $\eta'(u) = v(u)$ and the entropy potential $\psi(v) = v \cdot f(u(v)) - q(u(v))$.
Equation (2.52) is called the *entropy equality*. From now on, we impose the stability criterion that entropy never decreases. That enables us to identify the unique physically correct solution.

Definition 2.2. Let (η, q) be a convex entropy pair. Then, we call a function $u \in L^\infty(\mathbb{R}^d \times \mathbb{R}^+, \mathbb{R}^m)$ *admissible with respect to* (η, q) if

$$\eta(u)_t + div(q(u)) \leq 0 \tag{2.54}$$

in $\mathcal{D}'(\mathbb{R}^d \times \mathbb{R}^+, \mathbb{R})$, i.e. if

$$\int_{\mathbb{R}^d \times \mathbb{R}^+} \eta(u)\phi_t + q(u)div(\phi)dxdt + \int_{\mathbb{R}^d} \eta(u_0(x))\phi(x,0)dx \geq 0 \tag{2.55}$$

for all $0 \leq \phi \in \mathcal{D}(\mathbb{R}^d \times \mathbb{R}^+, \mathbb{R})$. Furthermore, we define *entropy solutions* as weak solutions of (2.3) that are admissible with respect to all convex entropy pairs.

Integrating (2.54) in space and time and assuming $|q(u)| \to 0$ as $|x| \to \infty$ yields

$$\int_{\mathbb{R}^d} \eta(u(x,t))dx \leq \int_{\mathbb{R}^d} \eta(u_0(x))dx, \tag{2.56}$$

i.e. the total amount of entropy of an entropy solution decreases over time.
Let us consider some examples of entropy pairs for different hyperbolic equations. For a scalar conservation law, all convex functions η give rise to an entropy pair (η, q) by defining

$$q(u) = \int_0^u \eta'(s)f'(s)ds. \tag{2.57}$$

For simplicity, one often chooses the square entropy $\eta(u) = \frac{u^2}{2}$. In this case the entropy variable is given by $v(u) = \eta'(u) = u$.
This even transfers to some linear systems. For the linear wave equation, (2.16) we can, for instance, use $\eta(u) = \frac{1}{2}\left((u_1)^2 + (u_2)^2\right)$ with the corresponding entropy flux $q(u) = \frac{1}{2}u^T Au = au_1u_2$.
Even though there is a variety of available entropy pairs for scalar conservation laws, the choices for general nonlinear systems are much more limited. For the shallow water system, the entropy we use in

the following is given by the total energy of the solution:

$$\eta(u) = \frac{1}{2}\left(hw^2 + gh^2\right), \quad q(u) = \frac{1}{2}hw^3 + gwh^2 \tag{2.58}$$

Using this, we can define the corresponding entropy variable $v = \eta'(u)$ and the entropy potential $\psi(v) = v \cdot f(u(v)) - q(u(v))$ by

$$v = \begin{pmatrix} gh - \frac{w^2}{2} \\ w \end{pmatrix}, \quad \psi(v) = \frac{1}{2}gwh^2. \tag{2.59}$$

In the following section, we discuss existence and uniqueness of entropy solutions.

2.6 Existence and Uniqueness

In 1970, it was shown by Kruzkow [12] that there exists a unique entropy solution in the class of functions of bounded variation for scalar conservation laws.

Theorem 2.1. *Let $u_0 \in L^\infty(\mathbb{R}^d)$ be the initial function in (2.3). Then, there exists a unique entropy solution u for (2.3). It satisfies*

$$\|u(\cdot,t)\|_{L^\infty(\mathbb{R}^d)} \leq \|u_0\|_{L^\infty(\mathbb{R}^d)} \quad \forall t > 0. \tag{2.60}$$

Unfortunately, for general systems of hyperbolic conservation laws, there is no similar global well-posedness result available, there are only partial results. The existence and uniqueness of the entropy solution to one-dimensional Riemann problems where the two constant initial states are sufficiently close has been shown by Lax in [13]. Furthermore, Glimm proved the existence of a weak solution to the Cauchy problem (2.3) for all $u_0 \in \mathbf{D}$, where $\mathbf{D}$ is the L^1-closure of the set of all piecewise constant functions with sufficiently small total variation [14]. This was done by showing that the *random choice method* is stable with respect to the *Glimm functional*. Lastly, Bressan et al. proved that the front tracking method is stable with respect to the Glimm functional and that it converges to the unique entropy solution for all $u_0 \in \mathbf{D}$ [15, 16].
In summary, we obtain the following theorem.

Theorem 2.2. *Consider the one-dimensional case, i.e. $d = 1$. Furthermore, assume that the conservation law (2.1) is strictly hyperbolic and all the wave families are either genuinely nonlinear or linearly degenerate. Then, for all initial functions $u_0 \in L^1$ with sufficiently small total variation, there exists a unique entropy solution to (2.1).*

3 Exact Solutions to Riemann Problems for the Shallow Water Equations

In this chapter, we analyse exact solutions to Riemann problems for the shallow water system. Whenever we evaluate a numerical scheme, we need a solution to compare the numerical results to, a so-called *reference solution*. This can either be a numerical solution computed with a different and already generally approved scheme on a very fine grid or an exact solution. However, for general conservation laws with arbitrary initial data, we are not able to derive an analytical solution. Hence, we can only use exact solutions in very specific cases, e.g. for obtaining solutions to the advection equation (2.13) where the solution is given by (2.14). While the linear advection equation can be solved regardless of the initial data, the choices for the shallow water equations are drastically limited. In this work, we consider initial Riemann problems and we analyse in this chapter how they can be solved.
Later, we also compare the solution of a Riemann problem on a network as described in Section 11.6 to the numerical solution obtained by the TeCNO scheme on a network. The results of this comparison can be found in Chapter 12.
This chapter is based on Chapter 5 in [9] and Chapter 13 in [5].
Consider the *Riemann problem*

$$u_t + f(u)_x = 0 \tag{3.1a}$$

where the initial data is piecewise constant, i.e.

$$u(x,0) = \begin{cases} u_l, & x < 0 \\ u_r, & x > 0 \end{cases} . \tag{3.1b}$$

We consider the shallow water equations as the underlying hyperbolic conservation law, i.e. u, $f(u)$ are given in (2.27b). For simplicity, we assume $g = 1$. The solution to this problem consists of two waves which we refer to as 1-wave and 2-wave. Furthermore, if we already know that the j-th wave is a rarefaction or a shock wave, we call them a j-rarefaction or j-shock, respectively.
In the following, we first consider two examples of standard Riemann problems for the shallow water system. Afterwards, we derive specific formulas in order to compute exact solutions to these kinds of problems.

3.1 Examples of Riemann Problems

Since the shallow water system (2.27) consists of two equations, the solution to the corresponding Riemann problem contains two waves. These waves are separated by an intermediate state u_m that has to be determined for each problem individually. Furthermore, both characteristic fields are genuinely nonlinear and therefore each wave family is either a shock or a rarefaction wave. However, the specific combination depends on the given initial data and hence we have to determine the kind of wave before computing the appropriate solution.
To emphasize the importance of determining the correct composition of shock and rarefaction waves, we give two examples of different Riemann problems in the following.

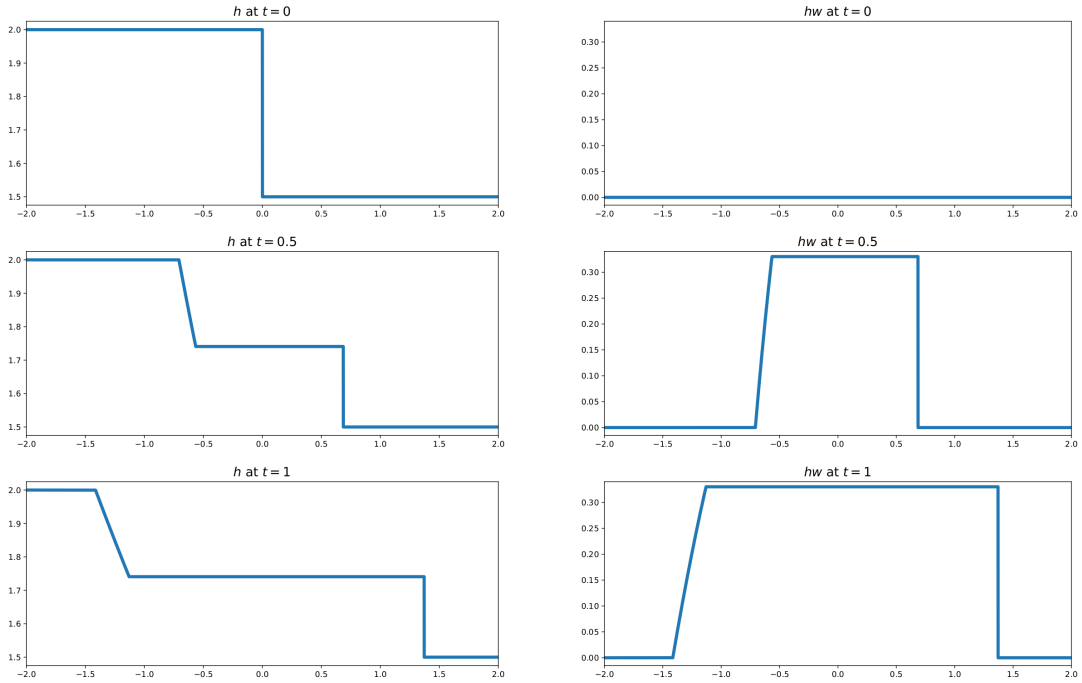


Figure 3.1: Solution to Dam Break Riemann Problem

The initial data is given by $h_l = 2$, $h_r = 1.5$. The left column shows the water height h at time steps 0, 0.5 and 1 whereas the right column shows the momentum hu at the same time.

3.1.1 Dam-Break Riemann Problem

The first example is the *dam-break Riemann problem*. It plays an important role in Chapter 12 when we analyse numerical results obtained by the TeCNO scheme on a network.
Consider (2.27) with initial data

$$h(x,0) = \begin{cases} h_l & x < 0 \\ h_r & x > 0 \end{cases}, \quad w(x,0) = 0 \tag{3.2}$$

where $h_l, h_r > 0$, $h_l \neq h_r$ i.e. our initial water wave is piecewise constant with a jump at $x = 0$. Note that the problem models the situation of a dam separating two water levels that breaks at $t = 0$. Without loss of generality, assume $h_l > h_r$. At the initial time, the water on both sides is at rest, i.e. $w_l = w_r = 0$. However, at time $t > 0$, water flows from the higher left water level to the right into shallower water. It is led by a shock wave moving at speed w_m separating the lower water height h_r from an intermediate water height h_m. Furthermore, on the left of the intermediate state, a rarefaction develops when the water on the higher level h_l is accelerated to the right.
The simulation in Figure 3.1 shows the exact solution in this situation and one can clearly recognize the different kinds of waves.
In fact, the solution to the dam break problem always consists of one rarefaction and one shock wave. However, their shape and direction is mirrored when we consider $h_l < h_r$.

3.1.2 Two-Shock Riemann Problem

Now, we consider initial data of the type

$$h(x,0) = h_0, \quad w(x,0) = \begin{cases} w_l, & x < 0 \\ -w_l, & x > 0 \end{cases} . \tag{3.3}$$

If $w_l > 0$, the problem models the situation of two streams of water coming from opposite directions at the same speed colliding at the initial time $t = 0$. On both sides of the collision, shock waves form and move sideways, bringing the fluid to rest.
The solution is symmetric in x at all times in the way that $h(-x,t) = h(x,t)$ and $w(-x,t) = -w(x,t)$. From the latter equation, we can derive $w_m = 0$ for the intermediate state.
Therefore, the two shock waves move apart at the same speed in a mirrored shape.

3.2 Determining Solutions to the Shallow Water Equations

As the two examples above illustrate, the combination of shock and rarefaction waves arising in the solution to the Riemann problem depends on the specific initial data and therefore has to be determined individually for each problem. To derive a universal approach for solving the Riemann problem exactly, we have to be able to solve it for every pair of states u_l, u_r.
Hence, the following steps have to be executed:

1. Determine the intermediate state u_m.
2. Determine for each of the two arising waves, whether they are a shock or a rarefaction wave.
3. Determine the structure of the solution through each rarefaction wave.

We address each of these steps in the subsequent sections in order to derive a procedure to compute the exact solution. To begin with, we want to identify all states that can be correctly connected to a given state, i.e. u_l or u_r.

3.3 Shock Waves

As we have seen in both examples above, the solution to certain Riemann problems can consist of discontinuous solutions, i.e. shock waves. However, note that not all arbitrary combinations of states lead to valid shock waves but only those satisfying the Rankine-Hugoniot jump conditions (2.39). For the shallow water equations, we obtain the system

$$s(h_* - h) = h_* w_* - hw \tag{3.4a}$$

$$s(h_* w_* - hw) = h_* w_*^2 - hw^2 + \frac{1}{2}(h_*^2 - h^2). \tag{3.4b}$$

Note that u_* denotes the fixed state, i.e. $u_* = u_l$ or $u_* = u_r$ and hence h_* and w_* are given and we want to determine all possible values for s, h and w such that (3.4) is fulfilled. Therefore, we have a system of two equations with three unknowns and expect to find a family of solutions with one remaining parameter.
We leave h as the parameter and compute w and s for each value of h. If we depict this in the $h - hw$ plane, we receive a curve called the *Hugoniot-locus*. In fact, we will obtain two different families of

one-parameter solutions, each of which refers to one wave. Since the intermediate state between two shocks must fulfill the Rankine-Hugoniot condition for u_l as well as u_r, u_m is given by the intersection of both corresponding Hugoniot-loci.
However, we still do not know how to compute w and s. First, we use (3.4a) to obtain

$$s = \frac{h_* w_* - hw}{h_* - h}. \tag{3.5}$$

Inserting this in (3.4b), multiplying with $(h_* - h)$ and dividing by $h_* h$ yields a formula for w depending on h:

$$\begin{aligned} &\frac{(h_* w_* - hw)^2}{h_* - h} = h_* w_*^2 - hw^2 + \frac{1}{2}(h_*^2 - h^2) \\ \iff &h_*^2 w_*^2 - 2h_* w_* hw + h^2 w^2 - h_*^2 w_*^2 + h_* h w_*^2 + h h_* w^2 - h^2 w^2 - \frac{1}{2}(h_*^3 - h_*^2 h - h_* h^2 + h^3) = 0 \\ \iff &w^2 - 2w_* w + \left[w_*^2 - \frac{1}{2}\left(\frac{h_*}{h} - \frac{h}{h_*}\right)(h_* - h)\right] = 0 \end{aligned} \tag{3.6}$$

This equation must now be solved for w given in terms of h. Hence, the equation holds for

$$\begin{aligned} w(h) &= w_* \pm \frac{1}{2}\sqrt{4w_*^2 - 4\left[w_*^2 - \frac{1}{2}\left(\frac{h_*}{h} - \frac{h}{h_*}\right)(h_* - h)\right]} \\ &= w_* \pm \sqrt{\frac{1}{2}\left(\frac{h_*}{h} - \frac{h}{h_*}\right)(h_* - h)}. \end{aligned} \tag{3.7}$$

As expected, for $h \neq h_*$ we obtain two different values for w corresponding to the two shock waves.
For $h_* = h$, we obtain $w_* = w$ which is expected due to the fact that the curve has to pass through (h_*, w_*) according to the Rankine-Hugoniot conditions.
To continue, we first introduce a linearized model for the shallow water equations. We assume the water has an approximately constant depth $h_0 > 0$ and is moving at a constant velocity $u_0 \geq 0$. Let u denote the perturbations from this constant state:

$$u = \begin{bmatrix} h - h_0 \\ hu - h_0 w_0 \end{bmatrix}, \quad u_0 = \begin{bmatrix} h_0 \\ h_0 w_0 \end{bmatrix}. \tag{3.8}$$

In the nonlinear model, the velocity of the wave increases proportionally to the water depth (see Section 2.4). If we consider small amplitude waves and therefore only small perturbations to the constant state h_0, we can ignore the slight variations in speed and obtain a linear system.
Expanding the flux function and dropping terms of order $\mathcal{O}(\|u\|^2)$ yields the linear system

$$u_t + Au_x = 0 \tag{3.9}$$

where $A = f'(u_0)$. If $u \approx u_*$, that is, we have a weak shock, we expect the linearized model to provide good results. Hence, we show now that each curve has to be tangent to one eigenvector $r^p(u_*)$ for $p = 1, 2$. We use this fact to determine which curve corresponds to which shock.

If we multiply (3.7) by h and define α as the difference of h and h_*, i.e. $h = h_* + \alpha$, we obtain

$$hw = h_* w_* + \alpha \left[w_* \pm \sqrt{h_* \left(1 + \frac{\alpha}{h_*}\right)\left(1 + \frac{\alpha}{2h_*}\right)} \right]. \tag{3.10}$$

Therefore, u is given by

$$u = u_* + \alpha \begin{bmatrix} 1 \\ w_* \pm \sqrt{h_* \pm \mathcal{O}(\alpha)} \end{bmatrix}. \tag{3.11}$$

As $\alpha \to 0$, u approaches u_*, since $h \approx h_*$ and $hw \approx h_* w_*$. In this case, we can neglect the $\mathcal{O}(\alpha)$ term and we find that these curves approach the point u_* tangent to the vectors

$$\begin{bmatrix} 1 \\ w_* \pm \sqrt{h_*} \end{bmatrix}. \tag{3.12}$$

Recall that these are the eigenvectors of the Jacobian matrix (2.32) for the shallow water system evaluated at u_*. Therefore, the curves are tangent to the eigenvectors $r^p(u_*)$ as we expected. Recall that the vector with the "−" sign gives the first eigenvector $r^1(u_*)$ and the one with the "+" sign the second one $r^2(u_*)$.
Hence, we can determine which equation in (3.11) refers to which shock. The one with the "−" sign gives the Hugoniot locus of the 1-shock and the one with the "+" sign the locus of the 2-shock.

All-Shock Riemann Solution With the considerations above, we can determine the middle state u_m between constant states u_r and u_l when the interfaces are shocks. Note that u_m has to be connected to u_l by a 1-shock and to u_r by a 2-shock. Therefore, u_m must satisfy (3.10) with a "−" sign and $u_* = u_l$, which is

$$h_m w_m = h_l w_l + \alpha \left[w_l - \sqrt{h_l \left(1 + \frac{\alpha}{h_l}\right)\left(1 + \frac{\alpha}{2h_l}\right)} \right]. \tag{3.13}$$

Dividing this by h_m and using $\alpha = h_m - h_l$ leads to

$$\begin{aligned} w_m &= \frac{h_l}{h_m} w_l + \frac{h_m - h_l}{h_m} \left[w_l - \sqrt{h_l \left(1 + \frac{\alpha}{2h_l} + \frac{\alpha}{h_l} + \frac{\alpha^2}{2h_l^2}\right)} \right] \\ &= w_l - (h_m - h_l) \sqrt{\frac{1}{2} \frac{h_l}{h_m^2} \left(\frac{h_m^2 + h_l h_m}{h_l^2} \right)} \\ &= w_l - (h_m - h_l) \sqrt{\frac{1}{2} \left(\frac{1}{h_m} + \frac{1}{h_l} \right)}. \end{aligned} \tag{3.14}$$

Analogously, w_m must satisfy (3.10) with a "+" sign and $w_* = w_r$ which yields

$$w_m = w_r + (h_m - h_r) \sqrt{\frac{1}{2} \left(\frac{1}{h_m} + \frac{1}{h_r} \right)}. \tag{3.15}$$

Note that both equations must be fulfilled in order for w_m to be a valid middle state between shocks. Therefore, we obtain a system of two equations with two unknowns h_m and w_m that needs to be solved.

Equating (3.14) and (3.15) yields

$$w_l - (h_m - h_l)\sqrt{\frac{1}{2}\left(\frac{1}{h_m} + \frac{1}{h_l}\right)} = w_r + (h_m - h_r)\sqrt{\frac{1}{2}\left(\frac{1}{h_m} + \frac{1}{h_r}\right)}. \tag{3.16}$$

This must be solved for the only remaining variable h_m which can be done by using an iterative method for nonlinear equations, e.g. Newton's method.

3.4 Rarefaction Waves

As we have seen in the dam-break example (3.2), the solution can also contain smooth waves, so-called rarefaction waves. Considering (3.1), it can easily be seen that the equation and the initial data are scale-invariant, i.e. they are invariant under the map $(x,t) \to (kx,kt)$. Hence, we are looking for scale-invariant solutions, i.e. solutions of the form

$$u(x,t) = \hat{u}(x/t) = \hat{u}(\xi) \tag{3.17}$$

where $\xi = x/t$.
Inserting this in the original equation yields

$$\hat{u}_t + f(\hat{u})_x = 0 \quad \Longleftrightarrow \quad -\frac{x}{t^2}\hat{u}' + \frac{1}{t}f'(\hat{u})\hat{u}' = 0 \quad \Longleftrightarrow \quad f'(\hat{u})\hat{u}' = \frac{x}{t}\hat{u}'. \tag{3.18}$$

Therefore, $\hat{u}'$ is an eigenvector of the Jacobian $f'(\hat{u})$ with eigenvalue $x/t = \xi$. Recall from Section 2.2 that $f'(\hat{u})$ has m eigenvectors $r_1, \dots, r_m$ with corresponding eigenvalues $\lambda_1, \dots, \lambda_m$ and consequently there exists a $j \in \{1, \dots, m\}$ such that (possibly after appropriate scaling)

$$\hat{u}'(\xi) = r_j(\hat{u}(\xi)), \quad \lambda_j(\hat{u}(\xi)) = \xi. \tag{3.19}$$

Hence, equation (3.18) becomes

$$f'(\hat{u})\hat{u}' = \lambda_j(\hat{u})\hat{u}'. \tag{3.20}$$

We call a function $\hat{u}$ satisfying (3.20) an *integral curve of the vector field* r_j.
From the second equation in (3.19), we see that if $\hat{u}(\xi_l) = u_l$ and $\hat{u}(\xi_r) = u_r$ for $\xi_l, \xi_r \in \mathbb{R}$, then $\xi_l = \lambda_j(u_l)$ and $\xi_r = \lambda_j(u_r)$. Hence, if we assume

$$\hat{u}(\lambda_j(u_l)) = u_l, \quad \hat{u}(\lambda_j(u_r)) = u_r \tag{3.21}$$

the function $\hat{u}(x/t)$ continuously connects the given state u_l to u_r for a fixed time t. Hence, $\xi = \lambda_j(\hat{u}(x/t))$ must increase which yields $\lambda_j(u_l) < \lambda_j(u_r)$.
In summary, we obtain a *rarefaction wave*, i.e. a solution of the form

$$u(x,t) = \begin{cases} u_l, & x \le t\lambda_j(u_l) \\ \hat{u}(x/t), & t\lambda_j(u_l) \le x \le t\lambda_j(u_r) \\ u_r, & x \ge t\lambda_j(u_r) \end{cases} \tag{3.22}$$

where $\hat{u}$ is required to fulfill (3.19) and (3.21).
Note that (3.19) fixes the normalization of the eigenvector $r_j(u)$ to

$$\nabla\lambda_j(u)\cdot r_j(u) = 1. \tag{3.23}$$

Hence, the wave family must be genuinely nonlinear in order for this normalization to be possible. This leads to the following theorem from [9].

Theorem 3.1 (Genuinely Nonlinear Waves)**.** *Let D be a domain in $\mathbb{R}^n$. Assume the equation (3.1a) is strictly hyperbolic and the j-th wave family is genuinely nonlinear. Hence, we can find a scaling of the j-th eigenvector r_j of the Jacobian $f'(u)$ with corresponding eigenvector $\lambda_j(u)$ such that*

$$\nabla\lambda_j(u)\cdot r_j(u) = 1 \tag{3.24}$$

for $u \in D$.
Now, let $u_l \in D$. Then, there exists a curve $R_j(u_l)$ emanating from u_l such that for each $u_r \in R_j(u_l)$ the rarefaction wave (3.22) is a solution for the Riemann problem (3.1) where $\hat{u}$ satisfies (3.19) and (3.21).

In the following, we derive the explicit formula for computing rarefaction waves for the shallow water system.
Recall from Section 2.4 that

$$\nabla\lambda_j(u)\cdot r_j(u) = \mp\frac{3}{2}\sqrt{1/h} \neq 0,\ \forall u \tag{3.25}$$

and hence both wave families are genuinely nonlinear. Therefore, we can normalize the eigenvectors in order to satisfy (3.24) and we redefine r_1, r_2 in the following by

$$r_1 = -\frac{2}{3}\sqrt{h}\begin{pmatrix} 1 \\ w-\sqrt{h} \end{pmatrix} \tag{3.26a}$$

$$r_2 = \frac{2}{3}\sqrt{h}\begin{pmatrix} 1 \\ w+\sqrt{h} \end{pmatrix}. \tag{3.26b}$$

Since $\hat{u}'(\xi) = r_j(\hat{u}(\xi))$, for r_1 we obtain the two ordinary differential equations

$$h'(\xi) = -\frac{2}{3}\sqrt{h(\xi)} \tag{3.27}$$

$$q'(\xi) = -\frac{2}{3}\sqrt{h(\xi)}\left(\frac{q(\xi)}{h(\xi)} - \sqrt{h(\xi)}\right). \tag{3.28}$$

Recall that $q = wh$. Using the chain rule, we see that this implies

$$\frac{dq}{dh} = \frac{dq}{d\xi}\frac{d\xi}{dh} = \frac{q}{h} - \sqrt{h} = \lambda_1. \tag{3.29}$$

Integrating this yields

$$q = q(h) = q_l\frac{h}{h_l} - 2h(\sqrt{h} - \sqrt{h_l}) \tag{3.30}$$

for a given state $u_l = (h_l, q_l)$. If we insert this formulation for q in λ_1, we can derive that $h < h_l$ is necessary in order for $\lambda_1(u)$ to increase.

We proceed similarly for the second wave family to derive $h > h_r$ in this case.
For r_2, we obtain

$$q = q(h) = q_r \frac{h}{h_r} + 2h(\sqrt{h} - \sqrt{h_r}) \tag{3.31}$$

for a given state $u_r = (h_r, q_r)$.
In summary, we obtain the following formulas for computing rarefaction waves in terms of h in the first and second wave family for the shallow water system:

$$\text{1-wave:} \quad q = R_1(h; u_l) = q_l \frac{h}{h_l} - 2h(\sqrt{h} - \sqrt{h_l}) \quad \text{for } h \in (0, h_l] \tag{3.32a}$$

$$\text{2-wave:} \quad q = R_2(h; u_l) = q_l \frac{h}{h_l} + 2h(\sqrt{h} - \sqrt{h_l}) \quad \text{for } h \in (h_l, \infty) \tag{3.32b}$$

While the expressions above follow for any normalization of the eigenvectors, we have to use (3.26a) when we want to compute the rarefaction waves in terms of ξ. With similar transformations as above, we obtain

$$\hat{u}_1(\xi) = R_1(\xi, u_l) = \begin{pmatrix} \frac{1}{9}(w_l + 2\sqrt{h_l} - \xi)^2 \\ \frac{1}{27}(w_l + 2\sqrt{h_l} + 2\xi)(w_l + 2\sqrt{h_l} - \xi)^2 \end{pmatrix} \tag{3.33a}$$

for a 1-rarefaction with $\xi \in [w_l - \sqrt{h_l}, w_l + 2\sqrt{h_l})$ and

$$\hat{u}_2(\xi) = R_2(\xi, u_l) = \begin{pmatrix} \frac{1}{9}(-w_l + 2\sqrt{h_l} + \xi)^2 \\ \frac{1}{27}(w_l - 2\sqrt{h_l} + 2\xi)(-w_l + 2\sqrt{h_l} + \xi)^2 \end{pmatrix} \tag{3.33b}$$

for a 2-rarefaction with $\xi \in [\lambda_2(u_l), \infty)$.
In conclusion, we can compute the exact solution to the Riemann problem (3.1) in the case of rarefactions by using the formula

$$u(x,t) = \begin{cases} u_l & \text{for } x \le \lambda_j(u_l)t \\ R_j(\xi; u_l) & \text{for } \lambda_j(u_l)t \le x \le \lambda_j(u_r)t \\ u_r & \text{for } x \ge \lambda_j(u_r)t \end{cases} . \tag{3.34}$$

Middle States between Rarefactions Now, we derive the formulas to determine the middle state u_m explicitly. Later, we use them in combination with the ones for shocks to determine the correct middle state despite the combination of waves.
Recall that we can determine the middle state between two shocks by computing the intersection of the Hugoniot-loci. The procedure of computing the middle state is similar for rarefactions. Note that dividing (3.32a) by h implies that w_m must satisfy

$$w_m = w_l - 2(\sqrt{h_m} - \sqrt{h_l}) \tag{3.35a}$$

$$w_m = w_r + 2(\sqrt{h_m} - \sqrt{h_r}) \tag{3.35b}$$

if we assume to have two rarefactions. Hence, we obtain two curves, one referring to all states that can be connected to u_l through a 1-rarefaction and analogously one that can be connected to u_r through a 2-rarefaction. Therefore, we can determine the middle state w_m as the intersection of both curves.

Note that we have two equations and two unknowns, w_m and h_m. Equating the right-hand sides and solving for h_m yields

$$h_m = \frac{1}{16}[w_l - w_r + 2(\sqrt{h_l} + \sqrt{h_r})]^2. \tag{3.36}$$

Then, we can compute w_m by using either the first or the second equation in (3.35).
In conclusion, we can determine

$$u_m = \begin{bmatrix} h_m \\ h_m w_m \end{bmatrix} \tag{3.37}$$

using (3.36) in (3.35).

3.5 Determining the Middle State

We already examined how the middle state can be determined provided the 1-wave and the 2-wave are of the same kind. Now, we also need to include the case that the 1-wave is a rarefaction and the 2-wave a shock. Note that in this situation, the middle state must lie on the integral curve for r^1 through u_l as well as on the Hugoniot-locus for u_r. Again, we can determine u_m as the intersection of both curves. This works analogously when the two waves change type.
Recall that in general, we do not know which combination of rarefactions and shock waves appears for given initial data.
Therefore, define the two functions

$$\phi_l(h) = \begin{cases} w_l + 2(\sqrt{h_l} - \sqrt{h}) & \text{if } h \le h_l, \\ w_l - (h - h_l)\sqrt{\frac{1}{2}(\frac{1}{h} + \frac{1}{h_l})} & \text{if } h > h_l \end{cases} \tag{3.38a}$$

and

$$\phi_r(h) = \begin{cases} w_r - 2(\sqrt{h_r} - \sqrt{h}) & \text{if } h \le h_r, \\ w_r + (h - h_r)\sqrt{\frac{1}{2}(\frac{1}{h} + \frac{1}{h_r})} & \text{if } h > h_r \end{cases}. \tag{3.38b}$$

The first case in each function returns the value for w such that (h, hw) can be connected to the respective given data u_r or u_l by a physically correct rarefaction wave while the second case returns the value for w through a shock wave. Therefore, for a given state h the first function $\phi_l(h)$ returns the value of w such that (h, hw) can be connected to $(h_l, h_l w_l)$ by a physically correct 1-wave and the second $\phi_r(h)$ the one that can be connected to $(h_r, h_r w_r)$ by a correct 2-wave.
We have to find the suitable state u_m such that it can be connected to u_r as well as to u_l in a physically correct manner. Therefore, we have to solve $\phi_l(h_m) = \phi_r(h_m)$. Note that this is a nonlinear equation. It can be solved by applying a nonlinear root-finder to the function $\phi(h) = \phi_l(h) - \phi_r(h)$.

4 Finite Volume Method

The finite volume method is a numerical method for solving hyperbolic conservation laws. It relies on the integral form of the conservation law and is therefore particularly suited for handling instabilities in the solution. The method is based on the idea of dividing the computational domain into grid cells. In each of these cells, the average value of the conserved variable u should be approximated. Since these values only change due to fluxes through the interfaces of the cells, a numerical flux function is applied to approximate them accurately.
For simplicity, we will only discuss the finite volume method in the case of one space dimension with a uniform grid where the grid cells are intervals of length Δx. For now, we also assume to have a fixed time step Δt. This approach can easily be extended to a multi-dimensional scenario or to the case of a nonuniform grid.

4.1 General Formulation

The formulation of finite volume methods for conservation laws is based on Chapter 4.1 in [5] and the preface of Chapter 10 in [8].
The general idea is based on dividing the spatial domain into *grid cells* which refer to intervals in the one-dimensional case. We define the points (x_j, t_n) by

$$x_j = j \cdot \Delta x, \quad j = \ldots, -1, 0, 1, 2, \ldots \tag{4.1}$$

$$t_n = n \cdot \Delta t, \quad n = 0, 1, 2, \ldots \tag{4.2}$$

Thus, we derive the grid points

$$x_{j+1/2} = x_j + \frac{\Delta x}{2} = \left(j + \frac{1}{2}\right)\Delta x. \tag{4.3}$$

The j-th grid cell is now defined by $\mathcal{C}_j = [x_{j-1/2}, x_{j+1/2})$ where the $\mathcal{C}_j$'s form a partition on the computational domain. Furthermore, let $u_j^n = u(x_j, t_n)$ denote the pointwise value of the actual solution. Then, the average value in every cell $\mathcal{C}_j$ at time t_n denoted by $\bar{u}_j^n$ is given by

$$\bar{u}_j^n = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n)dx. \tag{4.4}$$

Note that we cannot evaluate the interval exactly since the solution u is unknown. With the finite volume method we want to develop approximations $U_j^n \in \mathbb{R}^m$ to the average cell value (4.4).
Since the conservation law holds in every interval, the conserved variable $u(x,t)$ changes only due to fluxes at the boundaries of the respective cell. Therefore, we can describe the change of u by

$$\frac{d}{dt}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t)dx = f(u(x_{j-1/2}, t)) - f((x_{j+1/2}, t)). \tag{4.5}$$

We can use this expression to develop an explicit time-marching algorithm. That is, the approximation U_j^{n+1} depends only on the approximation of the cell average in the previous time step U_j^n. By integrating

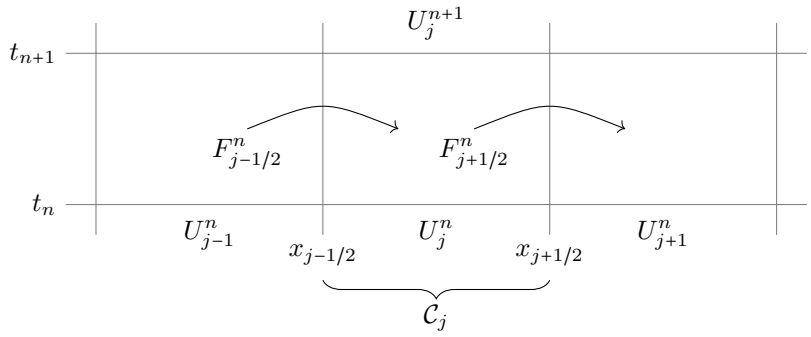


Figure 4.1: Scheme for updating the cell average U_j^n using the fluxes from adjacent cells.

over the interval $[t_n, t_{n+1}]$, we obtain

$$\int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_{n+1})dx - \int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n)dx = \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t))dt - \int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t))dt. \quad (4.6)$$

Dividing by Δx and using definition (4.4) yields the formula

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \frac{1}{\Delta x}\left[\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t))dt - \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t))dt\right]. \quad (4.7)$$

This describes how the solution for time step t_n can be evolved to t_{n+1}. However, in general, we cannot compute the right-hand side of this equation, since the required value of the true solution is unknown. Nevertheless, it suggests studying numerical methods of the form

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left[F_{j+1/2}^n - F_{j-1/2}^n\right] \quad (4.8)$$

where $F_{j\pm1/2}^n$ denotes the approximation to the average flux along $x_{j\pm1/2}$ as illustrated in Figure 4.1, i.e.

$$F_{j\pm1/2}^n \approx \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f(u(x_{j\pm1/2}, t))dt. \quad (4.9)$$

Note that we used the integral form of conservation laws to derive the update formula (4.8), which is the necessary form for modeling discontinuous solutions.
We can now assume that $F_{j-1/2}^n$ depends only on the average cell values U_{j-1}^n and U_j^n on each side of the interface since information propagates with finite speed of the eigenvalues in hyperbolic systems and we can choose the time step accordingly. In particular, we choose the time step such that it satisfies the *Courant–Friedrichs–Lewy condition (CFL condition)*

$$\frac{\Delta t}{\Delta x}\max_{i\in\mathbb{Z}}|f'(u_i)| = c \quad (4.10)$$

where the *CFL number* c must fulfill $c < 1$.
We obtain

$$F_{j-1/2}^n = \mathcal{F}(U_{j-1}^n, U_j^n) \quad (4.11)$$

where $\mathcal{F}$ is a *numerical flux function*. Using this in (4.8) leads to the formula

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\Big[\mathcal{F}(U_j^n, U_{j+1}^n) - \mathcal{F}(U_{j-1}^n, U_j^n)\Big]. \tag{4.12}$$

Even though the specific method depends on the choice of $\mathcal{F}$, in general, we receive a *three-point stencil*, that is, U_j^{n+1} is based only on the average values of the neighboring cells U_{j-1}^n, U_j^n and U_{j+1}^n at the previous time step.
Note that in semi-discrete form, we can consider finite volume methods as follows

$$\frac{d}{dt}u_i + \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = 0. \tag{4.13}$$

The choice of the numerical flux function crucially influences the quality of the resulting solution. In the numerical examples in Chapter 10, we compare a solution to the ELW scheme to the one obtained by the TeCNO scheme. We will see that there are visible improvements for the TeCNO scheme, it reduces spurious oscillations in the vicinity of shocks, even though both schemes are entropy stable and high-order accurate.
Remark that in order to achieve a higher order of accuracy in the numerical method (4.13), we also need a suitable method to integrate in time. For this purpose, we use Runge-Kutta and strong stability preserving Runge-Kutta methods. We explain them in detail in Chapter 10.

4.2 Pseudocode

This section describes our implementation of the finite volume method that can be found in the code accompanying this thesis.
Each run of the finite volume method requires a *Grid* that is initialized at the beginning of each run using the file *grid.py*. It fixes the following parameters:

- the order that is used in the specific run
- start, end of the computational domain including ghost cells on either side
- Δx for uniform grids
- list of grid points $x_{i-1/2}$, i.e. the respective points in the computational domain
- number of grid points and grid cells, with and without ghost cells

Since updating the cells at the right and left side of the computational domain requires adjacent cell states that can exceed the grid, we need to introduce p *ghost cells* on either side for a fixed order p. They are not updated by the finite volume method but by applying an appropriate boundary condition. For instance, if they are updated using periodic boundary conditions, the p ghost cells on the left side copy the values of the p rightmost cells (excluding the ghost cells on the right side) and vice versa.
Furthermore, note that some of the functions necessary in the finite volume method depend on the specific underlying conservation law, e.g. the computation of eigenvalues. The respective methods are defined in the function *initialize_fvm()*. The correct equation is fixed in *main.py* and chosen from the enumeration *EquationKind* given in *basics.equation_kind.py* that contains all implemented conservation laws, i.e. the linear advection equation (2.13), Burger's equation (2.15), the linear wave equation (2.16)

and the shallow water system (2.27).
The finite volume method is implemented in *solver.finite_volume_method.py* and works as follows:

Algorithm 1 *finite_volume_method()*

Performs one run of the finite volume method on intervals for order *order* and *nr_gridcells* grid cells up to time *end_time*. The initial function is given by *initial_function* and the kind of conservation law is determined by the enumeration *EquationKind*.

1: *initialize_fvm()* ▷ initializes e.g. the grid, cell averages and fluxes
2: t = *start_time*
3: **while** $t <$ *end_time* **do**
4: Δt = *compute.cfl_condition()*
5: *cell_averages* = *fvm_timestep()* ▷ update cell averages with respect to the required order
6: $t = t + \Delta t$
7: optional: plot current solution in folder *plots*
8: **end while**
9: cut off ghost cells
10: return cell_averages

At the beginning of a finite volume run, some parameters are initialized by the function *initialize_fvm()* in *basics.initialize.py*.

Algorithm 2 *initialize_fvm()*

Initializes the finite volume method run.

1: define numerical flux function ▷ e.g. TeCNO, Lax-Friedrichs or Godunov flux
2: initialize uniform grid
3: compute initial *cell_averages* depending on the initial function
4: pre-allocate memory for the fluxes
5: delete all current files in folder *plots*
6: plot initial *cell_averages* in *plots*
7: if necessary: define k and α_k in (7.2) ▷ e.g. for TeCNO or ELW scheme

5 First Order Entropy Conservative and Entropy Stable Methods

In this section, we consider a specific type of finite volume method satisfying a discrete version of the entropy admissibility criterion (2.54) for convex entropy pairs (η, q), so-called *entropy stable methods*. In 1971, Lax proved that the Lax-Friedrichs method is entropy stable and that if the method converges pointwise, the limit is the unique entropy solution [17]. In 1976, Harten et al. showed that in fact, all monotone schemes for scalar conservation laws are entropy stable [18]. Later, Osher et al. developed *E-schemes* which are designed such that they are entropy stable with respect to all convex entropy pairs [19]. However, these schemes are at most first-order accurate. In order to develop higher-order accurate entropy stable methods, Tadmor found a relation that guarantees that a method is *entropy conservative*, i.e. it satisfies a discrete version of the entropy equality (2.52) [20, 21]. They are automatically second-order accurate and can even be extended to higher accuracy as shown by LeFloch et al. in [22].
However, one issue that arises in entropy conservative methods is the lack of diffusion that causes oscillations close to discontinuities. Even though adding numerical diffusion can prevent these oscillations, it usually reduces the accuracy. We investigate this further in Chapter 7, but for now focus on first-order accurate schemes. Additionally, we assume $d = 1$, i.e. the conservation law is one-dimensional. However, the considerations can be generalized to more dimensions. This chapter is based on [7].

5.1 Conditions for Entropy Conservation and Stability

Definition 5.1. Let (η, q) be a convex entropy pair. We call the finite volume method (4.8) *entropy conservative* if computed solutions satisfy the *discrete entropy equality*, a discrete version of the entropy equality (2.52),

$$\frac{d}{dt}\eta(u_i) + \frac{Q_{i+1/2} - Q_{i-1/2}}{\Delta x} = 0 \tag{5.1}$$

where $Q_{i+1/2} = Q(u_{i-n+1}, \dots, u_{i+n})$ is a consistent $2n$-point numerical flux function, i.e. $Q(u, \dots, u) = q(u)$. Furthermore, we call the finite volume method *entropy stable* if the solutions satisfy the *discrete entropy inequality*, a discrete version of the entropy admissibility criterion (2.54),

$$\frac{d}{dt}\eta(u_i) + \frac{Q_{i+1/2} - Q_{i-1/2}}{\Delta x} \leq 0. \tag{5.2}$$

To obtain a discrete version of (2.56), sum (5.2) over $i \in \mathbb{Z}$ and integrate over $t \in [0, T]$. This yields

$$\sum_i \eta(u_i(T))\Delta x \leq \sum_i \eta(u_i(0))\Delta x. \tag{5.3}$$

Thus, analogous to the continuous version, the total amount of entropy of an entropy stable scheme decreases over time.
In the following, we discuss sufficient conditions for entropy conservation and stability. They play an important role when we finally construct entropy stable and entropy conservative schemes.
First, we derive a representation of entropy conservative fluxes that can then be extended to an entropy stable one. Afterwards, we investigate specific examples for scalar equations as well as for systems.
Define $v_i := v(u_i)$ and $\psi_i := \psi(v_i)$. Similar to the computations (2.53) we performed to derive the

entropy equality (2.52), we multiply the second summand in (4.13) by v_i:

$$\begin{aligned} v_i \cdot \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} &= \frac{(v_{i+1} + v_i) \cdot F_{i+1/2} - (v_i + v_{i-1}) \cdot F_{i-1/2} - (v_{i+1} - v_i) \cdot F_{i+1/2} - (v_i - v_{i-1}) \cdot F_{i-1/2}}{2\Delta x} \\ &= \frac{\overline{v}_{i+1/2} \cdot F_{i+1/2} - \overline{v}_{i-1/2} \cdot F_{i-1/2}}{\Delta x} - \frac{[\![v]\!]_{i+1/2} \cdot F_{i+1/2} + [\![v]\!]_{i-1/2} \cdot F_{i-1/2}}{2\Delta x} \\ &= \frac{(\overline{v}_{i+1/2} \cdot F_{i+1/2} - \overline{\psi}_{i+1/2}) - (\overline{v}_{i-1/2} \cdot F_{i-1/2} - \overline{\psi}_{i-1/2})}{\Delta x} \\ &\quad - \frac{([\![v]\!]_{i+1/2} \cdot F_{i+1/2} - [\![\psi]\!]_{i+1/2}) + ([\![v]\!]_{i-1/2} \cdot F_{i-1/2} - [\![\psi]\!]_{i-1/2})}{2\Delta x} \\ &= \frac{Q_{i+1/2} - Q_{i-1/2}}{\Delta x} - \frac{r_{i+1/2} - r_{i-1/2}}{2\Delta x} \end{aligned} \tag{5.4}$$

where

$$Q_{i+1/2} = Q(u_{i+1}, u_i) := \overline{v}_{i+1/2} \cdot F_{i+1/2} - \overline{\psi}_{i+1/2} \tag{5.5}$$

is the *numerical entropy flux* and

$$r_{i+1/2} := [\![v]\!]_{i+1/2} \cdot F_{i+1/2} - [\![\psi]\!]_{i+1/2} \tag{5.6}$$

is the *local entropy production.*
Hence, multiplying (4.13) by v_i yields

$$\frac{d}{dt}\eta(u_i) + \frac{Q_{i+1/2} - Q_{i-1/2}}{\Delta x} = \frac{r_{i+1/2} + r_{i-1/2}}{2\Delta x}. \tag{5.7}$$

If we now denote $v(s) := v_i + s[\![v]\!]_{i+1/2}$, we obtain

$$\begin{aligned} \int_0^1 [\![v]\!]_{i+1/2} \cdot \psi'(v(s))ds &= \int_0^1 \frac{d}{ds}\psi(v(s))ds \\ &= \psi(v(1)) - \psi(v(0)) \\ &= \psi(v_i + (v_{i+1} - v_i)) - \psi(v_i) \\ &= [\![\psi]\!]_{i+1/2}. \end{aligned} \tag{5.8}$$

Furthermore, we have

$$\begin{aligned} \psi'(v) &= f(u(v)) + v \cdot f'(u(v)) \cdot u'(v) - q'(u(v)) \cdot u'(v) \\ &= f(u(v)) + v \cdot f'(u(v)) \cdot u'(v) - \eta'(u(v)) \cdot f'(u) \cdot u'(v) \\ &= f(u(v)) \end{aligned} \tag{5.9}$$

where we first used the product and chain rule and then the definition of $q'(u(v))$ in (2.1).
Hence, we can write

$$\begin{aligned} r_{i+1/2} &= [\![v]\!]_{i+1/2} \cdot F_{i+1/2} - \int_0^1 [\![v]\!]_{i+1/2} \cdot \psi'(v(s))ds \\ &= \int_0^1 [\![v]\!]_{i+1/2} \cdot F_{i+1/2} - [\![v]\!]_{i+1/2} \cdot \psi'(v(s))ds \\ &= \int_0^1 [\![v]\!]_{i+1/2} \cdot (F_{i+1/2} - f(u(v(s))))ds. \end{aligned} \tag{5.10}$$

Using this in (5.7), we can deduce that the scheme is entropy stable according to Definition (5.1) if the integrand is non-positive everywhere.

Definition 5.2. Define $v(s) := v_i + s[\![v]\!]_{i+1/2}$ for all v_{i+1}, v_i and all convex entropy pairs (η, q). If the numerical flux F in (4.13) satisfies

$$[\![v]\!]_{i+1/2} \cdot (F_{i+1/2} - f(u(v(s)))) \leq 0, \quad \forall\ s \in [0,1], \tag{5.11}$$

we call the scheme (4.13) an *E-scheme*.
For scalar conservation laws, this can be written as

$$\operatorname{sgn}(u_{i+1} - u_i)(F_{i+1/2} - f(u)) \leq 0, \quad \forall\ u \in [u_i, u_{i+1}] \tag{5.12}$$

since

$$[\![v]\!]_{i+1/2} = \eta''(\xi_{i+1/2})[\![u]\!]_{i+1/2} \quad \text{for some } \xi_{i+1/2} \in [u_i, u_{i+1}] \tag{5.13}$$

which follows from the mean value theorem

$$\eta''(\xi_{i+1/2}) = v'(\xi_{i+1/2}) = \frac{v(u_{i+1}) - v(u_i)}{u_{i+1} - u_i}. \tag{5.14}$$

It has been shown by Osher in [19] that E-schemes for scalar conservation laws converge strongly to the unique entropy solution. However, since it has also been shown that they are at most first-order accurate, we cannot demand entropy stability with respect to all convex entropy pairs if we want the scheme to be entropy stable and high-order accurate.
Hence, Tadmor focused on entropy conservation and proved the following theorem in 1987 [21].

Theorem 5.1. *Consider (5.7) with $r_{i+1/2}$ given as in (5.6). Hence, if F satisfies*

$$[\![v]\!]_{i+1/2} \cdot F_{i+1/2} = [\![\psi]\!]_{i+1/2}, \quad \forall\ i \in \mathbb{Z}, \tag{5.15}$$

the scheme (4.13) is entropy conservative with numerical entropy flux Q given by (5.5).
Furthermore, for scalar conservation laws, i.e. $m = 1$, the scheme is second-order accurate in smooth regions of u.

Proof. The assumption above directly implies $r_{i+1/2} = 0$ and therefore the right-hand side in (5.7) is zero and the scheme is entropy conservative according to Definition 5.1.
To prove the accuracy requirement, we need to show that

$$\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = f(u(x))_x\big|_{x=x_i} + \mathcal{O}(\Delta x^2). \tag{5.16}$$

Assume $m = 1$. We can rewrite

$$\begin{aligned} F_{i+1/2} &= \frac{F_{i+1/2}[\![v]\!]_{i+1/2}}{[\![v]\!]_{i+1/2}} \\ &= \frac{[\![\psi]\!]_{i+1/2}}{[\![v]\!]_{i+1/2}} \\ &= \frac{1}{[\![v]\!]_{i+1/2}} \int_{v_i}^{v_{i+1}} \psi'(v)dv \\ &= \frac{1}{[\![v]\!]_{i+1/2}} \int_{v_i}^{v_{i+1}} f(u(v))dv \\ &\overset{(*)}{=} \frac{f(u_{i+1}) + f(u_i)}{2} - \frac{[\![v]\!]_{i+1/2}^2}{12} f''(\xi_{i+1/2}) \\ &= \frac{f(u_{i+1}) + f(u_i)}{2} - O(|[\![v]\!]_{i+1/2}|^2) \end{aligned} \tag{5.17}$$

with $\xi_{i+1/2} \in [v_i, v_{i+1}]$ where we used the trapezoidal rule to approximate the integral in $(*)$:

$$\int_a^b g(x)dx = (b-a)\frac{g(a)+g(b)}{2} - \frac{(b-a)^3}{12} g''(\xi) \quad \text{with some } \xi \in [a,b]. \tag{5.18}$$

Thus, we obtain

$$\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = \frac{f(u_{i+1}) - f(u_{i-1})}{2\Delta x} + \mathcal{O}(\Delta x^2) = f(u(x))_x\big|_{x=x_i} + \mathcal{O}(\Delta x^2). \tag{5.19}$$

□

Similarly, the scheme is entropy stable if $r_{i+1/2} \le 0$ for all $i \in \mathbb{Z}$, compare (5.7). We use this fact to derive a certain form of fluxes that leads to entropy stable methods. Let $F_{i+1/2}$ be any numerical flux and $\tilde{F}_{i+1/2}$ a flux satisfying (5.15). Then, considering $F_{i+1/2} - \tilde{F}_{i+1/2}$, we see that there exists a matrix $D_{i+1/2}$ such that

$$F_{i+1/2} = \tilde{F}_{i+1/2} - D_{i+1/2}[\![v]\!]_{i+1/2} \tag{5.20}$$

since

$$F_{i+1/2} - \tilde{F}_{i+1/2} = d_{i+1/2} = \tilde{d}_{i+1/2}[\![\psi]\!]_{i+1/2} = \tilde{d}_{i+1/2}\tilde{F}_{i+1/2}^T[\![v]\!]_{i+1/2} =: -D_{i+1/2}[\![v]\!]_{i+1/2} \tag{5.21}$$

where $\tilde{d}_{i+1/2} = \frac{1}{[\![\psi]\!]_{i+1/2}} d_{i+1/2}$ and $D_{i+1/2} = -\tilde{d}_{i+1/2}\tilde{F}_{i+1/2}^T$.
Hence, this results in the following theorem from [21].

Theorem 5.2. *Let $D_{i+1/2} \ge 0$. The, the scheme with flux (5.20) is entropy stable with numerical entropy flux*

$$Q_{i+1/2} = \overline{v}_{i+1/2} \cdot \tilde{F}_{i+1/2} - \overline{\psi}_{i+1/2} - \overline{v}_{i+1/2} \cdot D_{i+1/2}[\![v]\!]_{i+1/2}. \tag{5.22}$$

Proof. Since $\tilde{F}_{i+1/2}$ satisfies (5.15), we have

$$\begin{aligned} r_{i+1/2} &= [\![v]\!]_{i+1/2} \cdot \left(\tilde{F}_{i+1/2} - D_{i+1/2}[\![v]\!]_{i+1/2}\right) - [\![\psi]\!]_{i+1/2} \\ &= -[\![v]\!]_{i+1/2} \cdot D_{i+1/2}[\![v]\!]_{i+1/2} \le 0. \end{aligned} \tag{5.23}$$

Hence, the scheme is entropy stable according to (5.7). □

The theorem above showed that finite volume schemes with numerical fluxes of the form (5.20) where $D_{i+1/2}$ is a nonnegative matrix are always entropy stable. This suggests constructing entropy stable schemes by finding an entropy conservative flux $\tilde{F}$ fulfilling (5.15) and adding numerical diffusion. In the following sections, we first consider scalar conservation laws where the construction is simple. In the case of systems, however, this becomes more complicated.

5.2 Entropy Stable Methods for Scalar Equations

First, we consider scalar, one-dimensional conservation laws, i.e. $m = 1 = d$. Using condition (5.15), define the entropy conservative flux as

$$\tilde{F}_{i+1/2} = \begin{cases} \frac{[\![\psi]\!]_{i+1/2}}{[\![v]\!]_{i+1/2}} & \text{if } u_i \neq u_{i+1} \\ f(u_i) & \text{if } u_i = u_{i+1} \end{cases} \tag{5.24}$$

for a given convex entropy pair (η, q). Hence, each pair gives rise to a unique three-point entropy conservative scheme.
Let's consider specific examples in the following. As explained in Section 2.5, it is reasonable to use the square entropy $\eta(u) = \frac{u^2}{2}$ since all convex functions give rise to an entropy pair in the scalar case and the square entropy reduces the resulting entropy variable to $v(u) = u$.

Linear Advection Consider the linear advection equation (2.13). Using the square entropy leads to

$$q(u) = \int_0^u \eta'(s) f'(s)\, ds = \int_0^u as\, ds = \frac{au^2}{2} \tag{5.25}$$

and hence

$$\psi(u) = v \cdot f(u) - q(u) = au^2 - \frac{au^2}{2} = \frac{au^2}{2}. \tag{5.26}$$

Thus, the entropy conservative flux takes the form

$$\tilde{F}_{i+1/2} = \frac{a}{2} \frac{[\![u^2]\!]_{i+1/2}}{[\![u]\!]_{i+1/2}} = \frac{a}{2} \frac{(u_{i+1} - u_i)(u_{i+1} + u_i)}{u_{i+1} - u_i} = a\overline{u}_{i+1/2} \tag{5.27}$$

Burger's Equation Now, consider Burger's equation (2.15). Again, we use the square entropy which yields

$$q(u) = \int_0^u \eta'(s) f'(s)\, ds = \int_0^u s^2\, ds = \frac{u^3}{3} \tag{5.28}$$

and hence

$$\psi(u) = v \cdot f(u) - q(u) = \frac{u^3}{2} - \frac{u^3}{3} = \frac{u^3}{6}. \tag{5.29}$$

Thus, the corresponding entropy conservative flux is given by

$$\tilde{F}_{i+1/2} = \frac{u_{i+1}^2 + u_{i+1}u_i + u_i^2}{6} \tag{5.30}$$

To construct entropy stable methods, we will now use (5.20). Depending on the underlying conservation law, we can use either (5.27) or (5.30) as entropy conservative flux $\tilde{F}_{i+1/2}$. Recall from (5.13) that there exists a $\xi_{i+1/2}$ such that

$$\eta''(\xi_{i+1/2})[\![u]\!]_{i+1/2} = [\![v]\!]_{i+1/2}. \tag{5.31}$$

Hence, for all $D_{i+1/2} \geq 0$, we have $P_{i+1/2} := \eta''(\xi_{i+1/2})D_{i+1/2} \geq 0$ since η is convex. Therefore, we can rewrite (5.20) as

$$F_{i+1/2} = \tilde{F}_{i+1/2} - P_{i+1/2}[\![u]\!]_{i+1/2}. \tag{5.32}$$

Now, we may choose $P_{i+1/2}$ to be any nonnegative number. Two examples of different choices are the *Lax-Friedrichs diffusion*

$$P_{i+1/2} := \frac{1}{2}\max_{j\in\mathbb{Z}}(|f'(u_j)|) \tag{5.33}$$

and the *local Lax-Friedrichs diffusion*

$$P_{i+1/2} := \frac{1}{2}\max(|f'(u_i)|, |f'(u_{i+1}|). \tag{5.34}$$

In summary, we can now construct entropy stable methods for scalar conservation laws. Equation (5.15) provides us with an easy opportunity to compute entropy conservative fluxes and we gave two exemplary formulas, for the linear advection and Burger's equation. In the second half of this section, we discussed the diffusion term for scalar equations and presented two specific diffusion matrices (5.33) and (5.34) that can be used in the construction of entropy stable schemes.

5.3 Entropy Stable Methods for Systems

While the construction of entropy stable fluxes for scalar conservation laws was quite straightforward, it is more complicated for systems of equations. We start by investigating entropy conservative fluxes and constructing them for two different application areas. Then, we discuss how they can be extended to entropy stable schemes.

For $m = 1$, condition (5.15) could easily be solved for $\tilde{F}_{i+1/2}$. However, for $m > 1$, (5.15) is a scalar equation with m unknowns $\tilde{F}^1_{i+1/2}, \dots, \tilde{F}^m_{i+1/2}$ and can therefore in general not be uniquely solved.

Tadmor showed in [21] that a general solution $\tilde{F}_{i+1/2}$ is given as follows

$$\tilde{F}_{i+1/2} := \int_0^1 f(u(v(s)))\, ds \tag{5.35}$$

where again $v(s) := v_i + s[\![v]\!]_{i+1/2}$ for given u_{i+1}, u_i. This can be motivated by noting

$$\begin{aligned}[\![\psi]\!]_{i+1/2} &= \int_0^1 \frac{d}{ds}\psi(v(s))\, ds \\ &= (v(1) - v(0)) \cdot \int_0^1 \psi'(v(s))\, ds \\ &= [\![v]\!]_{i+1/2} \cdot \int_0^1 f(u(v(s)))\, ds\end{aligned} \tag{5.36}$$

and comparing this to (5.15).
Even though we now have a formula for explicitly calculating an entropy stable flux function, in practice the integral is difficult to evaluate even for simple problems.

Linear Wave Equation First, consider the linear wave equation (2.16). As mentioned in Section 2.5,

$$\eta(u) = \frac{1}{2}\left((u_1)^2 + (u_2)^2\right), \quad q(u) = au_1u_2 \tag{5.37}$$

is a corresponding convex entropy pair. Hence, the entropy variable is given by $v(u) = u$ and (5.35) leads to the entropy conservative flux

$$\tilde{F}_{i+1/2} = \int_0^1 A(u_i + s[\![u]\!]_{i+1/2})\, ds = \frac{1}{2}(Au_{i+1} + Au_i) \tag{5.38}$$

Shallow Water Equation Consider the shallow water system (2.27) and the entropy pair given in (2.58). Then, the entropy variable and the entropy potential are given in (2.59). Hence, (5.15) with $\tilde{F}_{i+1/2} = (\tilde{F}^1_{i+1/2}, \tilde{F}^2_{i+1/2})$ becomes

$$\tilde{F}^1 g[\![h]\!] - \frac{1}{2}\tilde{F}^1[\![w^2]\!] + \tilde{F}^2[\![w]\!] = \frac{1}{2}g[\![h^2w]\!]. \tag{5.39}$$

For simplicity, we omit the lower indices here. Using (1.2) yields

$$\begin{aligned}&\tilde{F}^1 g[\![h]\!] - \frac{1}{2}\tilde{F}^1\left(2[\![w]\!]\overline{w}\right) + \tilde{F}^2[\![w]\!] - \frac{1}{2}g\left(2[\![h]\!]\overline{h}\overline{w} + [\![w]\!]\overline{h^2}\right) = 0 \\ &\iff g[\![h]\!]\left(\tilde{F}^1 - \overline{h}\overline{w}\right) + [\![w]\!]\left(-\tilde{F}^1\overline{w} + \tilde{F}^2 - \frac{1}{2}g\overline{h^2}\right) = 0.\end{aligned} \tag{5.40}$$

Then, the flux

$$\tilde{F}_{i+1/2} = \begin{pmatrix} \tilde{F}^1_{i+1/2} \\ \tilde{F}^2_{i+1/2} \end{pmatrix} = \begin{pmatrix} \overline{h}_{i+1/2}\overline{w}_{i+1/2} \\ \overline{h}_{i+1/2}\overline{w}^2_{i+1/2} + \frac{1}{2}g\overline{h^2}_{i+1/2} \end{pmatrix} \tag{5.41}$$

satisfies (5.15) and hence, the corresponding finite volume scheme is entropy conservative.
However, choosing $F^1_{i+1/2}$ differently results in another entropy conservative flux, e.g.

$$\tilde{F}_{i+1/2} = \begin{pmatrix} \tilde{F}^1_{i+1/2} \\ \tilde{F}^2_{i+1/2} \end{pmatrix} = \begin{pmatrix} \overline{hw}_{i+1/2} \\ \overline{hw}_{i+1/2}\overline{w}_{i+1/2} + \frac{1}{2}g(h_{i+1}h_i) \end{pmatrix}. \tag{5.42}$$

Finally, we obtain the following lemma from [23, 24].

Lemma 5.1. *Consider the shallow water equations with corresponding entropy pair (2.58). Then, the*

finite volume scheme (4.13) with the numerical flux function (5.41) or (5.42) is second-order accurate and entropy conservative.

Since we are now capable of constructing entropy conservative fluxes for systems, we can use (5.20) and add numerical diffusion to obtain entropy stable methods.
However, we still do not know how exactly we need to choose the matrix $D_{i+1/2}$. Based on existing finite volume methods, we present two different possibilities for diffusion terms. However, in general, every positive matrix works.
If we extend (5.33) to systems, we obtain the *Lax-Friedrichs flux*

$$F_{i+1/2} := \frac{f(u_{i+1}) + f(u_i)}{2} - \frac{c_{i+1/2}}{2}[\![u]\!]_{i+1/2} \tag{5.43}$$

where $c_{i+1/2} = \max_{k=1,\dots,m}(|\lambda_k(u_{i+1})|, |\lambda_k(u_i)|)$ is the maximum of the absolute values of the eigenvalues of $f'(u)$ evaluated at $u = u_{i+1}$ and $u = u_i$.
Similarly, an extension of (5.34) is given by the *local Lax-Friedrichs flux*

$$F_{i+1/2} := \frac{f(u_{i+1}) + f(u_i)}{2} - \frac{1}{2}R_{i+1/2}|\Lambda_{i+1/2}|R^{-1}_{i+1/2}[\![u]\!]_{i+1/2} \tag{5.44}$$

where $R_{i+1/2} := (r_1(u_{i+1/2}), \dots, r_m(u_{i+1/2}))$ is the matrix of eigenvectors and

$$|\Lambda_{i+1/2}| := diag(|\lambda_1(u_{i+1/2})|, \dots, |\lambda_m(u_{i+1/2})|) \tag{5.45}$$

is the absolute value of the eigenvalue matrix of $f'(u_{i+1/2})$.
Both presented diffusion terms have the generic form

$$R_{i+1/2}A_{i+1/2}R^{-1}_{i+1/2}[\![u]\!]_{i+1/2} \tag{5.46}$$

where $A_{i+1/2}$ is a nonnegative matrix. However, we want to adapt this expression such that we obtain a diffusion term of the form (5.20).
We start by noting that heuristically, we can write

$$[\![u]\!]_{i+1/2} \approx \frac{du}{dv}(v_{i+1/2})[\![v]\!]_{i+1/2} \tag{5.47}$$

for some $v_{i+1/2}$. This is a heuristic generalization of the mean value theorem. Even though that usually does not hold for vector-valued functions, in many cases one can find an intermediate value fulfilling (5.47) with equality. In the other cases, we ignore the error of $O(|[\![v]\!]|)$.
To construct the diffusion coefficient in (5.20), we use the following lemma from [25].

Lemma 5.2. *Let $B = RAR^{-1}$ be a diagonalizable matrix and let $S \in \mathbb{R}^{m\times m}$ be a symmetric positive definite matrix such that BS is symmetric. Then, the columns of R can be scaled such that $S = RR^T$. In particular, it follows that $BS = RAR^T$.*

As mentioned in Section 2.5, $\frac{du}{dv}$ is a symmetric positive definite matrix and it right-symmetrizes $f'(u(v)) = R(u(v))\Lambda(u(v))R(u(v))^{-1}$. Hence, the columns of the eigenvector matrix $R(u(v))$ can be scaled such that $\frac{du}{dv} = R(u)R(u)^T$ for all $u = u(v)$. If we define $R_{i+1/2} := R(u(v_{i+1/2}))$, we can rewrite

(5.46) as

$$\begin{aligned} R_{i+1/2}A_{i+1/2}R_{i+1/2}^{-1}[\![u]\!]_{i+1/2} &\approx R_{i+1/2}A_{i+1/2}R_{i+1/2}^{-1}\frac{du}{dv}(v_{i+1/2})[\![v]\!]_{i+1/2} \\ &= R_{i+1/2}A_{i+1/2}R_{i+1/2}^{-1}R_{i+1/2}R_{i+1/2}^{T}[\![v]\!]_{i+1/2} \\ &= R_{i+1/2}A_{i+1/2}R_{i+1/2}^{T}[\![v]\!]_{i+1/2}. \end{aligned} \tag{5.48}$$

Hence, defining the diffusion matrix in (5.20) by

$$D_{i+1/2} := R_{i+1/2}A_{i+1/2}R_{i+1/2}^{T} \tag{5.49}$$

leads to an entropy stable method according to Theorem 5.2 since $A_{i+1/2}$ was assumed to be nonnegative. In the following, we present three different particular choices of this matrix inspired by the Lax-Friedrichs, the local Lax-Friedrichs and the Roe scheme. We denote the k-th eigenvalue of $f'(u)$ by $\lambda_k(u)$.

ELF: Define $A_{i+1/2} := \frac{1}{2}c\,I_m$ where $c := \max_{j\in\mathbb{Z}}(|\lambda_1(u_j)|,\dots,|\lambda_m(u_j)|)$. Thus, the diffusion matrix is given by

$$D_{i+1/2} = \frac{1}{2}cR_{i+1/2}R_{i+1/2}^{T}. \tag{5.50}$$

ELLF: Now, define $A_{i+1/2} := \frac{1}{2}c_{i+1/2}\,I_m$ where $c_{i+1/2} := \max_{k=1,\dots,m}(|\lambda_k(u_{j+1})|,|\lambda_k(u_j)|)$ now depends only on the adjacent cell averages. This yields the diffusion matrix

$$D_{i+1/2} = \frac{1}{2}c_{i+1/2}R_{i+1/2}R_{i+1/2}^{T}. \tag{5.51}$$

ERoe: Define $A_{i+1/2} := \frac{1}{2}|\Lambda_{i+1/2}|$ with $|\Lambda_{i+1/2}|$ as in (5.45). Hence, the diffusion matrix has the following form:

$$D_{i+1/2} = \frac{1}{2}R_{i+1/2}|\Lambda_{i+1/2}|R_{i+1/2}^{T} \tag{5.52}$$

Since the shallow water equations are the conservation law we mainly focus on in this thesis, we compute the explicit form of the scaled eigenvector matrix in this case. Recall that for this set of equations, the vector of conserved variables is given by

$$u = \begin{bmatrix} h \\ hw \end{bmatrix} \tag{5.53}$$

and that $u(v)$ denotes the inverse of $v(u)$, i.e. $u(v(u)) = u$. Since v is given by (2.59), this yields

$$u(v) = u\left((v_1,v_2)\right) = \frac{1}{g}\begin{bmatrix} v_1 + \frac{v_2^2}{2} \\ \left(v_1 + \frac{v_2^2}{2}\right)v_2 \end{bmatrix}. \tag{5.54}$$

Therefore, we obtain

$$\frac{du}{dv} = \frac{1}{g}\begin{bmatrix} 1 & v_2 \\ v_2 & \left(v_1 + \frac{v_2^2}{2}\right) + v_2^2 \end{bmatrix} = \frac{1}{g}\begin{bmatrix} 1 & w \\ w & w^2 + gh \end{bmatrix}. \tag{5.55}$$

Using these considerations results in the following lemma [23].

Lemma 5.3. *Consider the shallow water equation with corresponding entropy pair (2.58). Define*

$$u_v := \frac{du}{dv} = \frac{1}{g}\begin{bmatrix} 1 & w \\ w & w^2 + gh \end{bmatrix}. \tag{5.56}$$

If we define the scaled version of the eigenvector matrix of $f'(u)$ by

$$R = \frac{1}{\sqrt{2g}}\begin{bmatrix} 1 & 1 \\ w - \sqrt{gh} & w + \sqrt{gh} \end{bmatrix}, \tag{5.57}$$

we have $RR^T = u_v$.
For $u_i, u_{i+1} \in \mathbb{R}^+ \times \mathbb{R}$, we have

$$[\![u]\!]_{i+1/2} = (u_v)_{i+1/2}[\![v]\!]_{i+1/2} \tag{5.58}$$

where

$$(u_v)_{i+1/2} := \frac{du}{dv}(v(u_{i+1/2})) = \frac{1}{g}\begin{bmatrix} 1 & \overline{w}_{i+1/2} \\ \overline{w}_{i+1/2} & \overline{w}^2_{i+1/2} + g\overline{h}_{i+1/2} \end{bmatrix} \tag{5.59}$$

with $u_{i+1/2} = \left(\overline{h}_{i+1/2}, \overline{h}_{i+1/2}\overline{w}_{i+1/2}\right)$.

Thus, the entropy stable scheme for the shallow water system is the finite volume method (4.13) where we use an entropy stable flux of the form (5.20) with an entropy conservative flux given by (5.41) or (5.42), the diffusion matrix $D_{i+1/2}$ defined by (5.50), (5.51) or (5.52) and the eigenvector matrix $R_{i+1/2}$ is given by (5.57) evaluated at $h = \overline{h}_{i+1/2}$ and $w = \overline{w}_{i+1/2}$.

In summary, in this chapter, we investigated all components that are necessary for constructing entropy stable fluxes of the form (5.20). The entropy conservative flux can be computed by evaluating the integral (5.35). Furthermore, we presented explicit solutions for the linear wave and the shallow water equations. Afterwards, we introduced three different choices of diffusion matrices (5.50), (5.51) and (5.52) that can be used in the diffusion term. Finally, we are now able to apply (5.20) to obtain entropy stable fluxes for systems of equations.

6 Reconstruction

In this work, we want to derive high-order accurate, non-oscillatory and entropy stable methods. In order to achieve this, we employ *reconstruction methods.* In general, reconstructions are used to find piecewise approximations to a smooth function u from the given cell averages $\hat{u}_i$ of this function in every grid cell i.
In finite volume methods, they are often applied to the cell averages computed in every time step of the finite volume method to obtain well-balanced methods, i.e. methods that maintain a hydrostatic state over a long period of time, or a higher order of accuracy [6]. In this thesis, however, we use them in the derivation of the diffusion coefficient that smoothes the solution in the vicinity of shocks and therefore allows us to construct a non-oscillatory method. This chapter is based on [6].

Definition 6.1. Let $i \in \mathcal{I}$ be a grid index, $\overline{u}_j$ the cell-averaged states, $\mu \in \mathbb{N}$ odd and $x \in \mathcal{C}_i$.
A function

$$\begin{gathered}\mathcal{R}_i \in C(\mathcal{C}_i \times (\mathbb{R}^n)^\mu, \mathbb{R}^n), \\ \left(x, \overline{u}_{i-\frac{\mu-1}{2}}, \dots, \overline{u}_{i+\frac{\mu-1}{2}}\right) \mapsto u_i^{rec}(x) = \mathcal{R}_i\left(x, \overline{u}_{i-\frac{\mu-1}{2}}, \dots, \overline{u}_{i+\frac{\mu-1}{2}}\right)\end{gathered} \tag{6.1}$$

satisfying

$$\mathcal{R}_i(x, u, \dots, u) = u \tag{6.2}$$

is called *consistent reconstruction.* If $\mathcal{R}_i$ additionally satisfies

$$\frac{1}{\Delta x_i}\int_{\mathcal{C}_i} \mathcal{R}_i\left(x, \overline{u}_{i-\frac{\mu-1}{2}}, \dots, \overline{u}_{i+\frac{\mu-1}{2}}\right) dx = \overline{u}_i, \tag{6.3}$$

i.e. the reconstruction $\mathcal{R}_i$ maintains the average cell value in the respective cell, it is called a *conservative consistent reconstruction.*

As mentioned above, reconstructions are a common approach to obtaining methods of higher order or well-balanced methods. In this case, they are applied in the process of updating the cell averages across time steps in the finite volume method. Recall, that usually the update scheme is given by (4.12) where the numerical flux function depends on the neighboring cell averages. However, when we work with reconstructions they take the form

$$\overline{u}_j^{n+1} = \overline{u}_j^n - \frac{\Delta t}{\Delta x}\left[\mathcal{F}(\tilde{u}_{i+1}^-, \tilde{u}_i^+) - \mathcal{F}(\tilde{u}_i^-, \tilde{u}_{i-1}^+)\right] \tag{6.4}$$

where

$$\begin{aligned}\tilde{u}_i^+ &:= u_i^{rec}(x_{i+\frac{1}{2}}), \quad & \tilde{u}_{i+1}^- &:= u_{i+1}^{rec}(x_{i+\frac{1}{2}}), \\ \tilde{u}_{i-1}^+ &:= u_{i-1}^{rec}(x_{i-\frac{1}{2}}), \quad & \tilde{u}_i^- &:= u_i^{rec}(x_{i-\frac{1}{2}})\end{aligned} \tag{6.5}$$

with u_j^{rec} defined as in (6.1). Hence, the numerical flux function is now evaluated at the interface values of the reconstruction instead of the adjacent cell averages.
However, note that in this work, the reconstructions are used in the construction of high-order entropy stable methods as an extension to the jumps in the entropy variables evaluated at cell averages $[\![v]\!]_{i+1/2} = v_{i+1} - v_i$ (see Equation (5.20)). Hence, we are only interested in the jump of the reconstructed values

across the interfaces. In the following, the jump across $x_{i+1/2}$ is denoted by

$$\langle\!\langle v \rangle\!\rangle_{i+1/2} := v_{i+1}^- - v_i^+ = v_{i+1}(x_{i+1/2}) - v_i(x_{i+1/2}) \tag{6.6}$$

where $v_i(\cdot)$ denotes the reconstruction in cell $\mathcal{C}_i$ based on the surrounding entropy values $\{v_j\}_{j\in\mathbb{Z}}$.

Definition 6.2. We call a conservative consistent reconstruction *m-th order accurate* if

$$u(x) - \mathcal{R}_i\left(x, \overline{u}_{i-\frac{\mu-1}{2}}, \ldots, \overline{u}_{i+\frac{\mu-1}{2}}\right) = \mathcal{O}(h^m) \quad \text{for } x \in \mathcal{C}_i, \tag{6.7}$$

where u_j denotes the cell average of any function $u \in C^m(\mathcal{R}^n)$ in the i-th cell with $h > \Delta x_i$.

7 High-Order Entropy Stable Methods

In Chapter 5, we discussed the construction of first-order entropy stable methods. In this chapter however, we adapt them to obtain higher-order methods as required in the TeCNO scheme based on [7]. As before, we interpret an entropy stable flux as a composition of an entropy conservative flux and a diffusion term that depends on the entropy variables. Hence, we first investigate high-order entropy conservative fluxes. While their construction is simple, the adaption of the diffusion term is more involved. As in Chapter 5, we use different techniques for scalar conservation laws and systems.

7.1 High-Order Entropy Conservative Fluxes

The TeCNO scheme makes use of high-order entropy conservative fluxes. We have seen in Theorem 5.2 that F is an entropy stable numerical fluxes if it can be written in the form

$$F_{i+1/2} = \tilde{F}_{i+1/2} - D_{i+1/2}[\![v]\!]_{i+1/2} \tag{7.1}$$

where $\tilde{F}$ is an entropy conservative flux, $D_{i+1/2}$ any positive matrix and v the entropy variable.
The entropy conservative fluxes we considered earlier are only second-order accurate. However, we can follow the procedure developed by LeFloch, Mercier and Rohde in [22] and use linear combinations of these fluxes as building blocks to construct high-order schemes. They derived the following theorem:

Theorem 7.1. *For $k \in \mathbb{N}$ define the flux $\tilde{F}^{2k}$ by*

$$\tilde{F}^{2k}_{i+1/2} = \sum_{r=1}^{k} \alpha_r^k \sum_{s=0}^{r-1} \tilde{F}(u_{i-s}, u_{i-s+r}) \tag{7.2}$$

where $\tilde{F}(u_i, u_j)$ is a second-order accurate, entropy conservative flux and $\alpha_1^k, \dots, \alpha_r^k$ solve the k linear equations

$$\sum_{r=1}^{k} r\alpha_r^k = 1, \quad \sum_{r=1}^{k} r^{2s-1}\alpha_r^k = 0 \quad \text{for } s = 2, \dots, k. \tag{7.3}$$

Then, the finite volume scheme using this flux is $2k$-th order accurate, i.e. for sufficiently smooth solutions u we have

$$\frac{\tilde{F}^{2k}(u_{i-k+1}, \dots, u_{i+k}) - \tilde{F}^{2k}(u_{i-k}, \dots, u_{i+k-1})}{\Delta x} = f(u)_x\big|_{u=u_i} + O(\Delta x^{2k}). \tag{7.4}$$

Furthermore, it is entropy conservative, i.e. it satisfies the discrete entropy identity

$$\frac{d}{dt}\eta(u_i(t)) + \frac{\tilde{Q}^{2k}_{i+1/2} - \tilde{Q}^{2k}_{i-1/2}}{\Delta x} = 0 \tag{7.5}$$

with

$$\tilde{Q}^{2k}_{i+1/2} = \sum_{r=1}^{k} \alpha_r^k \sum_{s=0}^{r-1} \tilde{Q}(u_{i-s}, u_{i-s+r}). \tag{7.6}$$

Using this theorem, we employ the flux $\tilde{F}^{2k}$ as defined in (7.2) as the entropy conservative flux in the

construction of the entropy stable flux (5.20):

$$F_{i+1/2} = \tilde{F}^{2k}_{i+1/2} - D_{i+1/2}[\![v]\!]_{i+1/2}. \tag{7.7}$$

In the following, we investigate the *diffusion term* $D_{i+1/2}[\![v]\!]_{i+1/2}$. Recall that for $D_{i+1/2} \geq 0$ the scheme (5.20) is entropy stable according to Theorem 5.2. For $D \equiv 0$, we obtain a scheme with accuracy $2k$, but no numerical diffusion, hence the computed solution will exhibit oscillations around shocks. Even though this would require no additional computational effort, it is not suitable for practical implementations. Hence, we require a diffusion matrix $D_{i+1/2} > 0$. In the following, we first consider the *ELW scheme* which makes use of a particular choice of diffusion for scalar equations. Since this scheme however still produces spurious oscillations, we introduce reconstruction-based terms for scalar equations in Section 7.3 and for systems in Section 7.4.

7.2 Scalar ELW Scheme

If we choose

$$D_{i+1/2} = O([\![v]\!]^{p-1}_{i+1/2}) \tag{7.8}$$

and $2k \geq p$, this results in a scheme with truncation error

$$O(\Delta x^{2k} + |[\![v]\!]|^p) = O(\Delta x^p). \tag{7.9}$$

Hence, it is p-th order accurate but still has at least some numerical diffusion. One example of such a method is the ELW_p (Entropy stable Lax-Wendroff) scheme. Here, the diffusion matrix is given by

$$D_{i+1/2} = c_{i+1/2}|[\![v]\!]|^{p-1} \tag{7.10}$$

where

$$c_{i+1/2} = \frac{1}{2}\max(|f'(u_i)|, |f'(u_{i+1})|) \tag{7.11}$$

is determined by the local wave speeds. Hence, the ELW_p scheme is p-th order accurate and entropy stable according to Theorem 5.2. However, we now consider an example that points out the main drawback of using this scheme.

Example: Burger's Equation We consider Burger's equation (2.15) on the computational domain $[-1, 1]$ with periodic boundary conditions, a CFL number of 0.2 and initial data of the form

$$u(x, 0) = -\sin(\pi x). \tag{7.12}$$

The solution presented in Figure 7.1 results from numerically solving the equation with the ELW_3 scheme. We compare it to the exact solution obtained by applying the method of characteristics. Clearly, the ELW_3 solutions have large oscillations around the shock, which motivates the derivation and usage of high-order non-oscillatory reconstruction methods to prevent this phenomenon and obtain a smoother solution around discontinuities.

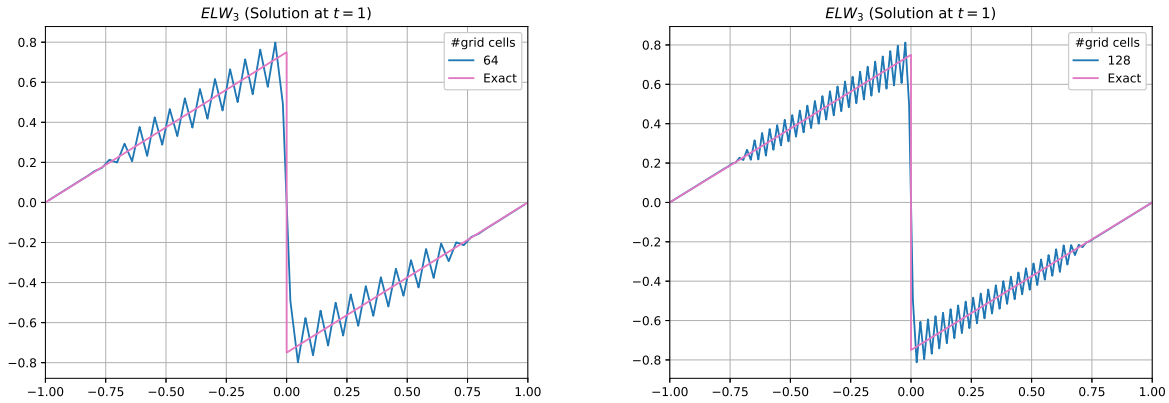


Figure 7.1: The exact and the ELW_3 solution to Burger's equation on 64 and 128 grid cells at $t = 1$.

7.3 Reconstruction Based Entropy Stable Schemes: Scalar Equations

In the following, we derive necessary conditions for the reconstruction to obtain an entropy stable scheme. First, we consider scalar equations and generalize this method to systems of equations in the next section.

As introduced in Chapter 6, reconstruction procedures are used to reconstruct a function from its given cell averages. We define a p-th order reconstruction as a map $\mathcal{R}_i : \{v_j\}_{j\in\mathbb{Z}} \to v_i(x)$ that produces a polynomial of degree $p-1$ and approximates the original function to order $O(\Delta x^p)$. Furthermore, we require the map to be conservative as defined in (6.3).

In the following, instead of considering the jump in the average values $[\![v]\!]_{i+1/2} := v_{i+1} - v_i$, we use the jump in the reconstructed values denoted by $\langle\!\langle v \rangle\!\rangle_{i+1/2} = v^-_{i+1} - v^+_i = v_{i+1}(x_{i+1/2}) - v_i(x_{i+1/2})$ where $v_k(x) = \mathcal{R}_k(\{v_j\}_{j\in\mathbb{Z}})$. Hence, when applying the reconstruction procedure, we are mainly interested in the values at the cell interfaces.

Therefore, we now consider fluxes of the form

$$F^p_{i+1/2} = \tilde{F}^{2k}_{i+1/2} - D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2}. \tag{7.13}$$

If $D_{i+1/2}$ is continuous with respect to the cell averages the truncation error is given by $O(\Delta x^{2k} + \Delta x^p)$. Hence, for $2k \geq p$, the scheme is formally p-th order accurate.

To see that the scheme is entropy stable, we use Lemma 5.2. First, we rewrite

$$D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} = D_{i+1/2}\frac{\langle\!\langle v \rangle\!\rangle_{i+1/2}}{[\![v]\!]_{i+1/2}}[\![v]\!]_{i+1/2} =: \tilde{D}[\![v]\!]_{i+1/2}. \tag{7.14}$$

Hence, if

$$\frac{\langle\!\langle v \rangle\!\rangle_{i+1/2}}{[\![v]\!]_{i+1/2}} \geq 0 \tag{7.15}$$

and therefore $D_{i+1/2}\frac{\langle\!\langle v \rangle\!\rangle_{i+1/2}}{[\![v]\!]_{i+1/2}} \geq 0$, we can apply Lemma 5.2 to prove that the system is entropy stable. Note that (7.15) can be written in the form

$$\langle\!\langle v \rangle\!\rangle_{i+1/2} = B_{i+1/2}[\![v]\!]_{i+1/2} \tag{7.16}$$

for a $B_{i+1/2} \geq 0$. Hence, the jump in the reconstructed values v_i^+, v_{i+1}^- must have the same sign as the jump in the original values v_i, v_{i+1}. We say that a reconstruction method satisfies the *sign property* if the reconstructed cell interface values satisfy (7.16). Later, we investigate the ENO reconstruction, a reconstruction procedure that fulfills the sign property and is, therefore, a suitable choice. However, note that the reconstruction procedure is only used in the diffusion term, not for reconstructing the cell averages in the entropy conservative flux $\tilde{F}^{2k}$.

7.4 Reconstruction Based Entropy Stable Schemes: Systems of Equations

Now, we want to generalize the construction of entropy stable schemes to systems of equations.
The higher-order entropy conservative flux (7.2) can easily be extended to multiple equations by considering a vector-valued function. Hence, in the following, we focus on the generalization of the diffusion term $D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2}$ since this construction is less trivial.
The first part of the diffusion term $D_{i+1/2}$ is no longer a scalar but a non-negative matrix. As in Section 5.3, we consider matrices of the form

$$D_{i+1/2} = R_{i+1/2} A_{i+1/2} R_{i+1/2}^T \tag{7.17}$$

where $R_{i+1/2}$ is an invertible and $A_{i+1/2} \geq 0$ a diagonal matrix.
For systems of equations, we obtain the following generalization of the sign property (7.16) [7]:

Lemma 7.1. *Let $D_{i+1/2}$ be given by*

$$D_{i+1/2} = R_{i+1/2} A_{i+1/2} R_{i+1/2}^T \tag{7.18}$$

with $R_{i+1/2}$ invertible, $A_{i+1/2} \geq 0$ diagonal. Furthermore, let $v_i(x)$ be a polynomial reconstruction of the entropy variables in the cell $\mathcal{C}_i$ such that there exists a diagonal matrix $B_{i+1/2} \geq 0$ with

$$\langle\!\langle v \rangle\!\rangle_{i+1/2} = (R_{i+1/2}^T)^{-1} B_{i+1/2} R_{i+1/2}^T [\![v]\!]_{i+1/2}. \tag{7.19}$$

Then the scheme with numerical flux (7.13) is entropy stable with numerical entropy flux

$$Q_{i+1/2}^p = \tilde{Q}_{i+1/2}^{2k} - \frac{1}{2}\overline{v}_{i+1/2} \cdot D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} \tag{7.20}$$

with $\tilde{Q}^{2k}$ defined as in (7.6).

Proof. Multiplying the finite volume scheme (4.13) by v_i^T yields

$$\begin{aligned} \frac{d}{dt}\eta(u_i) &= -\frac{\tilde{Q}_{i+1/2}^{2k} - \tilde{Q}_{i-1/2}^{2k}}{\Delta x} + \frac{v_i^T D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} - v_i^T D_{i-1/2}\langle\!\langle v \rangle\!\rangle_{i-1/2}}{\Delta x} \\ &= -\frac{Q_{i+1/2}^p - Q_{i-1/2}^p}{\Delta x} - \frac{[\![v]\!]_{i+1/2}^T D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} + [\![v]\!]_{i-1/2}^T D_{i-1/2}\langle\!\langle v \rangle\!\rangle_{i-1/2}}{2\Delta x}. \end{aligned} \tag{7.21}$$

Using (7.18), (7.19) and omitting indices for clarity, we obtain

$$[\![v]\!]^T D\langle\!\langle v \rangle\!\rangle = [\![v]\!]^T R A R^T R^{-T} B R^T [\![v]\!] = [\![v]\!]^T R A B R^T [\![v]\!] = (R^T [\![v]\!])^T A B (R^T [\![v]\!]) \geq 0 \tag{7.22}$$

since $A, B \geq 0$. Hence, the scheme is entropy stable. □

7.5 Reconstruction Along Scaled Entropy Variables

Using Lemma 7.1, we now want to construct a high-order accurate, entropy stable scheme for systems of equations.

In this work, we use the ENO reconstruction explained in Chapter 8 as the reconstruction procedure fulfilling the sign property necessary to construct high-order accurate entropy conservative fluxes. However, the application of the reconstruction is not trivial and therefore investigated in detail in this section.

First, assume the entropy values v_i, v_{i+1} and their reconstructed interface values v_i^+, v_{i+1}^- are given. Then, define the *scaled entropy variables*

$$w_i^\pm := R_{i\pm1/2}^T v_i, \quad \tilde{w}_i^\pm := R_{i\pm1/2}^T v_i^\pm. \tag{7.23}$$

With $\langle\!\langle \tilde{w} \rangle\!\rangle_{i+1/2} = \tilde{w}_{i+1}^- - \tilde{w}_{i+1/2}^+ = R_{i+1/2}^T \langle\!\langle v \rangle\!\rangle_{i+1/2}$ and $\langle\!\langle w \rangle\!\rangle_{i+1/2} = w_{i+1}^- - w_i^+ = R_{i+1/2}^T [\![v]\!]_{i+1/2}$, the sign equation for systems of equations (7.19) now takes the form

$$\langle\!\langle \tilde{w} \rangle\!\rangle_{i+1/2} = B_{i+1/2} \langle\!\langle w \rangle\!\rangle_{i+1/2}, \quad B_{i+1/2} \geq 0 \text{ diagonal}. \tag{7.24}$$

Hence, each component of $\tilde{w}_i$ satisfies the scalar sign property (7.16).

Note that the scaled entropy variables are centered at cell interfaces. Hence, to apply the ENO reconstruction (or any other reconstruction method $\mathcal{R}$ satisfying the sign property), we need to adapt the procedure accordingly. Fix $l \in \{1, \dots, m\}$ and denote the l-th component of w and $\tilde{w}$ by z and $\tilde{z}$, respectively. Thus, we have $\langle\!\langle z \rangle\!\rangle_{j+1/2} = z_{j+1}^- - z_j^+$. For each cell index i define $\alpha_i^i = z_i^-$ and inductively

$$\alpha_{j+1}^i = \alpha_j^i + \langle\!\langle z \rangle\!\rangle_{j+1/2}, \quad \text{for } j = i, i+1, \dots, \tag{7.25a}$$

$$\alpha_{j-1}^i = \alpha_j^i - \langle\!\langle z \rangle\!\rangle_{j-1/2}, \quad \text{for } j = i, i-1, \dots. \tag{7.25b}$$

Analogously, define $\beta_i^i = z_i^+$ and inductively

$$\beta_{j+1}^i = \beta_j^i + \langle\!\langle z \rangle\!\rangle_{j+1/2}, \quad \text{for } j = i, i+1, \dots, \tag{7.26a}$$

$$\beta_{j-1}^i = \beta_j^i - \langle\!\langle z \rangle\!\rangle_{j-1/2}, \quad \text{for } j = i, i-1, \dots. \tag{7.26b}$$

However, the computation of the $\alpha's$ can be simplified since

$$[\![\alpha^i]\!]_{j+1/2} = \alpha_{j+1}^i - \alpha_j^i = \alpha_j^i + \langle\!\langle z \rangle\!\rangle_{j+1/2} - \alpha_j^i = \langle\!\langle z \rangle\!\rangle_{j+1/2} \tag{7.27}$$

and

$$[\![\beta^i]\!]_{j+1/2} = \beta_{j+1}^i - \beta_j^i = \beta_j^i + \langle\!\langle z \rangle\!\rangle_{j+1/2} - \beta_j^i = \langle\!\langle z \rangle\!\rangle_{j+1/2}, \tag{7.28}$$

i.e. α and β retain the cell interface jumps of z.

Thus, it follows that

$$\alpha_j^{i+1} = \beta_j^i \tag{7.29}$$

since

$$\begin{aligned}\alpha_j^{i+1} - \beta_j^i &= \alpha_{j-1}^{i+1} - \langle\!\langle z \rangle\!\rangle_{j-1/2} - \beta_{j-1}^i + \langle\!\langle z \rangle\!\rangle_{j-1/2} \\ &= \alpha_{j-1}^{i+1} - \beta_{j-1}^i = \cdots = \alpha_{i+1}^{i+1} - \beta_{i+1}^i \\ &= z_{i+1}^- - \beta_i^i - \langle\!\langle z \rangle\!\rangle_{i+1/2} \\ &= z_{i+1}^- - z_i^+ - z_{i+1}^- + z_i^+ = 0\end{aligned} \tag{7.30}$$

where we first recursively applied the inductive definition of α and β and canceled out the cell interface jumps and then used the initial definitions $\alpha_{i+1}^{i+1} = z_{i+1}^-$ and $\beta_i^i = z_i^+$.
Finally, we apply the reconstruction procedure $\mathcal{R}$:
Define the reconstructions of α^i and β^i in the cell $\mathcal{C}_j$ as

$$\Phi_j^i(x) := \mathcal{R}_j(\{\alpha_k^i\}_{k\in\mathbb{Z}}) \quad \text{and} \quad \Psi_j^i(x) := \mathcal{R}_j(\{\beta_k^i\}_{k\in\mathbb{Z}}) \tag{7.31}$$

respectively. Hence, the reconstructed values at the cell interfaces are given by

$$\tilde{z}_i^- := \Phi_i^i(x_{i-1/2}) \quad \text{and} \quad \tilde{z}_i^+ := \Psi_i^i(x_{i+1/2}). \tag{7.32}$$

We repeat this procedure for each component $l \in \{1, \dots, m\}$ of $w_\pm^i$. Using the definitions of the scaled entropy variables (7.23) yields the reconstructed values

$$v_i^\pm := (R_{i\pm1/2}^T)^{-1}\tilde{w}_i^\pm. \tag{7.33}$$

Lemma 7.2. *Let the reconstruction procedure $\mathcal{R}$ satisfy the sign property (7.16). Then, the reconstructed values (7.33) satisfy (7.19).*

Proof. Using the definitions of the scaled entropy variables (7.23) yields

$$\begin{aligned}&\langle\!\langle v \rangle\!\rangle_{i+1/2} = (R_{i+1/2}^T)^{-1} B_{i+1/2} R_{i+1/2}^T [\![v]\!]_{i+1/2} \\ \iff &(R_{i+1/2}^T)^{-1}(\tilde{w}_{i+1}^- - \tilde{w}_i^+) = (R_{i+1/2}^T)^{-1} B_{i+1/2} R_{i+1/2}^T (v_{i+1} - v_i) \\ \iff &\langle\!\langle \tilde{w} \rangle\!\rangle_{i+1/2} = B_{i+1/2} \langle\!\langle w \rangle\!\rangle_{i+1/2}.\end{aligned} \tag{7.34}$$

From (7.29), we can deduce that the resulting functions of the reconstruction Φ^{i+1} is equal to Ψ^i. As above, we denote the l-th component of w_i and $\tilde{w}_i$ by z_i and $\tilde{z}_i$, respectively. We obtain

$$\begin{aligned}\langle\!\langle \tilde{z} \rangle\!\rangle_{i+1/2} = \tilde{z}_{i+1}^- - \tilde{z}_i^+ &= \Phi_{i+1}^{i+1}(x_{i+1/2}) - \Psi_i^i(x_i + 1/2) = \Psi_{i+1}^i(x_{i+1/2}) - \Psi_i^i(x_{i+1/2}) \\ &= \mathcal{R}_{i+1}(\{\beta_k^i\}_{k\in\mathbb{Z}}) - \mathcal{R}_i(\{\beta_k^i\}_{k\in\mathbb{Z}}).\end{aligned} \tag{7.35}$$

According to the sign property, this jump has the same sign as $\beta_{i+1}^i - \beta_i^i = \langle\!\langle z \rangle\!\rangle_{i+1/2}$, i.e. there exists a $b_{i+1/2} \geq 0$ such that

$$\langle\!\langle \tilde{z} \rangle\!\rangle_{i+1/2} = b_{i+1/2} \langle\!\langle z \rangle\!\rangle_{i+1/2} \tag{7.36}$$

and hence (7.34) holds. □

The procedure above can be performed to obtain the reconstructed values that are necessary for the computation of the diffusion coefficient in (7.13). In total, we obtain

$$D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} = R_{i+1/2} A_{i+1/2} R_{i+1/2}^T (R_{i+1/2}^T)^{-1} \langle\!\langle \tilde{w} \rangle\!\rangle_{i+1/2} = R_{i+1/2} A_{i+1/2} \langle\!\langle \tilde{w} \rangle\!\rangle_{i+1/2}. \tag{7.37}$$

Hence, even though the construction of the reconstructed values (7.33) suggests it, it is not necessary to compute the inverse of the matrix $R^T_{i+1/2}$. This is particularly useful since the computation of inverse matrices is computationally expensive.
However, we can simplify the computation even further by noting that the ENO reconstruction is linear with respect to constants, i.e.

$$\mathcal{R}_i(\{c + v_j\}_{j\in\mathbb{Z}}) = c + \mathcal{R}_i(\{v_j\}_{j\in\mathbb{Z}} \quad \text{for all } c \in \mathbb{R}. \tag{7.38}$$

Fix $k \in \mathbb{Z}$. Since

$$\beta^i_i - \beta^k_i = \beta^i_{i-1} + \langle\!\langle z \rangle\!\rangle_{i-1/2} - \beta^k_{i-1} - \langle\!\langle z \rangle\!\rangle_{i-1/2} = \beta^i_{i-1} - \beta^k_{i-1} = \cdots = \beta^i_j - \beta^k_j \tag{7.39}$$

and therefore

$$\beta^i_j = \beta^k_j + (\beta^i_i - \beta^k_i) \text{ for all } j \in \mathbb{Z} \tag{7.40}$$

we obtain with the linearity property (7.38)

$$\Psi^i_j(x) = \Psi^k_j(x) + (\beta^i_i - \beta^k_i) \quad \text{for all } j \in \mathbb{Z},\ x \in \mathcal{C}_j. \tag{7.41}$$

Now, the jump in the reconstructed variables can be written as

$$\tilde{z}^-_{i+1} - \tilde{z}^+_i = \Phi^{i+1}_{i+1}(x_{i+1/2}) - \Psi^i_i(x_{i+1/2}) = \Psi^i_{i+1}(x_{i+1/2}) - \Psi^i_i(x_{i+1/2}) = \Psi^k_{i+1}(x_{i+1/2}) - \Psi^k_i(x_{i+1/2}) \tag{7.42}$$

Hence, instead of recomputing the $\{\beta^i_j\}$'s for each grid cell $i \in \mathbb{Z}$, it suffices to compute the reconstructed values on only one fixed mesh $\{\beta^k_j\}_{j\in\mathbb{Z}}$.
In summary, we are now able to compute entropy stable fluxes of the form (5.20) for scalar equations as well as for systems provided we have a reconstruction procedure satisfying the sign property (7.16). In this work, we employ the ENO reconstruction procedure investigated in the following chapter.

8 ENO Reconstruction

The particular reconstruction method used in the TeCNO scheme is the *Essentially Non-Oscillatory (ENO)* reconstruction procedure introduced in 1987 by Harten et al. [26]. Given the cell averages, the ENO reconstruction produces a piecewise polynomial approximation to the original function. In contrast to other reconstruction methods, it avoids *Gibbs phenomenon*, i.e. it does not produce spurious oscillations near discontinuities.
Since the usual average operator

$$u \mapsto \overline{u} = \sum_i \overline{u}_i \mathbb{1}_{\mathcal{C}_i} \tag{8.1}$$

with $\overline{u}_i$ denoting the average value of u in each interval $\mathcal{C}_i = [x_{i-1/2}, x_{i+1/2})$

$$\overline{u}_i = \frac{1}{|\mathcal{C}_i|} \int_{\mathcal{C}_i} u(x)\, dx \tag{8.2}$$

is only first-order accurate, we use the ENO reconstruction to achieve a higher order of accuracy. This is achieved by generating a piecewise $(p-1)$-th order polynomial approximation $\tilde{u}$ of a function from its cell averages leading to a p-th order approximation.
Note that the ENO reconstruction fulfills the *sign property*, that is, the jump $\tilde{u}_{i+1}^- - \tilde{u}_i^+$ of the reconstructed values at the interface $x_{i+1/2}$ has the same sign as the jump $\overline{u}_{i+1} - \overline{u}_i$ in the underlying constant states. This is important since we want to apply this reconstruction technique later in the TeCNO scheme in the construction of entropy stable fluxes (see Chapter 7).
First, we impose the following requirements that should be fulfilled by the reconstruction:

1. It approximates u up to order p, i.e.

$$u(x) = \tilde{u}(x) + \mathcal{O}(h^p) \tag{8.3}$$

   where $h = \max_i |\mathcal{C}_i| = \Delta x$, since we only consider uniform grids. This requirement can be met by using standard polynomial interpolation. If the original function is p times continuously differentiable, the error in a polynomial is given by

$$u(x) - \tilde{u}(x) = \frac{u^{(p)}(\xi)}{(n+1)!} \prod_{i=0}^{p-1} (x - x_i) = \mathcal{O}(h^p) \tag{8.4}$$

   where the x_i's are the middle points of the cells C_i.

2. It conserves the underlying cell averages, i.e.

$$\overline{u}_i = \frac{1}{|\mathcal{C}_i|} \int_{\mathcal{C}_i} \tilde{u}_i(x)\, dx. \tag{8.5}$$

   We show that this can be done by performing the interpolation on the *primitive* $U(x) = \int_{-\infty}^{x} u(x)\, dx$ of u.

3. Lastly, it should avoid *Gibbs Phenomena*, i.e. is non-oscillatory around discontinuities. This criterion is met by interpolating over a variable stencil we investigate in detail in the following that depends on the current data.

Let $U(x) = \int_{-\infty}^{x} u(s)\, ds$ denote the primitive of u. Even though the underlying function u might be

unknown, we can evaluate U at the cell interfaces:

$$U_{i+1/2} = U(x_{i+1/2}) = \int_{-\infty}^{x_{i+1/2}} u(s)\, ds = \sum_{k=-\infty}^{i} \int_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} u(s)\, ds = \sum_{k=-\infty}^{i} |\mathcal{C}_k|\, \overline{u}_k \tag{8.6}$$

Furthermore, let $\tilde{U}$ be a function interpolating $\{U_{i+1/2}\}_{i\in\mathbb{Z}}$, i.e. $\tilde{U}(x_{i+1/2}) = U_{i+1/2},\ \forall j \in \mathbb{Z}$. Then, the derivative $\tilde{u}(x) := \tilde{U}'(x)$ fulfills the conservation requirement (8.5):

$$\begin{aligned}
\frac{1}{|\mathcal{C}_i|} \int_{\mathcal{C}_i} \tilde{u}(x)\, dx &= \frac{1}{|\mathcal{C}_i|} \int_{\mathcal{C}_i} \tilde{U}'(x)\, dx \\
&= \frac{1}{|\mathcal{C}_i|} \left[\tilde{U}(x_{i+1/2}) - \tilde{U}(x_{i-1/2})\right] \\
&= \frac{1}{|\mathcal{C}_i|} \left[U(x_{i+1/2}) - U(x_{i-1/2})\right] \\
&= \frac{1}{|\mathcal{C}_i|} \left[\sum_{k=-\infty}^{i} |\mathcal{C}_k|\, \overline{u}_k - \sum_{k=-\infty}^{i-1} |\mathcal{C}_k|\, \overline{u}_k\right] \\
&= \frac{1}{|\mathcal{C}_i|} \left[|\mathcal{C}_i|\, \overline{u}_i\right] \\
&= \overline{u}_i
\end{aligned} \tag{8.7}$$

Hence, we define $\tilde{U}_i(x)$ as the p-th order polynomial interpolating the $p+1$ point values $U_r, \ldots, U_{r+p}$. Note that the index r depends on the given data. The selection ensures the non-oscillatory behavior we required earlier.

When the underlying data is sufficiently smooth, the primitive U can be interpolated on any set of values $\{U_r, \ldots, U_{i-1/2}, U_{i+1/2}, \ldots U_{r+p}\}$ and still fulfill the accuracy requirement. Note that the set must contain the values $U_{i-1/2}, U_{i+1/2}$ in order to satisfy the conservation requirement.

However, note that we are interested in approximating piecewise smooth functions. To avoid spurious oscillations, the respective stencil must be chosen with particular caution. Therefore, we use a data-dependent index $r = r_i$ that depends on the smoothness of the underlying data.

8.1 Stencil Selection

Assume the $2p$ primitive values $U_{i-p+1/2}, \ldots, U_{i+p-1/2}$ are given. Then, we can compute the left stencil indices with the following algorithm:

Algorithm 3 Stencil selection

$r_i^0 = i - \frac{1}{2}$
for $k = 0 \to p-2$ **do**
  if $|U[x_{r_i^k-1}, \ldots, x_{r_i^k+k+1}]| < |U[x_{r_i^k}, \ldots, x_{r_i^k+k+2}]|$ **then**
    $r_i^{k+1} = r_i^k - 1$
  else
    $r_i^{k+1} = r_i^k$
  end if
end for

This is implemented in *stencil_selection()* in *recontruction.eno_reconstruction.py*.

Definition 8.1. First, compute the left stencil index r_i^p in each cell $\mathcal{C}_i$ according to Algorithm 3. Let $\tilde{U}_i(x)$ be given by the Newton representation of the p-th degree interpolant

$$\tilde{U}_i(x) = \sum_{k=-1}^{p-1} U[x_{r_i^k}, \ldots, x_{r_i^k+k+1}] \prod_{m=0}^{k} (x - x_{r_i^{k-1}+m}) \tag{8.8}$$

where we set $r_i^{-2} = r_i^{-1} = r_i^0 = i - 1/2$ Then, we define the *p-th order ENO reconstruction* $\tilde{u}(x) := \sum_i \tilde{u}_i(x)\mathbb{1}_{\mathcal{C}_i}(x)$ by

$$\tilde{u}_i(x) := \tilde{U}_i'(x) = \sum_{k=0}^{p-1} U[x_{r_i^k}, \ldots, x_{r_i^k+k+1}] \sum_{l=0}^{k} \prod_{\substack{m=0 \\ m\neq l}}^{k} (x - x_{r_i^{k-1}+m}). \tag{8.9}$$

8.2 ENO Sign Property

Theorem 8.1. *Let $p > 1$ be a fixed integer and $\tilde{u}(x) = \sum_i \tilde{u}_i(x)\mathbb{1}_{\mathcal{C}_i}(x)$ the p-th order ENO reconstruction with $\tilde{u}_i$ defined as in (8.9). Furthermore, define by $\tilde{u}_i^+ := \tilde{u}_i(x_{i+1/2})$ and $\tilde{u}_{i+1}^- := \tilde{u}_{i+1}(x_{i+1/2})$ the interface values of the reconstruction procedure at $x_{i+1/2}$. Then, the following sign property holds across all interfaces:*

$$\begin{aligned} &\text{if } \overline{u}_{i+1} - \overline{u}_i > 0, \quad \text{then } \tilde{u}_{i+1}^- - \tilde{u}_i^+ \geq 0, \\ &\text{if } \overline{u}_{i+1} - \overline{u}_i < 0, \quad \text{then } \tilde{u}_{i+1}^- - \tilde{u}_i^+ \leq 0, \\ &\text{if } \overline{u}_{i+1} - \overline{u}_i = 0, \quad \text{then } \tilde{u}_{i+1}^- - \tilde{u}_i^+ = 0. \end{aligned} \tag{8.10}$$

Therefore, the jump in the reconstructed interface values has the same sign as the jump in the original cell averages.

Furthermore, there is a constant c depending only on p and the mesh-ratio of adjacent grid cells $\max_{|j-i|\leq p}\left(\frac{|\mathcal{C}_{j+1}|}{|\mathcal{C}_j|}\right)$ such that

$$0 \leq \frac{\tilde{u}_{i+1}^- - \tilde{u}_i^+}{\overline{u}_{i+1} - \overline{u}_i} \leq c. \tag{8.11}$$

For a proof of this theorem, see Section 5.2 in [7].

9 TeCNO Scheme

In this chapter, we introduce the TeCNO scheme, a high-order accurate, entropy stable finite volume scheme. It was proposed by Fjordholm et al. in [1].
The TeCNO scheme is based on two essential components we investigated in detail in the previous chapters. First, we constructed high-order accurate, entropy conservative fluxes using linear combinations of second-order entropy conservative fluxes. Combining them with a diffusion matrix coupled with an interface jump of reconstructed entropy variables yields an entropy stable numerical flux function. Furthermore, we introduced the ENO reconstruction – a reconstruction procedure fulfilling the sign property necessary in the diffusion term. Now, we combine all these components to obtain an entropy stable method of arbitrary high order. In the next chapter, we verify the convergence rate numerically and also display the non-oscillatory property of the method.

9.1 TeCNO Scheme for Scalar Conservation Laws

Definition 9.1. For $p \in \mathbb{N}$, let $k \in \mathbb{N}$ be such that $2k \geq p$. Let $\tilde{F}$ be a two-point entropy conservative flux and define $\tilde{F}^{2k}$ by

$$\tilde{F}^{2k}_{i+1/2} = \sum_{r=1}^{k} \alpha_r^k \sum_{s=0}^{r-1} \tilde{F}(u_{i-s}, u_{i-s+r}). \tag{9.1}$$

Let $D_{i+1/2} \geq 0$ and $v_i^{\pm} := v_i(x_{i\pm 1/2})$ be the cell interface values of the p-th order ENO reconstruction $v_i(x)$ of the point values $\{v_j\}_{j\in\mathbb{Z}}$ where $v_i = v(u_i) = \eta'(u_i)$ denotes the entropy variable. Then, the *TeCNO$_p$ scheme* is the finite difference scheme

$$\frac{d}{dt}u_i + \frac{F^p_{i+1/2} - F^p_{i-1/2}}{\Delta x} = 0 \tag{9.2}$$

with flux

$$F^p_{i+1/2} = \tilde{F}^{2k}_{i+1/2} - D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2}. \tag{9.3}$$

Theorem 9.1. *The TeCNO$_p$ scheme for scalar conservation laws is (formally) p-th order accurate and entropy stable, with the numerical entropy flux*

$$Q^p_{i+1/2} = \tilde{Q}^{2k}_{i+1/2} - \overline{v}_{i+1/2} D_{i+1/2} \langle\!\langle v \rangle\!\rangle_{i+1/2} \tag{9.4}$$

Proof. First, note that in smooth regions, we have $\langle\!\langle v \rangle\!\rangle_{i+1/2} = O(\Delta x^p)$. Hence, the TeCNO$_p$ scheme is p-th order accurate since $\tilde{F}^{2k}_{i+1/2}$ is accurate to order $2k \geq p$. Furthermore, Theorem 5.2 shows the entropy stability of the scheme. Note that this requires the sign property of the ENO reconstruction since it ensures $\frac{\langle\!\langle v \rangle\!\rangle_{i+1/2}}{[\![v]\!]_{i+1/2}} \geq 0$ (see (7.14)). □

9.2 TeCNO Scheme for Systems of Equations

Definition 9.2. For $p \in \mathbb{N}$, let $k \in \mathbb{N}$ be such that $2k \geq p$. Let $\tilde{F}$ be a two-point entropy conservative flux and

$$D_{i+1/2} = R_{i+1/2} A_{i+1/2} R^T_{i+1/2} \tag{9.5}$$

with $R_{i+1/2}$ invertible and $A_{i+1/2} \geq 0$ a diagonal matrix. Furthermore, the values $v_i^{\pm}$ refer to the reconstructed entropy variables defined in (7.33) obtained by using the ENO reconstruction.

Then, the TeCNO$_p$ scheme is the finite volume scheme

$$\frac{d}{dt}u_i + \frac{F^p_{i+1/2} - F^p_{i-1/2}}{\Delta x} = 0 \tag{9.6}$$

with flux

$$F^p_{i+1/2} = \tilde{F}^{2k}_{i+1/2} - D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2} \tag{9.7}$$

where $D_{i+1/2}$ is a nonnegative matrix and $\langle\!\langle v \rangle\!\rangle_{i+1/2} = v^-_{i+1} - v^+_i$.

Theorem 9.2. *The TeCNO$_p$ scheme for systems of conservation laws is (formally) p-th order accurate and entropy stable with numerical entropy flux*

$$Q^p_{i+1/2} = \tilde{Q}^{2k}_{i+1/2} - \overline{v}_{i+1/2} \cdot D_{i+1/2}\langle\!\langle v \rangle\!\rangle_{i+1/2}. \tag{9.8}$$

Proof. The accuracy can be shown analogously to the scalar case. Furthermore, according to Theorem 7.1, the method is also entropy stable. □

9.3 Pseudocode

The following section contains pseudocode for computing the numerical fluxes with the TeCNO scheme.

Algorithm 4 *tecno_flux()*

Computes the numerical fluxes used in the update of the cell averages with the TeCNO scheme.

1: *diffusion*: computes the diffusion term using *compute.diffusion_scalar* or *compute.diffusion_system*
2: *fluxes = higher_order_entropy_conservative_flux - diffusion*

Note that the diffusion term in the case of scalar equations and systems uses the ELF-type diffusion matrix per default. This can be changed by using either the *ELLF()* or the *ERoe()* function from *basics.compute.py*. Furthermore, the diffusion type in *compute.diffusion_scalar* can be changed such that it uses the diffusion matrix for the ELW scheme.
In the next chapter, we numerically verify the convergence rate and non-oscillatory property of the TeCNO scheme.

10 Numerical Examples – Interval

In the following section, we present our numerical results obtained by the TeCNO scheme. Most importantly, we compute the numerical convergence rate and show that our method behaves the way we expect it from the formal rate of convergence. For this, we require a reference solution to compare our numerical results to. For certain problems such as the advection equation or initial Riemann data for the shallow water equations, we analytically compute an exact solution that serves as a reference solution. However, in general, we do not have access to a closed-form solution. Therefore, we need to employ reference solutions computed on a very fine grid using the local Lax-Friedrichs method

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(\tilde{F}_{i+1/2}^n - \tilde{F}_{i-1/2}^n) \tag{10.1a}$$

with

$$\tilde{F}_{i+1/2}^n = \frac{1}{2}(f_{i+1} + f_i) - \max_{j=1,\dots m}(\lambda_j(u_{i+1}), \lambda_j(u_i))\,(u_{i+1}^n - u_i^n). \tag{10.1b}$$

To increase the order of accuracy, we apply the minmod reconstruction procedure:

$$\mathcal{R}_{\text{minmod}_i}(x; u_{i-1}, u_i, u_{i+1}) = u_i + minmod(\sigma_i^L, \sigma_i^r)(x - x_i) \tag{10.2}$$

where

$$\sigma_i^L = 2\frac{u_i - u_{i-1}}{\Delta x_i + \Delta x_{i-1}} = \frac{u_i - u_{i-1}}{\Delta x_i} \tag{10.3a}$$

$$\sigma_i^R = 2\frac{u_{i+1} - u_i}{\Delta x_i + \Delta x_{i+1}} = \frac{u_{i+1} - u_i}{\Delta x_i} \tag{10.3b}$$

and

$$\text{minmod}(a,b) = \begin{cases} a, & \text{if } |a| \le |b| \text{ and } ab > 0 \\ b, & \text{if } |b| < |a| \text{ and} ab > 0 \\ 0, & \text{if } ab \le 0 \end{cases}. \tag{10.4}$$

Note that the reconstruction procedure is applied to each component of the cell averages separately for systems of equations. Finally, this results in a second-order accurate method.
For the integration in time we use strong stability preserving Runge-Kutta methods [27] for order two and three and for order four and five the respective Runge-Kutta method.
Strong stability preserving Runge-Kutta methods consist of subsequent applications and convex combinations of forward Euler steps. The second-order method takes the form

$$u^{(1)} = u^n - \frac{\Delta t}{\Delta x}L(u^n) \tag{10.5a}$$

$$u^{n+1} = \frac{1}{2}\left(u^n + u^{(1)} - \frac{\Delta t}{\Delta x}L(u^{(1)})\right) \tag{10.5b}$$

where

$$L(u)_i = F_{i+1/2} - F_{i-1/2} \tag{10.6}$$

denotes the flux at the interface i computed using the respective cell averages u.
Similarly, the third-order strong stability preserving Runge-Kutta scheme is given by

$$u^{(1)} = u^n - \frac{\Delta t}{\Delta x} L(u^n) \tag{10.7a}$$

$$u^{(2)} = \frac{3}{4}u^n + \frac{1}{4}\left(u^{(1)} - \frac{\Delta t}{\Delta x} L(u^{(1)})\right) \tag{10.7b}$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}\left(u^{(2)} - \frac{\Delta t}{\Delta x} L(u^{(2)})\right) \tag{10.7c}$$

with $L(u)$ as defined above.
Since the computation of a fourth- and fifth-order strong stability preserving Runge-Kutta method involves negative coefficients and is therefore computationally expensive, we use the standard fourth- and fifth-order Runge-Kutta method in this case.
They are all implemented in *update()* in the file *basics.time_integration.py*. If the passed order is not an integer in $\{1, \dots, 5\}$, the program raises an exception.
Furthermore, for the stability of the system, we require that the CFL-number

$$c = \frac{\Delta t}{\Delta x} \max_{i \in \mathbb{Z}} |f'(u_i)| \tag{10.8}$$

is less than one, i.e. we need to choose Δt accordingly. In the following numerical experiments, we consider the TeCNO scheme with Lax-Friedrichs type ELF (5.50) diffusion operator and a CFL number of 0.2, i.e.

$$\Delta t = \frac{0.2 \cdot \Delta x}{\max_{i \in \mathbb{Z}} |f'(u_i)|}. \tag{10.9}$$

Furthermore, recall that k in (7.2) needs to be chosen such that $2k \geq p$. In our numerical experiments, we mostly choose $k = p/2$ for p even and $k = (p+1)/2$ for p odd. Only for the dam-break shallow water problem we use $k = p$ since this smooths the solutions in the vicinity of the shock.

10.1 Experimental Order of Convergence

The *experimental order of convergence* is a measure for the efficiency of the numerical method in practice, i.e. we verify how quickly the numerical solutions $u_{\Delta x}$ for $\Delta x \to 0$ converge to the actual solution u. In this chapter, we describe how we can compute the numerical convergence rate (see [28]).
First, we compute the *relative error*

$$\varepsilon_{\Delta_x} = 100 \cdot \frac{||u_{\Delta_x} - u_{\text{ref}}||_{L_1}}{||u_{\text{ref}}||_{L_1}}. \tag{10.10}$$

The required reference solution u_{ref} is either given by the exact solution or by a numerical solution computed on a very fine grid where the number of grid cells should be approximately one order of magnitude larger than the highest one used for any $u_{\Delta x}$. Then, the experimental order of convergence can be computed using the formula

$$EOC_{\Delta x, \Delta y} = \frac{\log(\varepsilon_{\Delta x}) - \log(\varepsilon_{\Delta y})}{log(\Delta x) - log(\Delta y)}. \tag{10.11}$$

Since we use increasing powers of two as the number of grid cells, we always have $\Delta y = 2\Delta x$. Furthermore, even though the order depends on both solutions $u_{\Delta x}$ and $u_{\Delta y}$, we display it in the row corresponding to Δx, i.e. the higher number of grid cells.

10.2 Pseudocode

The following algorithm explains how we compute the convergence rate on an interval. All solutions need to be computed on 2^x grid cells in order to be comparable to each other.

Algorithm 5 *compute_convergence_rate_interval()*

Computes convergence rate on intervals for a fixed order. TeCNO and reference solutions must be computed on 2^x grid cells. For each solution that should be compared to the reference solution, the number of grid cells is stored in the list *grid_cells*.

load reference solution as *ref_sol*
norm_exact_solution ▷ compute l_1-norm of exact solution
errors = [ ] ▷ list to store l_1-errors
for $g \in$ *grid_cells* **do** ▷ i.e. all numerical solutions
    load TeCNO solution on g grid cells as *num_sol*
    step = length(ref_sol) / g
    ref_sol_new = [sum(ref_sol[j · step : j · step + step]) / step for $j \in \{1, \dots, g\}$ *]* ▷ reduce reference solution to g grid cells
    errors.append(norm_l1(*(ref_sol_new - num_sol))*$\cdot\Delta x$*)*
end for
convergence_rate_formula() ▷ compute convergence rate depending on the errors and the norm *norm_exact_solution*

10.3 Advection Equation

The first equation we consider is the linear advection advection equation (2.13) with $a = 1$. In this case, the exact solution (2.14) serves as reference solution.
The initial data is of the form

$$u_0 = \sin(\pi x) \tag{10.12}$$

and we use periodic boundary conditions.
The solutions computed with the TeCNO$_p$ scheme on different numbers of grid cells and the respective exact solution are displayed in Figure 10.1. Clearly, the solution is computed to a high degree of accuracy and improves as the number of grid cells increases for a fixed p. Furthermore, in Figure 10.2, we fix the number of grid cells $n = 128$ and can observe an increase in accuracy as the expected order of convergence p increases.
The error computed by the l_1-norm and the experimental orders of convergence for this problem can be found in Table 10.1. We can easily see that they are very close to the expected orders and improve for increasing numbers of grid cells. Hence, the TeCNO scheme behaves as expected for this advection problem.

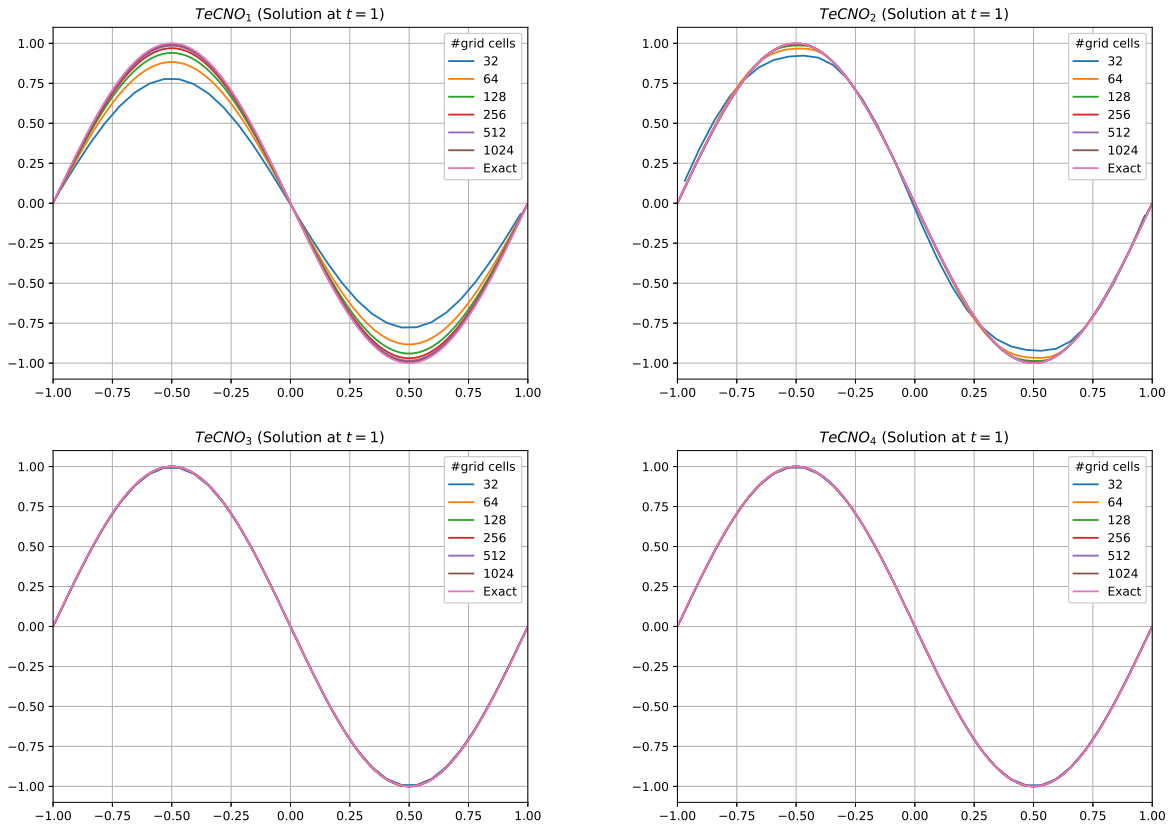


Figure 10.1: Solutions to the advection equation computed by using the TeCNO$_p$ scheme for different orders p and numbers of grid cells.

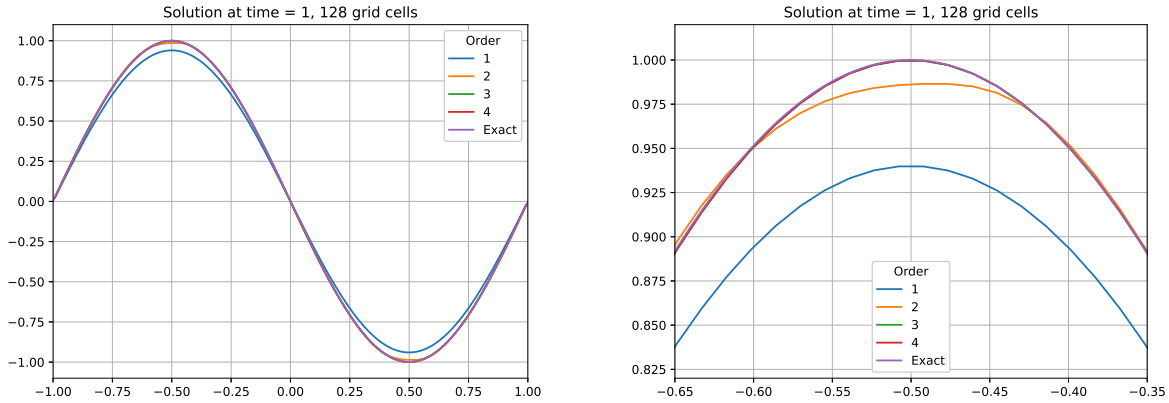


Figure 10.2: Solutions to the advection equation computed by the TeCNO$_p$ scheme for different orders p. *Right:* Close-up around $x = -0.5$.

	TeCNO$_1$		TeCNO$_2$		TeCNO$_3$		TeCNO$_4$		TeCNO$_5$	
n	Error	Order	Error	Order	Error	Order	Error	Order	Error	Order
32	2.78e-1	–	6.62e-2	–	2.52e-3	–	4.90e-4	–	1.93e-5	–
64	1.48e-1	0.914	1.99e-2	1.732	3.16e-4	2.992	3.42e-5	3.842	6.07e-7	4.990
128	7.62e-2	0.956	5.47e-3	1.864	3.96e-5	2.998	2.30e-6	3.892	1.899e-8	4.997
256	3.87e-2	0.978	1.46e-3	1.906	4.95e-6	3.000	1.55e-7	3.892	5.94e-10	4.999
512	1.95e-2	0.989	3.90e-4	1.906	6.18e-7	3.000	1.02e-8	3.923	1.86e-11	5.000
1024	9.78e-3	0.994	1.02e-4	1.931	7.73e-8	3.000	6.65e-10	3.941	5.80e-13	5.000
2048	4.90e-3	0.997	2.65e-5	1.946	9.66e-9	3.000	4.30e-11	3.950	1.82e-14	4.994

Table 10.1: Experimental order of convergence and the L_1-error of the TeCNO$_p$ scheme for the advection equation. n denotes the number of grid cells.

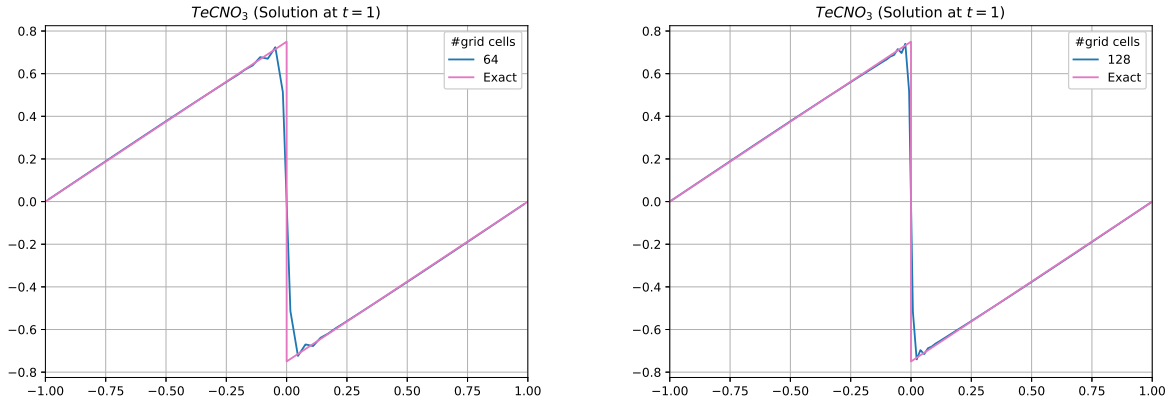


Figure 10.3: The exact and the TeCNO$_3$ solution to Burger's equation on 64 and 128 grid cells at $t = 1$.

10.4 Burger's Equation

The next scalar equation we consider is Burger's equation (2.15). To present the non-oscillatory property of the TeCNO scheme around shocks, we repeat the numerical example from 7.2 and expect a numerical solution that mimics the shock more accurately and prevents spurious oscillations. The result is displayed in Figure 10.3.

We can clearly see that the TeCNO scheme has a smoothing effect on the solution close to the discontinuity, there are only a few very small oscillations left at the shock. Especially in comparison to the ELW scheme solution in Figure 7.1, the TeCNO scheme performs very well around in the vicinity of the shock and can therefore be used to model discontinuities in the solution.

10.5 Linear Wave Equation

Now, we move on to investigating systems of equations and begin with the linear wave equation as defined in (2.16) with $a = 1$. We consider initial data of the form

$$u_0 = \begin{bmatrix} -\sin(\pi x) \\ 0 \end{bmatrix} \tag{10.13}$$

on the domain $[-1, 1]$ with periodic boundary conditions. We compute the solution up to $t = 0.4$. The resulting l_1-errors can be found in Table 10.2. Again, we see that the numerical convergence rates are very close to the expected ones.

	TeCNO_1		TeCNO_2		TeCNO_3		TeCNO_4		TeCNO_5	
n	Error	Order	Error	Order	Error	Order	Error	Order	Error	Order
32	1.48e-1	–	4.50e-2	–	1.22e-3	–	3.22e-4	–	9.36e-6	–
64	7.65e-2	0.951	1.35e-2	1.738	1.56e-4	2.970	2.27e-5	3.824	2.99e-7	4.967
128	3.89e-2	0.975	3.65e-3	1.889	1.97e-5	2.985	1.55e-6	3.871	9.47e-9	4.981
256	1.96e-2	0.988	9.87e-4	1.885	2.48e-6	2.992	1.05e-7	3.886	2.98e-10	4.992
512	9.85e-3	0.994	2.62e-4	1.915	3.11e-7	2.996	6.87e-9	3.936	9.33e-12	4.996
1024	4.94e-3	0.997	6.88e-5	1.929	3.89e-8	2.998	4.52e-10	3.925	2.92e-13	4.998

Table 10.2: Experimental order of convergence and the L_1-error of the TeCNO_p scheme for the linear wave equation. n denotes the number of grid cells.

The solution for increasing orders up to four with a fixed number of grid cells $n = 128$ and a corresponding close-up of the hump in each component can be found in Figure 10.4. Clearly, the solutions already nearly coincide with the exact solution, so the TeCNO scheme gives very good results even for small numbers of grid cells.

10.6 Shallow Water Equation

Finally, we consider the shallow water system (2.27). Since these equations are the most important ones in the context of this work, we investigate two different examples to display the different strengths of the TeCNO scheme.

Dam Break Problem: First, we consider the dam break problem with initial data given by (3.2) with $h_l = 2$ and $h_r = 1.5$. We have seen in Section 3.1.1 that this specific problem results in a right-going shock wave and a left-going rarefaction. In Figure 10.5, we observe that the numerical solution computed by the TeCNO scheme does not exhibit any oscillations around the discontinuity and remains smooth, similar to the results we have seen in Section 10.4. This shows again the non-oscillatory property of the TeCNO scheme.

Gaussian Wave: Next, we consider an initial wave that results in a smooth solution. The initial data is a Gaussian wave with zero velocity, i.e.

$$h(x,0) = a\ \exp(-(x-x_0)^2/(2s^2)) + h_0, \quad w(x,0) = 0 \tag{10.14}$$

where $a = 0.15$ denotes the amplitude of the wave, the position of the peak of the initial hump is at $x_0 = 0$, the standard deviation $s = 0.5$ controls the width of the curve and $h_0 = 2$ is the vertical displacement of the function.
As time evolves, the initial water hump splits into two waves propagating in opposite directions. For some initial Gaussian functions, the waves develop leading shock waves, however if the initial amplitude is very small compared to the water depth d, they propagate with basically unchanged shape at speed $\pm\sqrt{gh_0}$ and remain smooth.
The resulting water height on varying numbers of grid cells for order one and three are displayed in Figure 10.6. The reference solution is computed on 2^{15} grid cells.
The numerical convergence rates for the TeCNO_p scheme with $p = 1, 2, 3$, are given in Table 10.3.

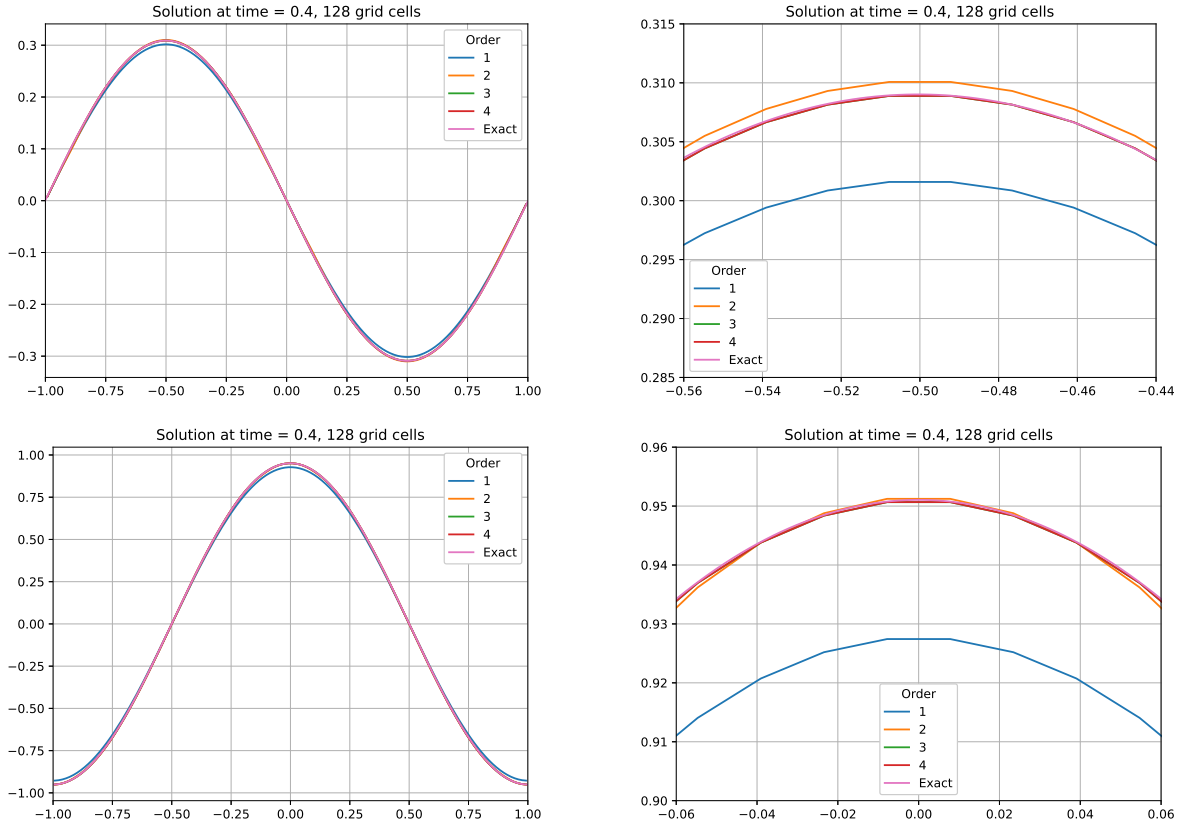


Figure 10.4: Solutions and close-ups to the linear wave equation computed by the TeCNO$_p$ scheme for different orders p on 128 grid cells. The first row shows the first component and the second row the second component.

	TeCNO$_1$		TeCNO$_2$		TeCNO$_3$	
n	Error	Order	Error	Order	Error	Order
32	1.68e-1	–	8.44e-2	–	2.96e-2	–
64	1.06e-1	0.666	3.12e-2	1.434	4.67e-3	2.666
128	6.14e-2	0.785	1.06e-2	1.559	7.79e-4	2.583
256	3.33e-2	0.884	3.52e-3	1.589	1.13e-4	2.779
512	1.74e-2	0.936	9.96e-4	1.821	1.56e-5	2.860
1024	8.90e-3	0.967	2.68e-4	1.891	2.11e-6	2.889

Table 10.3: Experimental order of convergence and the l_1-error of the TeCNO$_p$ scheme for the shallow water system with Gaussian initial data. n denotes the number of grid cells.

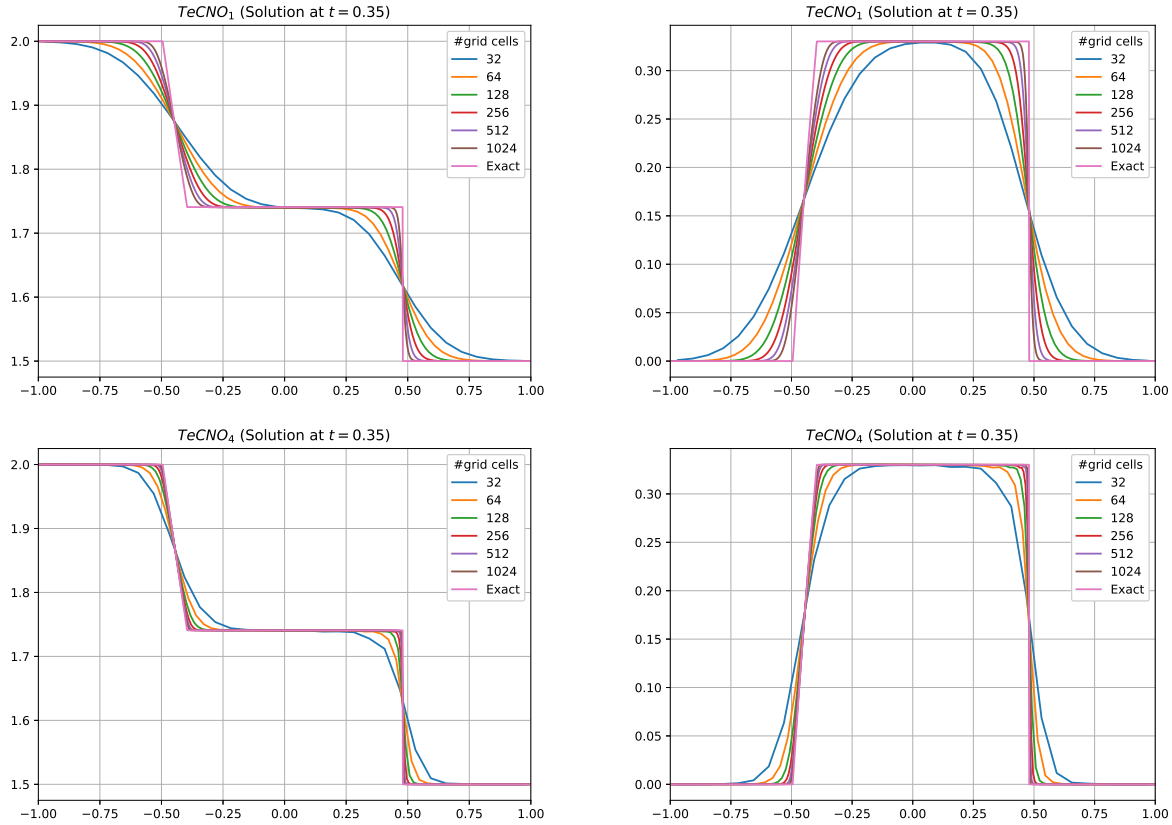


Figure 10.5: Solutions to the shallow water equations computed with the TeCNO_1 and TeCNO_4 scheme on different numbers of grid cells at $t = 0.35$. The left column displays the water height h and the right one the momentum hw.

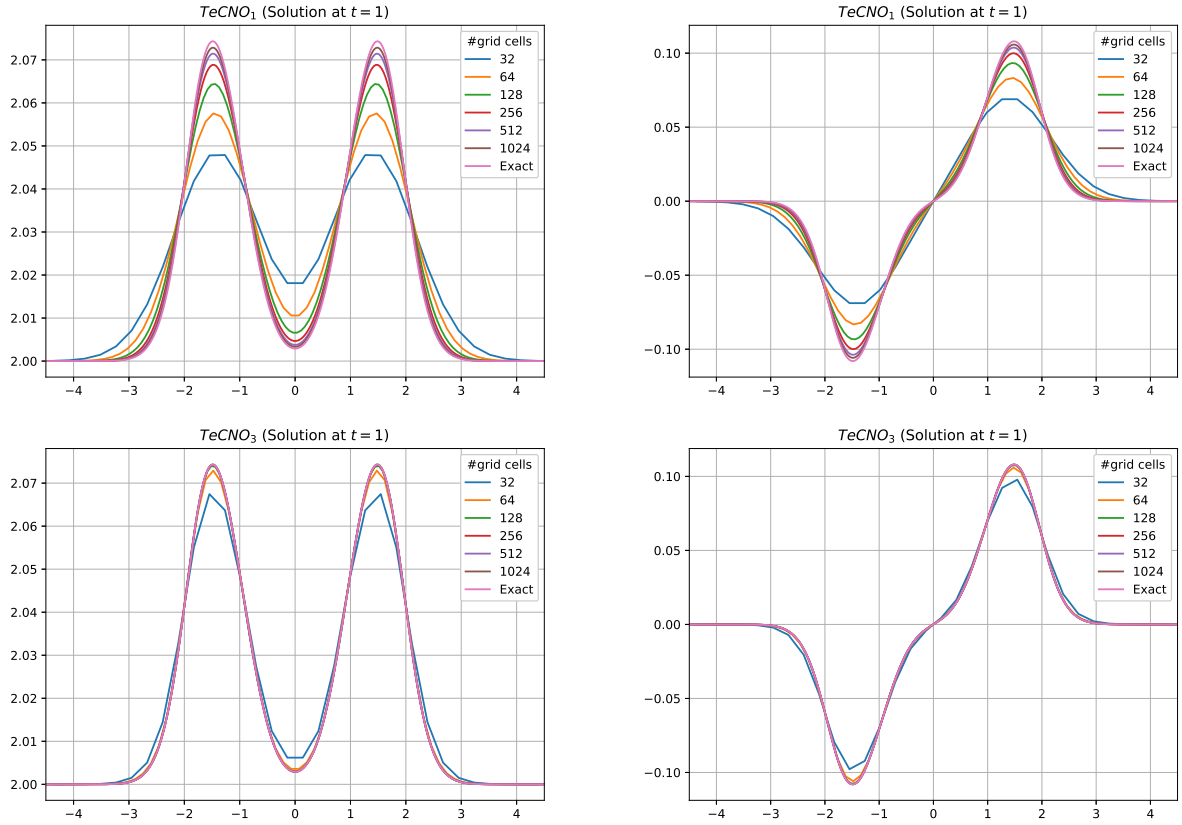


Figure 10.6: Solutions to the shallow water equations computed with the TeCNO_1 and TeCNO_3 scheme on different numbers of grid cells at $t = 1$. The left column displays the water height h and the right one the momentum hw.

11 Water Flow Through Networks

In the following, we model water flow through a network, i.e. a collection of straight channels connected by junctions. For this purpose, we employ the shallow water equations (2.27) and solve them with the TeCNO scheme for different orders. From a numerical point of view, the most straightforward treatment of a network would consist of computing solutions with a two-dimensional numerical solver. However, this procedure is computationally expensive and can be simplified by exploiting the specific structure of the problem. Since the network consists of channels separated by junctions, we can solve these two problems independently. The main difficulty here is the treatment of the junctions. Throughout the channels, we can use a one-dimensional numerical solver to compute solutions. For this purpose, we use the TeCNO scheme defined in Chapter 9. To update the cell averages across the junctions, however, we use the coupling conditions published by Briani et al. in [2] that are derived in this chapter.
Note that the following considerations require a *fluvial regime*, i.e. we expect $|w| < \sqrt{gh}$. Hence, the eigenvalues of the shallow water system (2.33) fulfill

$$\lambda_1 < 0, \quad \lambda_2 > 0 \tag{11.1}$$

which results in two waves propagating in opposite directions.
In this chapter, we first describe the overall setting of the networks we consider in this thesis. Then, we derive the coupling conditions that are used to compute the states across the junctions. In the next chapter, we present the numerical results for the solution throughout a network obtained by coupling the conditions at the junctions with the one-dimensional TeCNO scheme in the channels.

11.1 Definition of the Network

In this work, we only consider networks consisting of straight channels that are connected by junctions. Furthermore, we assume that each junction consists of one incoming and one or two outgoing channels. For simplicity, we only define networks consisting of one junction in the following. However, by connecting several of these simple networks – and possibly adapting the reference system – we can construct longer and more complicated structures.
First, we define the terminology for the junctions that is used throughout this thesis. For a moment, assume the channels are one-dimensional, i.e. they have width zero and that we have three channels meeting at the junction.
We fix the intersection of the channels at the origin of the reference system. The incoming channel is labeled *Channel 1* and is assumed to be parallel to the x-axis. On the other hand, the outgoing channels are labeled *Channel 2* and *Channel 3* and they are at the angles ϕ and θ to the x-axis, respectively. Moreover, we assume $\theta \geq 0$ and $\phi \leq 0$. Therefore, we obtain the geometry depicted in Figure 11.1.
However, to interpret the channels as two-dimensional objects, they need to be equipped with a certain width. Then, the one-dimensional setup described above serves as the skeleton of the two-dimensional setting. Let $2s_k$, $k = 1, 2, 3$ denote the width of each channel. Furthermore, we define the intersection point of the channel walls of channel i and j as P_{ij}, $i = 1, 2$, $j = 2, 3$, $j \neq i$. These points form a triangle at the junction as can be seen in Figure 11.2 and we will refer to that triangle as T and to its edges as e_k, $k = 1, 2, 3$. The outward-pointing normals are denoted by n_k. Note that they are not necessarily parallel to the respective channel, this only happens in the special case when $\theta = -\phi$.
Moreover, let I_k, $k = 1, 2, 3$ denote the interface separating channel k from the junction, i.e. the point

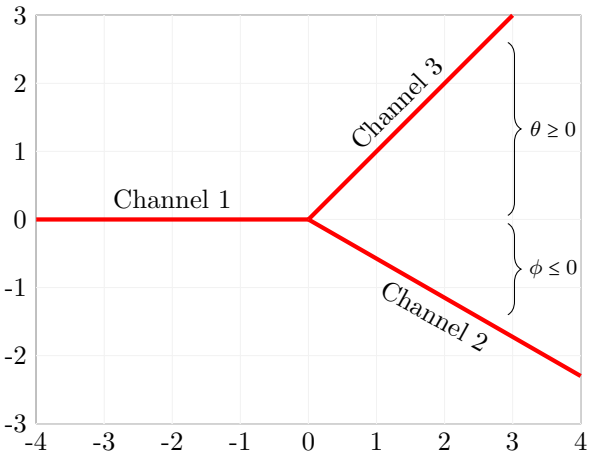


Figure 11.1: One-dimensional setup for a channel junction where $\theta = \frac{\pi}{4}$ and $\phi = -\frac{\pi}{6}$

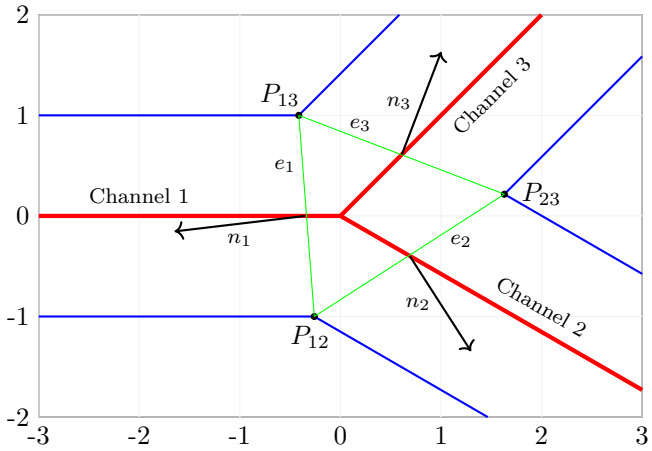


Figure 11.2: Two-dimensional setup for a channel junction

Parameters are $s_1 = s_2 = s_3 = 1$, $\theta = \frac{\pi}{4}$ and $\phi = -\frac{\pi}{6}$

where the one-dimensional channel intersects e_k. Furthermore, we refer to the state variable at I_k facing the channel k as $U_k^\star$ and to the one facing the junction as U_k, $k = 1, 2, 3$. Note that the states in the channel can be computed with the one-dimensional numerical solver. Hence, the purpose of the junction Riemann solver is to compute the junction states U_k given $U_k^\star$.

11.2 Junction Solver

In the following, we derive the conditions at the junctions that enable us to compute the state variables U_k for each channel. Note that each of these state variables is composed of the height and velocity, i.e. h_k and w_k, which leads to six unknowns at a junction. Hence, we need six equations in order to obtain a unique solution to the system. The first three equations result from Riemann problems across the interfaces I_k while the other three are derived from conservation of mass and momentum across the junction.

Recall from Chapter 3 that the intersection of the equations (3.38) defines the intermediate state between

two waves under fluvial flow. Hence, the first three equations in the junction system are

$$w_1 = \phi_l(h_1; U_1^\star) \tag{11.2a}$$
$$w_2 = \phi_r(h_2; U_2^\star) \tag{11.2b}$$
$$w_3 = \phi_r(h_3; U_3^\star) \tag{11.2c}$$

To derive the remaining three equations, we exploit the two-dimensional configuration of the channels and introduce the following notation:

1. h denotes the water height
2. $\mathbf{w} = (w_x, w_y)$ denotes the two-dimensional velocity
3. $\mathbf{q} = h\mathbf{w}$ denotes the two-dimensional discharge
4. $\mathbf{q}_k = h_k(w_{x,k}, w_{y,k})$, $k = 1, 2, 3$ denotes the average discharge at the interface I_k

Recall that the shallow water equations in two dimensions are given in (2.41).
If $\mathbf{n}$ denotes the outward-pointing normal of ∂T conservation of mass leads to

$$\int_{\partial T} \mathbf{q} \cdot \mathbf{n} = 0. \tag{11.3}$$

Similarly, the conservation of the two components of momentum yields

$$\int_{\partial T} \left(w_x \mathbf{q} + \frac{1}{2} g h^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \mathbf{n} = 0, \tag{11.4a}$$
$$\int_{\partial T} \left(w_y \mathbf{q} + \frac{1}{2} g h^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \mathbf{n} = 0. \tag{11.4b}$$

Recall that the boundary ∂T of the triangle is composed of the edges e_k, $k = 1, 2, 3$, and $\mathbf{q}_k$ denotes the average value of the discharge along each edge. We define the length of e_k by l_k. Therefore, we obtain the three conservation equations

$$\sum_{k=1,2,3} l_k \mathbf{q}_k \cdot \mathbf{n}_k = 0 \tag{11.5a}$$
$$\sum_{k=1,2,3} l_k \left(w_{x,k} \mathbf{q}_k + \frac{1}{2} g h_k^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \mathbf{n}_k = 0 \tag{11.5b}$$
$$\sum_{k=1,2,3} l_k \left(w_{y,k} \mathbf{q}_k + \frac{1}{2} g h_k^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \mathbf{n}_k = 0. \tag{11.5c}$$

Note that the discharge in the three channels can be written as

$$\mathbf{q}_1 = q_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{q}_2 = q_2 \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix}, \quad \mathbf{q}_3 = q_3 \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{11.6}$$

where $q_k = \|\mathbf{q}_k\|$.
Using this and recalling $w_k = \frac{q_k}{h_k}$, we can rewrite (11.5a) as

$$l_1 h_1 w_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \mathbf{n}_1 + l_2 h_2 w_2 \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix} \cdot \mathbf{n}_2 + l_3 h_3 w_3 \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \cdot \mathbf{n}_3 = 0, \tag{11.7a}$$

(11.5b) as

$$\begin{aligned} &l_1 \left(h_1 w_1^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{2} g h_1^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \mathbf{n}_1 + l_2 \left(h_2 w_2^2 \cos(\phi) \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix} + \frac{1}{2} g h_2^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \mathbf{n}_2 \\ &+ l_3 \left(h_3 w_3^2 \cos(\theta) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} + \frac{1}{2} g h_3^2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \mathbf{n}_3 = 0 \end{aligned} \tag{11.7b}$$

and (11.5c) as

$$\begin{aligned} &l_1 \left(\frac{1}{2} g h_1^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \mathbf{n}_1 + l_2 \left(h_2 w_2^2 \sin(\phi) \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix} + \frac{1}{2} g h_2^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \mathbf{n}_2 \\ &+ l_3 \left(h_3 w_3^2 \sin(\theta) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} + \frac{1}{2} g h_3^2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \mathbf{n}_3 = 0. \end{aligned} \tag{11.7c}$$

In order to work with these equations, we have to derive the formulas for computing the normals $\mathbf{n}_k$ to the respective edges e_k and the lengths l_k, $k = 1, 2, 3$.
The equations for the one-dimensional channels as depicted in Figure 11.1 are given by

$$y_1 = t_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{11.8a}$$

$$y_2 = t_2 \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix}, \tag{11.8b}$$

$$y_3 = t_3 \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, \tag{11.8c}$$

with $t_k \in \mathbb{R}$ for $k = 1, 2, 3$. Hence, the walls of the channels are the lines parallel to (11.8) at a distance of s_k, i.e.

$$y_1^{\pm} = t_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \pm s_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{11.9a}$$

$$y_2^{\pm} = t_2 \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix} \pm s_2 \begin{pmatrix} -\sin(\phi) \\ \cos(\phi) \end{pmatrix}, \tag{11.9b}$$

$$y_3^{\pm} = t_3 \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \pm s_3 \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix}. \tag{11.9c}$$

As can be seen in Figure 11.2, the points P_{ij} are obtained by intersecting the respective channel walls. Therefore, intersecting the walls y_1^+ and y_3^+ yields the point P_{13}, i.e.

$$\begin{aligned} y_1^+ = y_3^+ &\iff t_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + s_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = t_3 \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} + s_3 \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix} \\ &\iff \left\{ \begin{array}{l} t_1 = t_3 \cos(\theta) - s_3 \sin(\theta) \\ s_1 = t_3 \sin(\theta) + s_3 \cos(\theta) \end{array} \right\}. \end{aligned} \tag{11.10}$$

Assume for the moment that $\theta \neq 0$. Then, we obtain

$$t_3 = \frac{s_1 - s_3 \cos(\theta)}{\sin(\theta)} \tag{11.11}$$

that can now be inserted in y_3^+ to derive the intersection point

$$\begin{aligned} P_{13} &= \frac{s_1 - s_3 \cos(\theta)}{\sin(\theta)} \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} + s_3 \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix} \\ &= \begin{pmatrix} \frac{s_1 \cos(\theta)}{\sin(\theta)} - s_3 \left(\frac{\cos^2(\theta) + \sin^2(\theta)}{\sin(\theta)} \right. \\ s_1 - s_3 \cos(\theta) + s_3 \cos(\theta) \end{pmatrix} \\ &= \begin{pmatrix} \frac{s_1 \cos(\theta) - s_3}{\sin(\theta)} \\ s_1 \end{pmatrix}. \end{aligned} \tag{11.12}$$

However, when $\theta = 0$, the system only has a solution if $s_1 = s_3$, that is, y_1^+ and y_3^+ coincide. In this case, define $P_{13} = (0, s_1)$.

Analogous to the previous derivation, one can compute the remaining intersection points.

Intersecting y_1^- and y_2^- yields

$$P_{12} = \begin{pmatrix} \frac{-s_1 \cos(\phi) + s_2}{\sin(\phi)} \\ -s_1 \end{pmatrix} \tag{11.13}$$

assuming $\phi \neq 0$. However, for $\phi = 0$, $s_1 = s_2$ must hold in order for the system to have a solution and we define $P_{12} = (0, -s_1)$.

Lastly, we can intersect the channel walls y_3^- and y_2^+ to obtain the remaining point

$$P_{23} = \begin{pmatrix} \frac{s_3 \cos(\phi) + s_2 \cos(\theta)}{\sin(\theta - \phi)} \\ \frac{s_3 \sin(\phi) + s_2 \sin(\theta)}{\sin(\theta - \phi)} \end{pmatrix}. \tag{11.14}$$

This quantity is no longer well defined if $\sin(\theta - \phi) = 0$, that is $\phi = \theta = 0$ or $-\phi = \theta = \frac{\pi}{2}$. These special cases are considered later.

Using the intersection points P_{12}, P_{13} and P_{23}, we can compute the necessary quantities n_k and l_k, $k = 1, 2, 3$.

Recall that l_k is the length of e_k, i.e. the distance between its two endpoints. Therefore, we obtain

$$l_1 = ||P_{12} - P_{13}||, \quad l_2 = ||P_{12} - P_{23}||, \quad l_3 = ||P_{13} - P_{23}||. \tag{11.15}$$

Furthermore, the normals are given by

$$\mathbf{n}_1 = \frac{1}{l_1}\begin{pmatrix} -2s_1 \\ \frac{s_1\sin(\theta+\phi)-s_2\sin(\theta)-s_3\sin(\phi)}{\sin(\phi)\sin(\theta)} \end{pmatrix}, \tag{11.16a}$$

$$\mathbf{n}_2 = \frac{1}{l_2}\begin{pmatrix} s_1 + \frac{s_2\sin(\theta)+s_3\sin(\phi)}{\sin(\theta-\phi)} \\ -\frac{s_2\cos(\theta)+s_3\cos(\phi)}{\sin(\theta-\phi)} - \frac{s_1\cos(\phi)-s_2}{\sin(\phi)} \end{pmatrix}, \tag{11.16b}$$

$$\mathbf{n}_3 = \frac{1}{l_3}\begin{pmatrix} s_1 - \frac{s_2\sin(\theta)+s_3\sin(\phi)}{\sin(\theta-\phi)} \\ \frac{s_2\cos(\theta)+s_3\cos(\phi)}{\sin(\theta-\phi)} - \frac{s_1\cos(\theta)-s_3}{\sin(\theta)} \end{pmatrix}. \tag{11.16c}$$

Consequently, all necessary components in (11.7) are defined and the system can now be solved in order to derive the state variables U_k for each channel.
In the following, we examine the special cases we excluded earlier and determine the points P_{ij} in these cases. Afterwards, we are able to compute the U_k's regardless of the specific channel setting.

T-junction: First, we consider T-junctions, i.e. channels where $\theta = -\phi = \frac{\pi}{2}$. If $s_2 = s_3$, we can define the points P_{ij} as illustrated in Figure 11.3 (i):

$$P_{12} = \begin{pmatrix} -s_2 \\ -s_1 \end{pmatrix}, \quad P_{13} = \begin{pmatrix} -s_2 \\ s_1 \end{pmatrix}, \quad P_{23} = \begin{pmatrix} s_2 \\ 0 \end{pmatrix} \tag{11.17}$$

Hence, the equations in (11.7) can be written as

$$\begin{cases} -s_1h_1w_1 + s_2h_2w_2 + s_2h_3w_3 = 0 \\ -2\left(h_1w_1^2 + \frac{1}{2}gh_1^2\right) + \frac{1}{2}gh_2^2 + \frac{1}{2}gh_3^2 = 0 \\ -\left(h_2w_2^2 + \frac{1}{2}gh_2^2\right) + h_3w_3^2 + \frac{1}{2}gh_3^2 = 0. \end{cases} \tag{11.18}$$

Furthermore, if $s_2 \neq s_3$ the point P_{23} can instead be defined as

$$P_{23} = \begin{pmatrix} \min(s_2, s_3) \\ 0 \end{pmatrix} \tag{11.19}$$

(compare Figure 11.3 (ii)).

Straight channel: Next, we consider the straight channel, i.e. $\theta = \phi = 0$. The x-coordinate of the intersection of the channels in the one-dimensional skeleton and hence the center of the reference system is set to the end of channel 1. Naturally, we define the points

$$P_{12} = \begin{pmatrix} 0 \\ -s_1 \end{pmatrix}, \quad P_{13} = \begin{pmatrix} 0 \\ s_1 \end{pmatrix}. \tag{11.20}$$

For the point P_{23} it is sensible to set the y-coordinate to 0. Since there is no natural definition of the x-coordinate, we set it to s_1. Hence, we obtain

$$P_{23} = \begin{pmatrix} s_1 \\ 0 \end{pmatrix}. \tag{11.21}$$

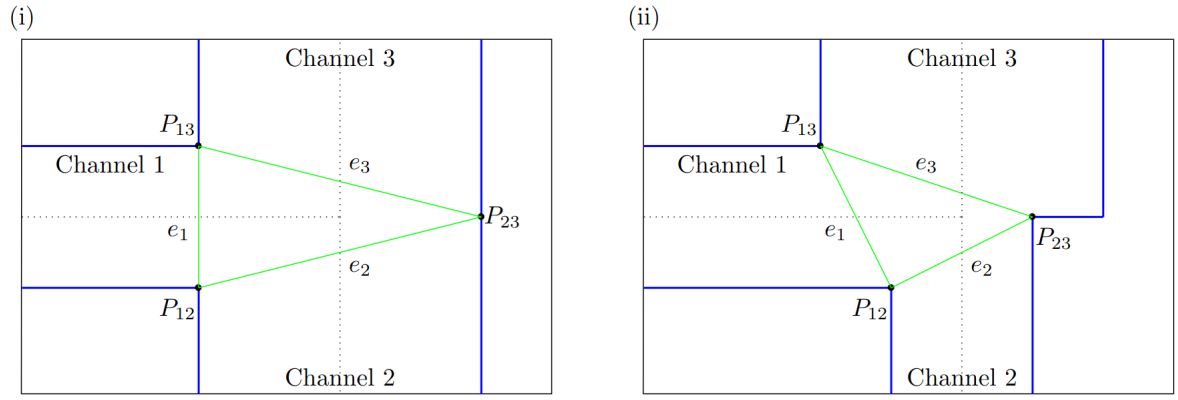


Figure 11.3: Two different versions of T-junctions: in both cases $\theta = -\phi = \frac{\pi}{2}$.
(i) $s_1 = 1, s_2 = s_3 = 2$. (ii) $s_1 = 1, s_2 = 1, s_3 = 2$.

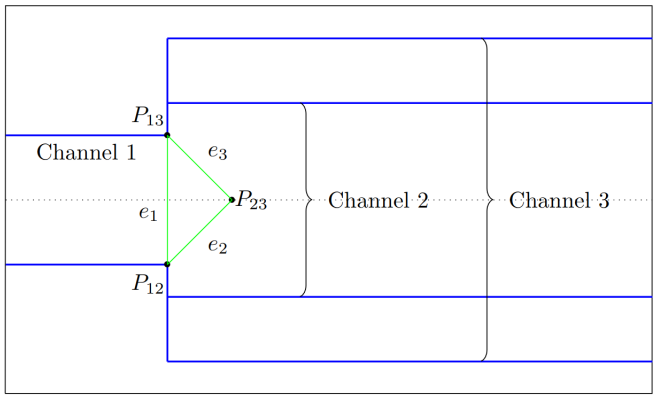


Figure 11.4: Straight channel with $s_1 = 1, s_2 = 1.5, s_3 = 2.5$.

The resulting triangle T is illustrated in Figure 11.4.
Hence, the equations (11.7) reduce to

$$\begin{cases} -2h_1w_1 + h_2w_2 + h_3w_3 = 0 \\ -2\left(h_1w_1^2 + \frac{1}{2}gh_1^2\right) + \left(h_2v_2^2 + \frac{1}{2}gh_2^2\right) + \left(h_3w_3^2 + \frac{1}{2}gh_3^2\right)\big) = 0 \\ -\frac{1}{2}gh_2^2 + \frac{1}{2}gh_3^2 = 0. \end{cases} \tag{11.22}$$

Finally, we considered all possible cases that can come up in our setting. To derive the six unknowns necessary for U_k, we solve the six equations in (11.2) and (11.7) as analyzed in this section.
In the next section, we investigate how the resulting states can be used to update the cell averages in the finite volume method, especially the ones adjacent to the junction. This enables us to compute a solution in networks without applying a two-dimensional solver.

11.3 Coupling of Junction Conditions and Finite Volume Scheme

Recall that we solve the system along each channel using a one-dimensional finite volume scheme. Therefore, we apply (4.8) in each channel, i.e. for $k = 1, 2, 3$

$$U_{j,k}^{n+1} = U_{j,k}^{n} - \frac{\Delta t}{\Delta x}\left[F_{j+1/2,k}^{n} - F_{j-1/2,k}^{n}\right]. \tag{11.23}$$

For simplicity, we assume all channels have length L. If we construct a uniform grid with $M+1$ grid cells $\mathcal{C}_0^k, \dots, \mathcal{C}_M^k$ in each channel k, they all have length $\Delta x = \frac{L}{M+1}$. Hence, the interfaces between the channels and the junction are given by $x_{M+1/2,1}$ and $x_{-1/2,k}$, $k = 2, 3$, where the first subindex refers to the interface value and the second to the respective channel.
However, at the junction, we update the states using the equations we derived above. The states U_k^*, i.e. the cell values in $\mathcal{C}_M^1$, the rightmost cell of Channel 1, and $\mathcal{C}_0^k$, $k = 2, 3$, the leftmost cell in Channel 2 and 3, are updated using the junction solver, i.e. the numerical flux at the interface facing the junction is computed by the coupling conditions as described in the following and the other ones by the one-dimensional solver in the channels.
Recall that combining (11.2) and (11.7) results in a nonlinear system of six equations. Given U_k^*, its solution yields the three states $U_k = (h_k, h_k w_k)$, $k = 1, 2, 3$. Hence, we can update the cell averages adjacent to the junction by defining $F_{M+1/2,1}^n = f(U_1)$, $F_{-1/2,2}^n = f(U_2)$ and $F_{-1/2,3}^n = f(U_3)$ where f is the original flux function given by the conservation law, i.e. (2.27b).

11.4 Pseudocode

The numerical program accompanying this work contains the file *network.py* which computes the solution of a finite volume method on a network. This section explains the two most important functions in pseudocode. Note that in our numerical program, a network can consist of several junctions connected by channels. However, it needs to have one in-going channel and each junction must be of the type described in Chapter 11.
Each run of the network solver requires a *network* that is initialized at the beginning using *Network()* in the file *network.py*. It fixes the following parameters:

1. the order that is used in the specific run
2. length of the network, i.e. number of channels
3. number of junctions (given by the number of channels, since each junction connects three channels)
4. since the network solver is constructed for the shallow water equations, the only valid choice in *EquationKind* is the shallow water system
5. number of grid cells for each channel
6. define each channel using *Channel()* in *channel.py* ▷ see below
7. define each junction using *Junction()* in *junction.py* ▷ see below
8. define numerical flux function ▷ e.g. TeCNO, Lax-Friedrichs or Godunov flux
9. if necessary: k and α_k in 7.2 ▷ e.g. for TeCNO or ELW scheme

Each channel has the fixed parameters:

1. start and end points, length of the channel
2. half width s
3. uniform grid ▷ see *grid.py*
4. number of grid cells

They are fixed for each run of the network solver by calling the function *Channel()*.
Furthermore, each junction consists of the following:

1. three channels ▷ Channel 1 aligned with x-axis
2. outgoing channel angles θ and ϕ
3. intersection points P_{12}, P_{13}, P_{23}
4. lengths of the 'junction triangle' walls l_i, $i = 1, 2, 3$
5. outward-pointing normals n_i, $i = 1, 2, 3$

They are fixed for each run of the network solver by calling the function *Junction()*.
Furthermore, the file *junction.py* contains all functions necessary to compute the values above, e.g. *intersection_points()* or *normals()*.
Now, we introduce the general method that computes solutions across networks.

Algorithm 6 *solve_network()*

Computes solution on a network using the finite volume method in the channels and the junction solver to update states across the junction.

net = Network(): ▷ initializes a new network consisting of channels and junctions
cell_averages: compute the initial cell averages depending on the initial function in each channel
fluxes: initialize fluxes with zero-arrays for each channel
t = *start_time*
while t < end_time **do**:
    compute Δt using *compute.cfl_condition()* ▷ choose minimum over all channels
    update *cell_averages* with respect to the required order ▷ see *solve_network_timestep()*
    $t = t + \Delta t$
end while
cut off ghost cells
return *cell_averages*

Now, we consider the function that computes one time step of the finite volume method.

Algorithm 7 *solve_network_timestep()*

Computes one time step in the finite volume method on networks.

1: **for** each channel **do**
2: compute fluxes according to the respective scheme to update *cell_averages* in each channel
3: **end for**
4: **for** each junction **do**
5: solve junction solver to obtain fluxes at the junction interfaces
6: **end for**

Note that updating the cell averages can be done by employing either the TeCNO scheme, the Lax-Friedrichs scheme or the Godunov scheme that are implemented in the files *solver.tecno.py*, *solver.lax-friedrichs.py* and *solver.godunov.py*.

11.5 Solutions to the Junction Solver

In this section, we want to discuss solutions to the nonlinear system at the junction. However, since existence and uniqueness results for solutions to general nonlinear systems are difficult to prove, we can only present partial solutions here.
As before, let h_k^*, w_k^* denote the values obtained by the one-dimensional solver in Channel k, $k = 1, 2, 3$ adjacent to the junction and h_k, w_k the values facing the junction. Furthermore, we define

$$X = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}, \quad X^* = \begin{pmatrix} h_1^* \\ h_2^* \\ h_3^* \\ w_1^* \\ w_2^* \\ w_3^* \end{pmatrix} = \begin{pmatrix} h_{M,1} \\ h_{0,2} \\ h_{0,3} \\ w_{M,1} \\ w_{0,2} \\ w_{0,3} \end{pmatrix} \tag{11.24}$$

where, again, the first index refers to the cell and the second to the channel. Since we considered a fluvial regime, the open set of admissible states is defined by

$$\Omega = \left\{ h_k \in \mathbb{R}^+, w_k \in \mathbb{R} : |w_k| < \sqrt{g h_k},\ k = 1, 2, 3 \right\}. \tag{11.25}$$

Solving the equations (11.2) and (11.7) can now be interpreted as solving the system

$$\Psi(X; X^*) = 0 \tag{11.26}$$

with $\Psi : \Omega \times \Omega \to \mathbb{R}^6$.
Assume the system has been solved up to $t = t_n$, i.e. we have a solution $X^{n,*}$ adjacent to the junction. Furthermore, assume we have found a solution X^n such that $\Psi(X^n; X^{n,*}) = 0$. Let $D\Psi$ denote the Jacobian with respect to the first argument. If $\mathrm{Det}(D\Psi(X^n; X^{n,*})) \neq 0$, i.e. $D\Psi(X^n; X^{n,*})$ is invertible, we can apply the implicit function theorem. It states that there exists an open set $\Theta(X^{n,*})$ containing $X^{n,*}$ such that there exists a unique, continuously differentiable function $X : \Theta \to \mathbb{R}^3$, $X^* \mapsto X(X^*)$ such that $\Psi(X; X^*) = 0$. Hence, for a smooth flow, we can find Δt small enough such that $X^{n+1,*} \in \Theta(X^{n,*})$ and therefore there exists a unique X^{n+1} such that $\Psi(X^{n+1}; X^{n+1,*}) = 0$.

In summary, we can apply this procedure in each iteration n if we can prove $D\Psi(X^n; X^{n,*}) \neq 0$. This guarantees the existence of a solution X^{n+1} to the system $\Psi(X^{n+1}; X^{n+1,*}) = 0$.
In particular, if all waves are rarefaction waves, the equations in (11.2) yield

$$\begin{cases} w_1 + 2\sqrt{gh_1} = w_1^* + 2\sqrt{gh_1^*} \\ w_2 - 2\sqrt{gh_2} = w_2^* - 2\sqrt{gh_2^*} \\ w_3 - 2\sqrt{gh_3} = w_3^* - 2\sqrt{gh_3^*} \end{cases} . \tag{11.27}$$

Since the equations (11.7) do not depend on X^*, the Jacobian $D\Psi$ does not either. Hence, in this case, it can be proven that a solution exists at each time step if we start with data X^* and a solution X such that $\Psi(X; X^*) = 0$ and can show that $\mathrm{Det}(D\Psi(X; X^*)) \neq 0,\ \forall X \in \Omega$.

11.6 Reference Solutions on Networks

To evaluate the numerical solution obtained by coupling the junction solver and the one-dimensional scheme, we additionally require a solution on the network that can serve as a reference solution. In general, we can use another numerical solver, e.g. the Lax-Friedrichs method (10.1), in the channel and compare the resulting solution to the one obtained by applying the TeCNO scheme. However, for specific initial data, we can exploit the junction solver in Section 11.2 to derive a reference solution without applying a numerical solver in the channels. This works for example for a dam-break Riemann problem where the jump is located at the junction. Since the flow in each channel in the network can be modeled as a one-dimensional problem, we use the formulas for rarefaction and shock waves derived in Chapter 3. However, in order to apply these formulas, we require knowledge about the specific type of wave in each channel. As explained in Section 3.1.1, the solution to the dam-break problem (3.2) always consists of a rarefaction wave going in the direction of the higher initial water height and a shock wave leading in the opposite direction.
Using this fact enables us to derive solutions to certain Riemann problems on networks. First, we assume a constant water height in each channel and a global velocity $w^* \equiv 0$. Using the notation from Chapter 11, we denote the initial water height in channel k by h_k^*, i.e. the initial data is given by

$$h(x,0) = \begin{cases} h_1^*, & x \text{ in Channel 1} \\ h_2^*, & x \text{ in Channel 2} \\ h_3^*, & x \text{ in Channel 3} \end{cases}, \quad w^*(x,0) = 0. \tag{11.28}$$

If $h_2^* = h_3^* > h_1^*$, we know that the solution consists of a left-going shock wave in Channel 1 and right-going rarefaction waves in Channel 2 and 3. Analogously, for $h_2^* = h_3^* < h_1^*$, we have a left-going rarefaction and two right-going shock waves.
However, the formulas for computing rarefaction and shock waves require knowledge about the middle state. Instead of applying a root finder to the function $\phi(h)$ defined in Section 3.5, we exploit the junction solver from Section 11.2.
Hence, to determine the middle state across all channels, we apply the junction solver to the states $U_k^* = (h_k^*, h_k^* w_k^*) = (h_k^*, 0)$ given by the initial data. The resulting states U_k then yield the respective middle state for each channel. Using this approach, we can compute the evolution of the waves in the network and obtain a reference solution.
The corresponding implementation can be found in *solver.exact_solution.py*. This method uses the

functions *rarefaction_wave_1()*, *rarefaction_wave_2()*, *shock_1()* and *shock_2()* that compute the waves based on Chapter 3.

Algorithm 8 *riemann_solution_on_network()*

Computes the Riemann solution on a network as explained in this section.

junction_states = network.solve_nonlinear_system: ▷ returns cell states across junction

In Channel 1:

if $h_m < h_l$ **then**: ▷ i.e. rarefaction wave in Channel 1

solution_channel_1 = rarefaction_wave_1()

else ▷ i.e. shock in Channel 1

solution_channel_1 = shock_1()

end if

In Channel i=2, 3:

if $h_m < h_l$ **then**: ▷ i.e. rarefaction wave in Channel 2/3

solution_channel_i = rarefaction_wave_2()

else ▷ i.e. shock in Channel 2/3

solution_channel_i = shock_2()

end if

12 Numerical Results – Network

This Chapter presents numerical solutions on networks. To evaluate their quality, we compare them to a reference solution. For initial data of the form (11.28), we can apply the procedure explained in Section 11.6. However, for general initial data, we need to compute a numerical reference solution. Similar to Chapter 10, we use the local Lax-Friedrichs method (10.1) with minmod reconstruction (10.2) as the one-dimensional solver in each channel. Furthermore, we again use the junction solver to determine the fluxes across the junction.
In this chapter, the number of grid cells n for a numerical solution always denotes the number of cells in each channel, i.e. if e.g. $n = 64$, we have a total of $3 \cdot 64 = 192$ cells. In this section, we always use $k = p$ in (7.2).

12.1 Dam-Break Riemann Problem

In this section, we compare the exact solution to the Riemann problem on networks as explained in Section 11.6 to the numerical solution obtained by the junction solver derived in Section 11.2.
We consider a network consisting of three channels with $\theta = -\phi = \frac{\pi}{4}$, i.e. the solutions in Channel 2 and 3 coincide. Furthermore, we choose the channel width $2s_k = 2$, $k = 1, 2, 3$.
The initial data is given in (11.28) with $h_1^* = 2$, $h_2^* = h_3^* = 1.5$. Hence, we expect a left-going rarefaction wave in Channel 1 and a right-going shock in Channel 2 and 3.
The numerical solutions on different numbers of grid cells for fixed orders as well as the reference solution derived as explained in Section 11.6 are depicted in Figure 12.1. Clearly, the numerical solutions approach the reference solution as the number of grid cells and the order increases.

12.2 Dam-Break Riemann Problem for Different Angles

Now, we study the influence of the angle on the solution to the dam-break Riemann problem on networks. Furthermore, we show that the solution in a network on a straight channel (see Section 11.2) is consistent with the solution on an interval. We repeat the same numerical examples we considered in the previous section. However, now we vary the angle $\theta = -\phi \in \{0, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}\}$ with $s_1 = 1$, $s_2 = s_3 = 0.5$. We use the TeCNO_2 scheme in the channels with 256 grid cells. The resulting solutions for the water height h can be found in Figure 12.2. Note that the solutions on Channel 2 and Channel 3 coincide, so we derive the displayed solution by connecting the solutions to Channel 1 and Channel 2 at the junction, which is here represented by a dashed vertical line. Note that the blue line in the figure represents the numerical network solution on a straight channel. As expected, it is very close to the exact Riemann solution on an interval (pink line), so these two variants are consistent.
We can see that the angle of the outgoing channels has an effect on the resulting numerical network solutions. For a straight channel, there is no jump in water height at the junction. However, the jump expands as the angle increases. Furthermore, we can see the angle has apparently no impact on the location of the shock and rarefaction wave.

12.3 Smooth Initial Data

In the following, we investigate numerical solutions to smooth initial data. Since we are not able to derive exact solutions for our problems, we use reference solutions on fine grids computed with the local Lax-Friedrichs scheme (10.1) with minmod reconstruction (10.2). The channel widths are given

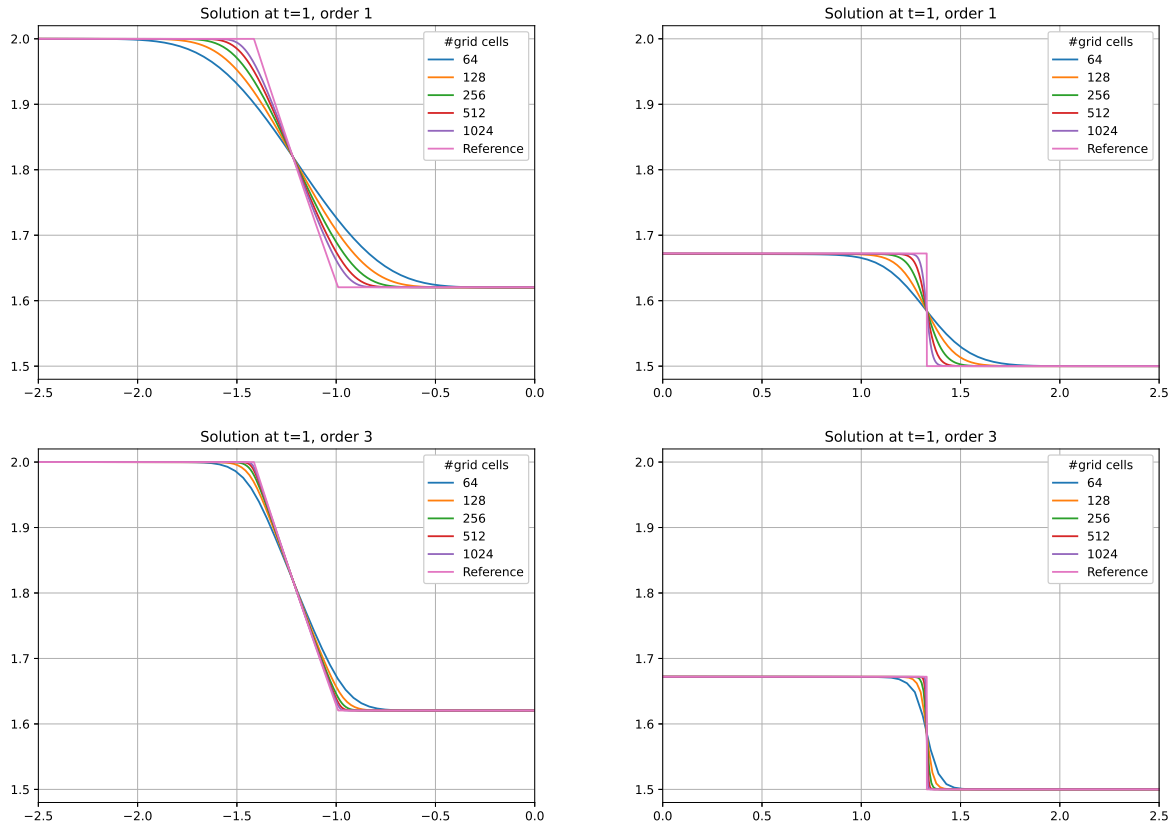


Figure 12.1: Solutions to the dam-break Riemann problem on varying numbers of grid cells for order one and three on a network with $\theta = -\phi = \frac{\pi}{4}$ resulting from combining the one-dimensional TeCNO$_p$ scheme and the junction solver. The left column shows the solutions in Channel 1 and the right column the solutions in Channel 2 and 3.

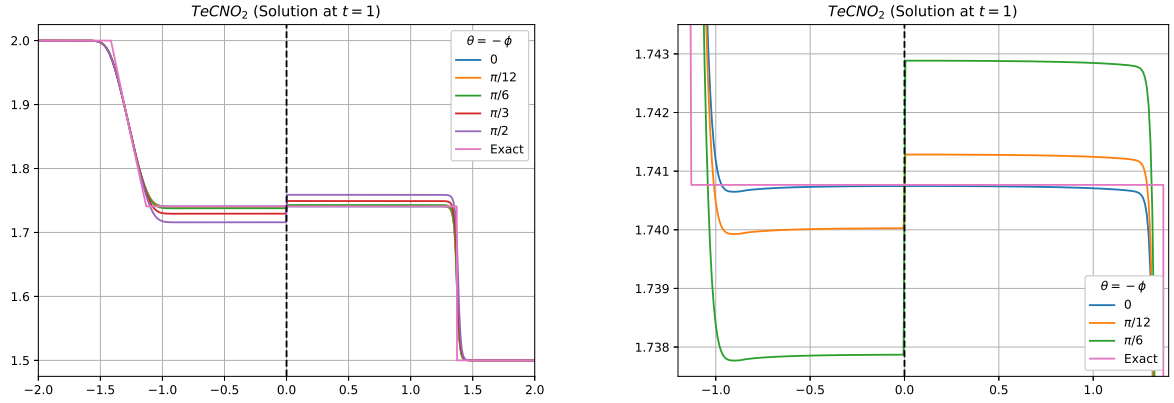


Figure 12.2: Comparison of solutions to different network geometries for the same initial problem. The pink line shows the exact solution on an interval. The others are composed of the numerical network solutions of Channel 1 and Channel 2 where the junction is represented by the vertical dashed line. The left image shows the resulting water height and the right image a close-up for some angles.

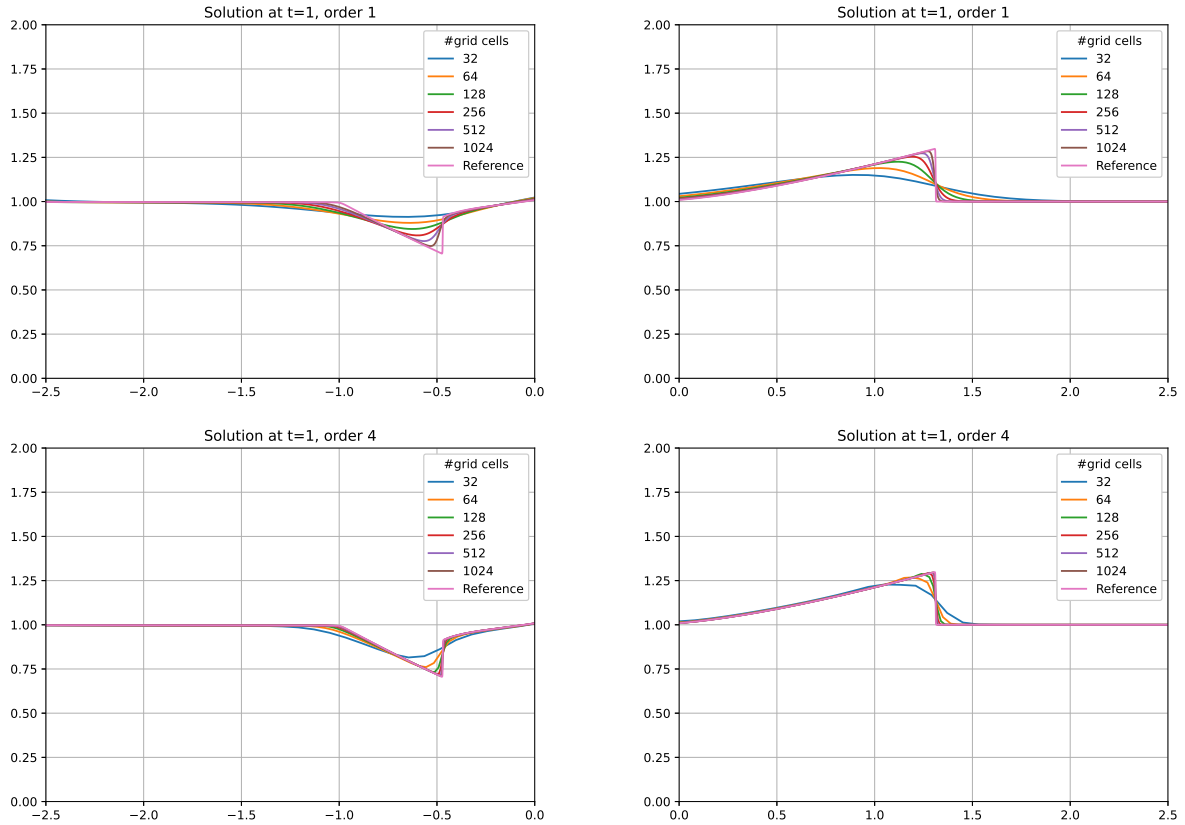


Figure 12.3: Solutions to an initial Gaussian wave on varying numbers of grid cells for order one and three on a network with $\theta = -\phi = \frac{\pi}{4}$ resulting from combining the one-dimensional TeCNO$_p$ scheme and the junction solver. The left column shows the solutions in Channel 1 and the right column the solutions in Channel 2 and 3.

by $s_k = 1$, $k = 1, 2, 3$.

The initial data is determined by a Gaussian function (10.14) with $a = 1$, $x_0 = -1.5$, $s = 0.2$ and $h_0 = 1$. We have seen in Section 10.6 that an initial Gaussian hump in a one-dimensional channel develops two waves propagating in different directions. However, as soon as the right-going wave reaches the junction, it splits again into three waves, two continue to propagate into Channel 2 and Channel 3 and the other one ricochets back into Channel 1.

The solution at time $t = 2$ is depicted in Figure 12.3. The reference solution is computed on 4096 grid cells in each channel. We can clearly see that the TeCNO solutions approach the reference solution as the grid is refined. Furthermore, we see that TeCNO$_4$ solutions capture the shocks more accurately than the TeCNO$_1$ solutions using the same number of grid cells.

12.4 Smooth Initial Problem for Different Angles

In this section, we consider smooth initial data on different networks to determine the influence of the angle on the water height as the wave propagates through the junction. As initial data, we use the Gaussian wave defined in the previous section and compute solutions up to $t = 2$. We consider the TeCNO$_2$ scheme on 256 grid cells in each channel and assume $\theta = -\phi$. Hence, the solutions in Channel 2 and Channel 3 coincide. We choose $s_1 = 1$, $s_2 = s_3 = 0.5$. The results can be found in Figure 12.4. As

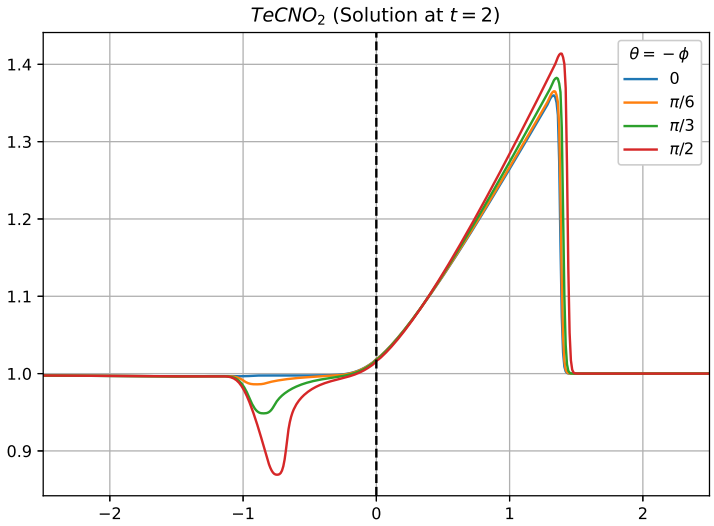


Figure 12.4: Comparison of solutions to different networks for the same initial problem. The solutions are composed of the numerical network solutions of Channel 1 and Channel 2 (which coincides with the solution on Channel 3) where the junction is represented by the vertical dashed line.

in Section 12.2, the image shows the solutions in Channel 1 and Channel 2, composed at the junction which is represented by a vertical dashed line. We can see, that the angle indeed has an impact on the resulting wave. The wave height in Channel 2 rises as the angle increases, but only slightly. The impact on the reflecting wave however is much greater, while there is essentially no reflecting wave for a straight channel or a small angle, it develops a downward deflection as the angle increases. This fits our intuitive expectations since the incoming wave hits the channel walls differently for larger angles. These findings are important for real-world applications. For example in the construction of harbours, it is important to take into account how the angle between channels influences the water and can lead to large perturbations. To ensure that the water remains as smooth as possible – as it is desired to prevent inundations or large waves impacting boats – the angle should be chosen small.

13 Stochastic Collocation Method

The model described previously can be used to model water in a collection of channels separated by junctions. So far, we required specifically given initial data in order to compute a solution. In practical applications, however, this data is not always available and one has to account for a certain level of inaccuracy that could, for instance, be caused by measurement errors. Hence, the initial variables can vary in a small interval without the possibility of determining them exactly. Recall Example 3.1.1 where the initial water heights are piecewise constant and the velocity is zero. In practice, we might obtain something of the form

$$u(x,t) = \begin{cases} u_l + \nu_1 \\ u_r + \nu_2 \end{cases} \tag{13.1}$$

where ν_1, ν_2 are small, unknown deviations from the base states u_l, u_r.
Hence, in this chapter, we investigate random differential equations and apply the stochastic collocation method to solve them based on [10, 11] to derive their sample mean and standard deviation.

13.1 Random Differential Equations

First, let $(\Omega, \mathcal{A}, \mathcal{P})$ be a complete probability space. Ω denotes the event space, $\mathcal{A} \subset 2^\Omega$ the σ-algebra and $\mathcal{P}$ the probability measure. In this section, we introduce differential equations that contain a random component, so-called *random differential equations*. In their general form, they are given by

$$\mathcal{L}(\tilde{\omega}, x; u) = g(\tilde{\omega}, x), \quad x \in D \tag{13.2a}$$

with boundary conditions

$$\mathcal{B}(\tilde{\omega}, x; u) = f(\tilde{\omega}, x), \quad x \in \partial D \tag{13.2b}$$

where $D \subset \mathbb{R}^d$ is a d-dimensional bounded domain with boundary ∂D, $\mathcal{L}$ a differential operator, $\mathcal{B}$ a boundary operator and u the unknown function we seek to determine. We assume that this governing equation is well-posed for $\mathcal{P}$-a.e. $\tilde{\omega} \in \Omega$.
To solve the differential equation numerically, the infinite-dimensional probability space needs to be reduced to a finite-dimensional one, see [10] for more details on that. From now on, we assume the random inputs can be represented by r real variables $\{\omega^i\}_{i=1}^r$. Then, following the Doob-Dynkin lemma [29], the problem (13.2a) can be written as

$$\mathcal{L}\left(\omega^1(\tilde{\omega}), \ldots, \omega^r(\tilde{\omega}), x; u\right) = g\left(\omega^1(\tilde{\omega}), \ldots, \omega^r(\tilde{\omega}), x\right), \quad x \in D. \tag{13.3}$$

In the following, we often just write

$$\mathcal{L}(\omega(\tilde{\omega}), x; u) = \mathcal{L}(\omega, x; u) = g(\omega, x), \quad x \in D \tag{13.4}$$

instead of using the expression in (13.3).
Now, we assume the random variables $\{\omega^i\}_{i=1}^r$ are independent with probability density functions

$$\rho_i : \Gamma^i \to \mathbb{R}^+ \tag{13.5}$$

and images $\Gamma^i \equiv \omega^i(\Omega)$ that are bounded intervals in $\mathbb{R}$. Then, we can write the joint probability density function of $\omega = (\omega^1, \dots, \omega^r)$ as

$$\rho(\omega) = \prod_{i=1}^{r} \rho_i(\omega^i), \quad \omega \in \Gamma \tag{13.6}$$

with support

$$\Gamma \equiv \bigtimes_{i=1}^{r} \Gamma^i \subset \mathbb{R}^r. \tag{13.7}$$

Note that we can extend (13.3) to time-dependent problems by defining D as a $d+1$ dimensional space where the spacial variable $x \in \mathbb{R}^d$ and the temporal variable $t \in \mathbb{R}^+$. Then, for the one-dimensional shallow water equations (2.27), we have $\mathcal{L}(\omega, (x,t); u) = u(\omega, x, t)_t + f(u(\omega, x, t))_x$ and $g(\omega, (x,t)) = 0$. Hence, (13.3) takes the form

$$u(\omega, x, t)_t + f(u(\omega, x, t))_x = 0 \tag{13.8a}$$

with

$$u(\omega, x, t) = \begin{bmatrix} h \\ hw \end{bmatrix}, \quad f(u(\omega, x, t)) = \begin{bmatrix} hw \\ hw^2 + \frac{1}{2}gh^2 \end{bmatrix} \tag{13.8b}$$

where $h = h(\omega, x, t)$ and $w(\omega, x, t)$ – now depending on the random variable ω – denote the water height and the velocity, respectively.
There are several numerical methods available to solve this type of equation. One of the most intuitive ones is the *Monte Carlo method* which we briefly introduce in the following. Note that for each fixed $\omega \in \Gamma$, (13.3) is a deterministic problem and can be solved with a standard deterministic solver such as the finite volume method with the TeCNO scheme. However, it has the drawback that for changing ω we always have to recompute the solution which highly increases the computational complexity. The Monte Carlo method relies on the law of large numbers, i.e. that the average of the results approaches the expected value with an increasing number of samples. While the stochastic collocation method is, in principle, similar to the Monte Carlo method, the random variables are fixed at certain points and not randomly chosen. Here, the choice of *collocation points*, i.e. the points in the random variable space that are fixed in order to solve the deterministic problem is crucial. We discuss their importance and different choices in Section 13.3.2.

13.2 Monte Carlo Method

The Monte Carlo method is a standard method for computing statistics from a number of random samples. Let $\omega_1, \dots \omega_n$ denote n independent and identically distributed points in the random variable space. Then, the problem (13.3) can be solved for $u(\omega_i, x)$ for each ω_i, $i = 1, \dots n$ with a deterministic

solver. The resulting solutions $u_i(\omega_i, x)$, $i = 1, \dots, n$ can now be used to approximate the mean by the *sample mean*:

$$\mathbb{E}[u(x)] = \int_\Gamma \rho(\omega) u(\omega, x) d\omega \approx \mathbb{E}_n[u(x)] = \sum_{i=1}^{n} u(z_i, x)\rho(z_i). \tag{13.9}$$

By the law of large numbers, we expect the sample mean to be close to the actual mean if n is big enough.
This approach has the advantage of being quite intuitive and easy to implement. However, a typical Monte Carlo simulation consisting of n realizations has a very slow convergence rate of $\frac{1}{\sqrt{n}}$, i.e. in order to halve the error, one has to quadruple the number of samples. Hence, the procedure requires many sample points, i.e. many applications of the deterministic solver in order to be accurate. This is computationally very expensive and, therefore, less applicable in practice.

13.3 Stochastic Collocation Method

Now, we introduce the stochastic collocation method. Similar to the Monte Carlo method, it is intuitive and easy to implement once the deterministic solver is available, while it has a higher resolution resulting from polynomial approximations in random spaces.
The construction of high-order stochastic collocation methods is based on polynomial interpolations in the random space. In this section, we first introduce the *Lagrange interpolation problem* which is crucial for the stochastic collocation method. Then, we discuss suitable choices of collocation points such that the resulting function is as accurate as possible with the least number of points.

13.3.1 Lagrange Interpolation

To introduce Lagrange interpolation, first denote by Π_r the space of all r-variate polynomials with real coefficients and by Π_r^q the subspace of polynomials with a degree of at most q.

Definition 13.1. Let x_i, $i = 1, \dots, n$ be a given set of n nodes. Then, the n *Lagrange basis polynomials* are given by

$$L_i(x) := \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{13.10}$$

Note that they satisfy $\deg(L_i) = n - 1$ and $L_i(x_j) = \delta_{ij}$, $i, j = 1, \dots, n$.

Definition 13.2. Let $\omega_1, \dots, \omega_M \in \Gamma$ denote points in the random space and $b_1 \dots, b_M \in \mathbb{R}$ real constants. Given a subspace $V \subset \Pi_r$, find a polynomial $p \in V$ such that

$$p(\omega_i) = b_i, \quad i = 1, \dots, M. \tag{13.11}$$

This is called *Lagrange interpolation*. The points ω_i, $i = 1, \dots, M$ are called *interpolation nodes* and the subspace V *interpolation space*.

Definition 13.3. Let $\omega_1, \dots, \omega_M \in \mathbb{R}^r$ be given interpolation points. The Lagrange interpolation problem in Definition (13.2) is called *poised in* V if for all $b_1, \dots, b_M \in \mathbb{R}$ there exists a function $g \in V$ such that $g(\omega_i) = b_i$, $i = 1, \dots, M$. When the Lagrange interpolation problem is poised in V for all M distinct points in $\mathbb{R}^r$, then V is called a *Haar space of order M*.

However, even though there are several available Haar spaces for $r = 1$, there are none for $r > 1$, this is the so-called *loss-of-Haar* (see [30, 31] for more details on that). Hence, especially in higher dimensional spaces it is important to select suitable points ω_i such that the approximation by a Lagrange interpolation is still good.
So, for now, assume we are given M distinct points $\{\omega_1, \dots, \omega_M\} = \Theta_r$ and an interpolation space $V \in \Pi_r$ such that the interpolation problem is poised in V. If we want to interpolate a smooth function $f : \mathbb{R}^r \to \mathbb{R}$ using Lagrange interpolation, equation (13.11) takes the form

$$\mathcal{I}(f)(\omega_i) = f(\omega_i), \quad i = 1, \dots, M \tag{13.12}$$

where $\mathcal{I}(f) \in V$ is the polynomial function we want to find. This polynomial can be written as

$$\mathcal{I}(f)(\omega) = \sum_{i=1}^{M} f(\omega_i) L_i(\omega) \tag{13.13}$$

where $L_i(\omega) \in V$ are the Lagrange polynomials. Since $L_i(x_j) = \delta_{ij}$ for $1 \le i, j \le M$, $\mathcal{I}(f)$ satisfies the interpolation property (13.12).
If we equip the interpolation space V with the supremum-norm $\|\cdot\|_\infty$ and define

$$\|\mathcal{I}\| = \sup_{f \neq 0} \frac{\|\mathcal{I}(f)\|_\infty}{\|f\|_\infty}. \tag{13.14}$$

Then, Lebesgue's theorem guarantees that the interpolation error is uniformly bounded as

$$\|f(\omega) - f^*(\omega)\|_\infty \le \|f(\omega) - \mathcal{I}(f)(\omega)\|_\infty \le (1 + \Lambda)\|f(\omega) - f^*(\omega)\|_\infty \tag{13.15}$$

where $f^*(\omega)$ is the best approximating polynomial for f and $\Lambda = \|\mathcal{I}\|$ is the Lebesgue constant. However, it is important to note that finding $f^*(\omega)$ is in general an unsolved problem for r-dimensional spaces with $r > 1$. Furthermore, even for $r = 1$, estimating the Lebesgue constant is a non-trivial task.

13.3.2 Choices of Collocation Points

As mentioned earlier, the stochastic collocation method solves the deterministic system that arises for a fixed random variable at predetermined points. Hence, it works similarly to the Monte Carlo method where the points were chosen at random. The crucial part in the application of the stochastic collocation method is the choice of the collocation points. Here, two conflicting objectives have to be balanced:
Firstly, the computational complexity highly depends on the number of points – for each point we have to solve a deterministic problem, i.e. for n points the computational effort is n times that of the underlying deterministic solver. Therefore, in order to decrease the computational costs, the amount of collocation points should remain small. Secondly, however, too few points lead to an inaccurate interpolation – we need sufficiently many points to be able to accurately reconstruct the original function f. Hence, we need to choose a nodal set Θ_n with the fewest possible number of collocation points from the random variable space under a prescribed accuracy requirement.
Without loss of generality, we restrict the bounded support of the random space to the r-dimensional hypercube $[-1, 1]^r$. In the following, we investigate two different choices of points. We mainly focus on multi-dimensional random parameter spaces, i.e. $\Gamma \subset \mathbb{R}^r$, $r > 1$.

For $r = 1$ and smooth functions $f : [-1, 1] \to \mathbb{R}$, the interpolating function is of the form

$$\mathcal{U}(f) = \sum_{k=1}^{m} f(\omega_k) \cdot a_k \tag{13.16}$$

where $\Theta_r = \{\omega_1, \ldots, \omega_m\} \subset [-1, 1]$ and $a_k \equiv a_k(\omega_k) \in C([-1, 1], \mathbb{R})$. This problem is well studied and there is a variety of collocation points available to obtain a good approximation, e.g. Gaussian quadrature points or extrema of Chebyshev polynomials. However, for higher-dimensional random variables, the construction of suitable collocation points is more difficult and we investigate that problem in the remainder of this section.

Tensor Product Since there are several good interpolation formulas available for one-dimensional random spaces, it is a natural choice to use a tensor product of one-dimensional nodes in more dimensions. For every component $i = 1, \ldots, r$, we use a nodal set

$$\Theta_r^i = \{\omega_1^i, \ldots, \omega_{m_i}^i\} \tag{13.17}$$

with $\omega_j^i \in [-1, 1]$. In this case, (13.16) takes the form

$$\mathcal{U}^i(f) = \sum_{k=1}^{m_i} f(\omega_k^i) \cdot a_k^i \tag{13.18}$$

with $a_k^i \equiv a_k(\omega^i) \in C([-1, 1], \mathbb{R})$. Combining them results in the following tensor product formula:

$$\mathcal{I}(f) \equiv (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_r})(f) = \sum_{k_1=1}^{m_{i_1}} \cdots \sum_{k_r=1}^{m_{i_r}} f(\omega_{k_1}^{i_1}, \ldots, \omega_{k_r}^{i_r}) \cdot (a_{k_1}^{i_1} \otimes \cdots \otimes a_{k_r}^{i_r}) \tag{13.19}$$

Hence, the tensor product of these sets consists of $M = m_1 \cdot \ldots \cdot m_r$ nodes. If we use the same number of nodes for each dimension, i.e. $m_1 = \ldots = m_r = m$, we obtain a total of $M = m^r$ points. Unfortunately, in high-dimensional spaces, this number grows quickly. Even for $m = 2$, i.e. only two collocation points in each component – which would still lead to a very poor approximation – we obtain a set with total size $M = 2^r \gg 1$ for $r \gg 1$. This *curse of dimensionality* is the reason that this approach is not applicable in practice whenever the underlying random space is high-dimensional. Hence, in this case, we can instead use the sparse grid approach considered in the next section.

Sparse Grid In order to prevent these quickly growing nodal sets, we solve the problem on a *sparse grid* constructed by the *Smolyak algorithm*. Here, we have a linear combination of product formulas chosen such that an interpolation property for $r = 1$ is preserved for $r > 1$.
Define $\mathcal{U}^i(f)$ by (13.18). The Smolyak algorithm is given by

$$\mathcal{I}(f) \equiv A(q, r) = \sum_{q-r+1 \le |\mathbf{i}| \le q} (-1)^{q-|\mathbf{i}|} \cdot \begin{pmatrix} r-1 \\ q-|\mathbf{i}| \end{pmatrix} \cdot (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_r}) \tag{13.20}$$

where $\mathbf{i} = (i_1, \ldots, i_r) \in \mathbb{N}^r$ and $|\mathbf{i}| = i_1 + \cdots + i_r$. Hence, we only need to evaluate f on the *sparse grid*

$$\Theta_r \equiv \mathcal{H}(q, r) = \bigcup_{q-r+1 \le |i| \le q} (\Theta_1^{i_1} \times \cdots \times \Theta_1^{i_r}). \tag{13.21}$$

In this work, we use Smolyaks formulas based on collocation points that are based on the extrema of Chebyshev polynomials:

$$\omega_i^j = -\cos\left(\frac{\pi \cdot (j-1)}{m_i - 1}\right), \quad j = 1, \ldots, m_i \tag{13.22}$$

for $m_i > 1$ and $\omega_i^1 = 0$ if $m_i = 1$.
Furthermore, choose $m_1 = 1$ and $m_i = 2^{i-1} + 1$ for $i > 1$. Then, the one-dimensional sets Θ_1^i are nested, i.e. $\Theta_1^i \subset \Theta_1^{i+1}$, and therefore $\mathcal{H}(q, r) \subset \mathcal{H}(q+1, r)$. This makes it easier to extend the collocation set if required without the necessity to compute a new solution at every point.
If we set $q = r + k$, then $A(r+k, r)$ is exact for all polynomials in Π_r^k [32]. We call k the *level of the Smolyak construction*. Furthermore, the total number of nodes for $r \gg 1$ is given by [33]

$$M = \dim(A(r+k, r)) \sim \frac{2^k}{k!} r^k, \quad k \text{ fixed}, \; r \gg 1. \tag{13.23}$$

Figure 13.1 emphasises the importance of using the sparse grid approach instead of the tensor product in terms of computational complexity. The two-dimensional interpolation nodes are given by the sparse grid $\mathcal{H}(r+k, r)$ with $r = 2$, $k = 5$ sum up to 145 points. On the other side, the tensor product of the same one-dimensional nodes results in 1089 points, i.e. a seven-fold increase.
The numerical solutions obtained by applying the deterministic solver to the equation (13.3) with fixed $\omega \in \Theta_r$ can now be used to approximate the mean by the sample mean

$$\mathbb{E}(\hat{u})(x) = \sum_{k=1}^{n} u(\omega_k, x) \int_\Gamma L_k(\omega)\rho(\omega)d\omega. \tag{13.24}$$

Since the integral requires explicit knowledge about the polynomials L_k, this expression is not straightforward to evaluate, especially in the multivariate case.
However, it can be simplified by using cubature rules to approximate the integral. Here, choosing M cubature points yields

$$\int_\Gamma f(\omega)d\omega \simeq \sum_{i=1}^{M} f(\omega_i) w_i \tag{13.25}$$

where $\{w_i\}_{i=1}^M$ are the corresponding weights. If the cubature point set is chosen as the collocation point set Θ_r, the computation of the mean can be performed by

$$\mathbb{E}(\hat{u}) = \sum_{k=1}^{n} u(\omega_k, x)\rho(\omega_k) w_k \tag{13.26}$$

since $L_i(\omega_j) = \delta_{ij}$.

13.4 Pseudocode

The code for the stochastic collocation method can be found in *stochastics.stochastic_collocation_method.py*.

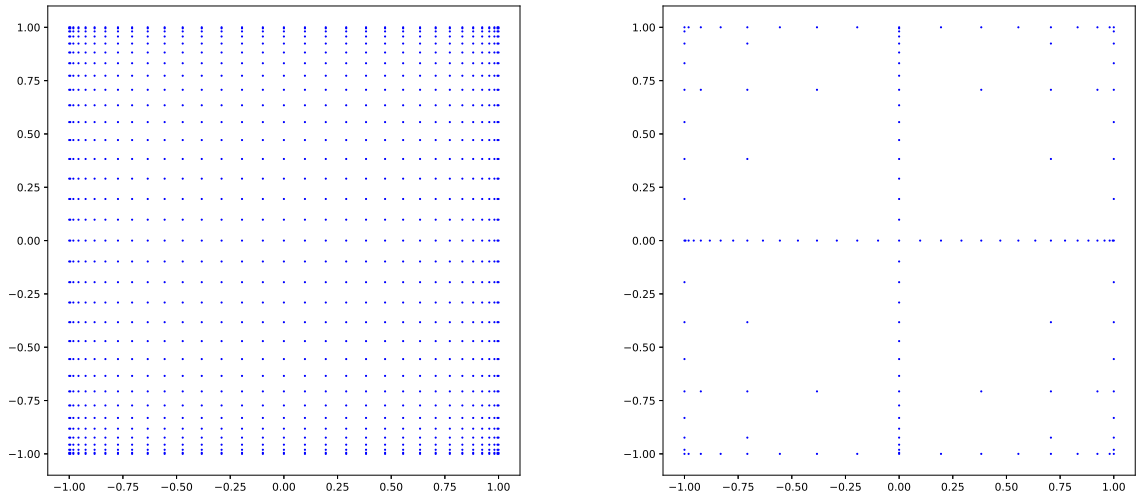


Figure 13.1: Two-dimensional interpolation nodes based on the extrema of Chebyshev polynomials (13.22). Left: tensor product with 1089 nodes. Right: sparse grid $\mathcal{H}(r+k, r)$ from Smolyak algorithm with $k = 5$: 145 nodes.

Algorithm 9 *stochastic_collocation_method()*

Solves the random differential equation using the stochastic collocation method for $r = 2$ for a fixed number of grid cells g and order *order*.

```
define probability density function
compute collocation points Θ either on a sparse grid or tensor product   ▷ with potential scaling
for ω ∈ Θ do
    define initial function depending on ω
    compute solution u(ω, x, t) with deterministic solver                 ▷ e.g. solve_network()
    save u(ω, x, t)
end for
interpolate with respect to ω using Lagrange interpolation
for i in range(g) do                                                      ▷ for each grid cell
    compute sample mean and standard deviation
end for
plot the computed sample mean and standard deviation
```

13.5 Numerical Experiments

In the following, we consider numerical examples for the stochastic collocation method using the shallow water equations as the underlying system of equations. First, we consider one-dimensional random spaces that determine the amplitude or width of an initial Gaussian wave on an interval. Afterwards, we apply the sparse grid approach to a dam-break Riemann problem on a network. In both cases, we compute the sample mean and standard deviation for both components.

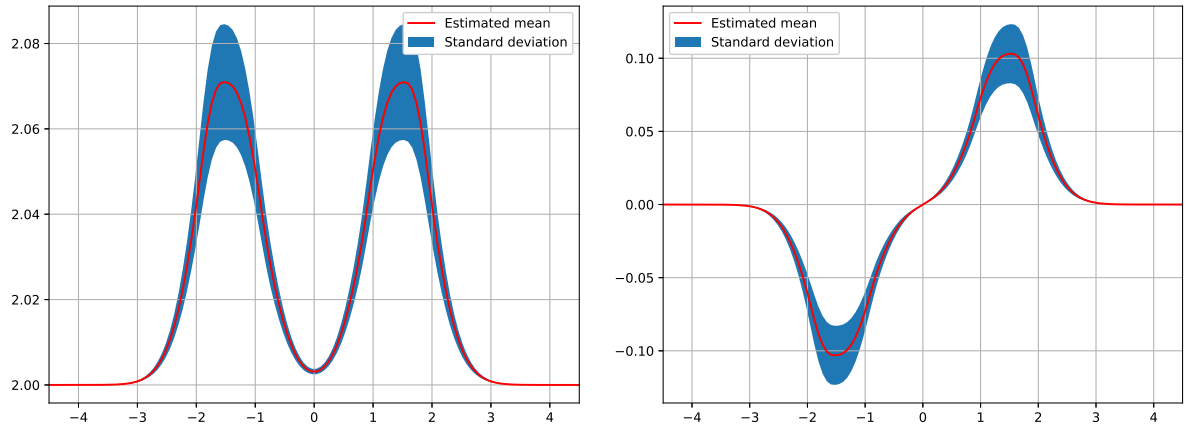


Figure 13.2: Estimated mean and standard deviation for an initial Gaussian wave on an interval with varying amplitude. The left image shows the water height and the right image the momentum.

13.5.1 One-Dimensional Random Space

Consider an initial Gaussian wave of the form (10.14). We repeat the numerical experiment in Section 10.6, i.e. the position of the peak of the initial hump is at $x_0 = 0$ and $h_0 = 2$ is the vertical displacement of the function. However, we now consider the amplitude a or the width s to be a random variable that is uniformly distributed in the interval $\Gamma = [0.1, 0.2]$ or $\Gamma = [0.4, 0.6]$, respectively. We use eleven collocation points in the stochastic collocation method that are given as the extrema of the Chebyshev polynomial of degree twelve, i.e. we apply the deterministic solver eleven times for each problem.
We consider the $TeCNO_2$ scheme on 128 grid cells. Using the Lagrange interpolation formula

$$I_f(\omega, x, t) = \sum_{i=1}^{11} u_i(\omega_i, x, t) L_i(\omega) \tag{13.27}$$

in each cell $\mathcal{C}_j$, $j = 1, \ldots, 128$, we compute the sample mean and standard deviation on 500 sample points uniformly distributed on the random variable space.

Varying amplitude: First, we consider the amplitude a to be a random variable that is uniformly distributed in the interval $\Gamma = [0.1, 0.2]$ and set the width to $s = 0.5$. The resulting sample mean and standard deviation can be found in Figure 13.2. We can clearly see, that the locations of the humps are almost not affected by the varying amplitude. The height of the humps however differs with changing a.

Varying width: Now, we repeat the experiment, but with varying widths. We fix the amplitude $a = 0.15$ and consider the width s as a uniformly distributed random variable in the space $\Gamma = [0.4, 0.6]$. The results are displayed in Figure 13.3. In contrast to the previous experiment, the wave height is now only slightly influenced by the random variable, while it greatly affects the lower widths of the humps and the middle state between them.

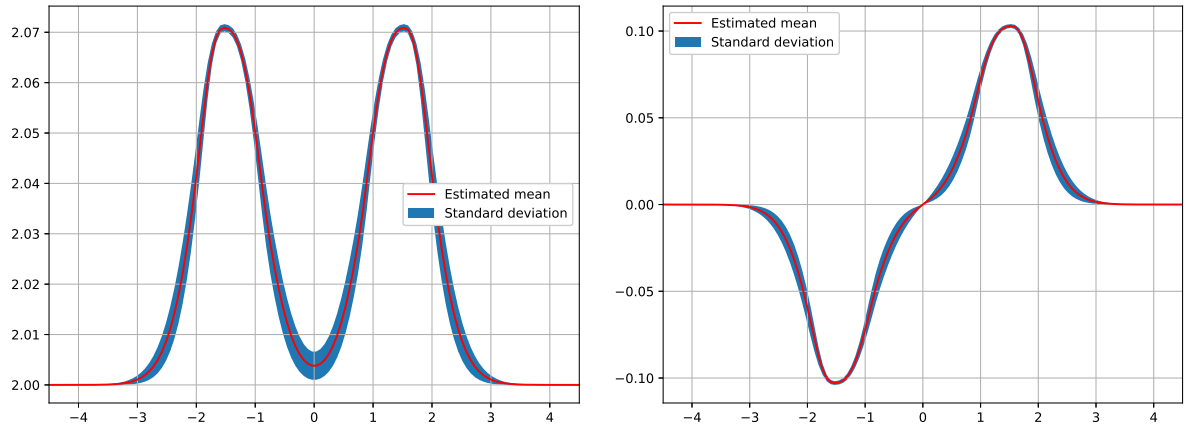


Figure 13.3: Estimated mean and standard deviation for an initial Gaussian wave on an interval with varying width. The left image shows the water height and the right image the momentum.

13.5.2 Multi-Dimensional Random Space

In this section, we apply the stochastic collocation method to the dam-break Riemann problem in networks with initial data

$$h(x,0) = \begin{cases} h_1^* + \omega_1, & x \text{ in Channel } 1 \\ h_2^* + \omega_2, & x \text{ in Channel } 2 \\ h_3^* + \omega_2, & x \text{ in Channel } 3 \end{cases}, \quad w^*(x,0) = 0 \tag{13.28}$$

where $(\omega_1, \omega_2) \in \Gamma$ denotes the random variable.
In our experiment, we set $h_1^* = 2$, $h_2^* = 1.5$ and $h_3^* = 1.5$ and the final time $t = 1$. This problem contains two random variables ω_1, ω_2 and is therefore the smallest possible problem one can apply the Smolyak algorithm to. We fix the number of grid cells in each channel to 128 and apply the TeCNO_2 method in the channels.
We define the random space $\Gamma = [-0.3, 0.3]^2$. To obtain the sparse grid of collocation points, we use the Smolyak algorithm at level $k = 5$, i.e. we obtain the grid depicted on the left in Figure 13.1 with appropriate scaling. Hence, we have 145 executions of the deterministic solver.
Furthermore, we fix the channel widths $s_k = 1$, $k = 1, 2, 3$ and the channel lengths to 2.5 in each channel, the angles $\theta = -\phi = \frac{\pi}{4}$ and use $k = p = 2$ in (7.2).
Note that the choice of the collocation points does not depend on the probability density functions of the random variables. Hence, we compute the estimated mean and standard deviation on the same solutions to the deterministic solver on the sparse grid but for different ρ_i. First, we consider a uniform distribution, i.e. each deviation from the water height h_i^* is equally likely. Since this is not a particularly realistic assumption, we also consider a truncated normal distribution around zero where deviations close to zero from the basic water height h_i^* are more likely than other ones.

Uniform Distribution: First, we assume the probability density functions ρ_i are uniform distributions. The resulting estimated mean and standard deviation can be found in Figure 13.4. We can clearly see that the water height is influenced by the varying initial data on the whole computational domain. This was expected, since the water heights on the left and right are given by the initial water heights, as

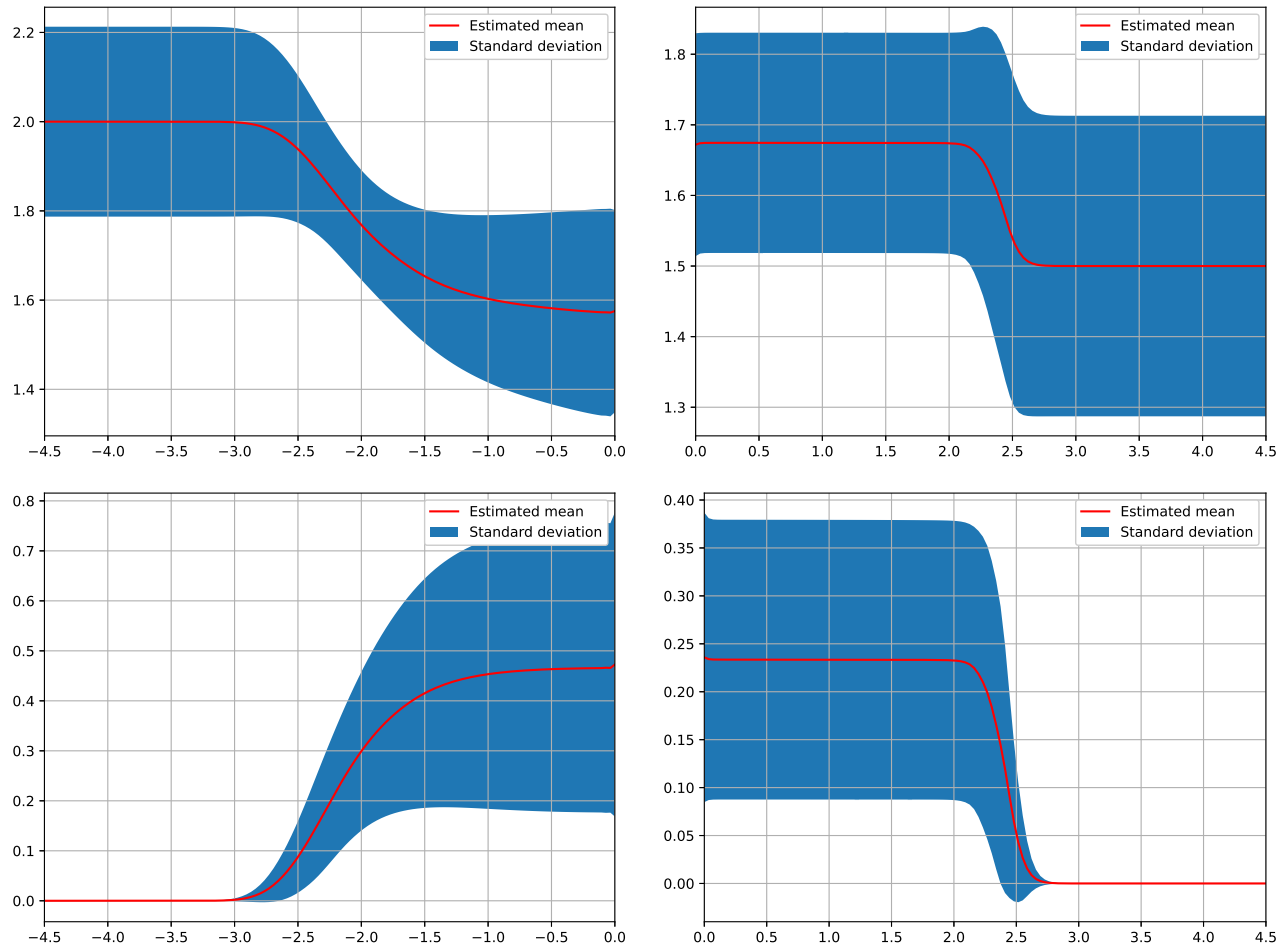


Figure 13.4: Estimated mean and standard deviation for a dam-break Riemann problem with varying initial water heights where the probability density functions are uniform distributions. The left images show Channel 1 while the right images show Channel 2 (which coincides with the solutions on Channel 3). The first row displays the water height, the second the momentum.

long as the rarefaction or shock wave has not reached these points. Furthermore, even though we only computed a solution on 128 grid cells, we can clearly distinguish the rarefaction from the shock wave. The momentum however is not influenced on the whole domain. Since the water only gets accelerated by the rarefaction and shock wave and it is at rest otherwise, the standard deviation is zero in these intervals.

Normal Distribution: Now, we consider a probability density function where small deviations in terms of the absolute value are more likely than higher ones. In a real-world scenario, this is plausible, if the basic water heights h_i^*, $i = 1, 2, 3$ result from measurements since it is more likely to have smaller measurement errors than larger ones. This behavior is usually modeled by a normal distribution. However, since it is not compactly supported, we need to adapt it to our setting by using a *truncated normal distribution* [34]. Here, we choose a normal distribution as "parent function", truncate it to the interval (a, b) and scale it appropriately. For a normal distribution with mean μ and variance σ, this is

defined as

$$\psi(\mu,\sigma,a,b;x) = \begin{cases} 0 & \text{if } x \le a \\ \frac{\phi(\mu,\sigma;x)}{\Phi(\mu,\sigma;b)-\Phi(\mu,\sigma;a)} & \text{if } a < x < b \\ 0 & \text{if } x \ge b \end{cases} \tag{13.29}$$

where $\phi(\mu,\sigma;x)$ denotes the probability density of the normal distribution with mean μ and standard deviation σ, i.e.

$$\phi(\mu,\sigma;x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{13.30}$$

and Φ denotes the *cumulative distribution function*

$$\Phi(\mu,\sigma;x) = \frac{1}{2}(1 + \operatorname{erf}(\frac{\xi}{\sqrt{2}})) \tag{13.31}$$

where $\xi = \frac{x-\mu}{\sigma}$ and $\operatorname{erf}(*)$ denotes the *error function*.
As underlying normal distribution for our experiments, we choose $\mathcal{N}(0, 0.15)$, i.e. a normal distribution with mean $\mu = 0$ and variance $\sigma = 0.15$. Note that they do not necessarily coincide with the mean and standard deviation of the resulting truncated normal distribution. In our case, they are given by $\bar{\mu} = 0$ and $\bar{\sigma} = 0.017$.
The resulting estimated mean and standard deviation can be found in Figure 13.5. As expected, the standard deviation decreased in comparison to the uniform distribution, since the values of the random variables close to zero, i.e. initial water heights $h_i = h_i^* + \omega_i$ close to h_i^* are more likely than other ones. However, apart from that the resulting standard deviations have a similar form than in the previous example.

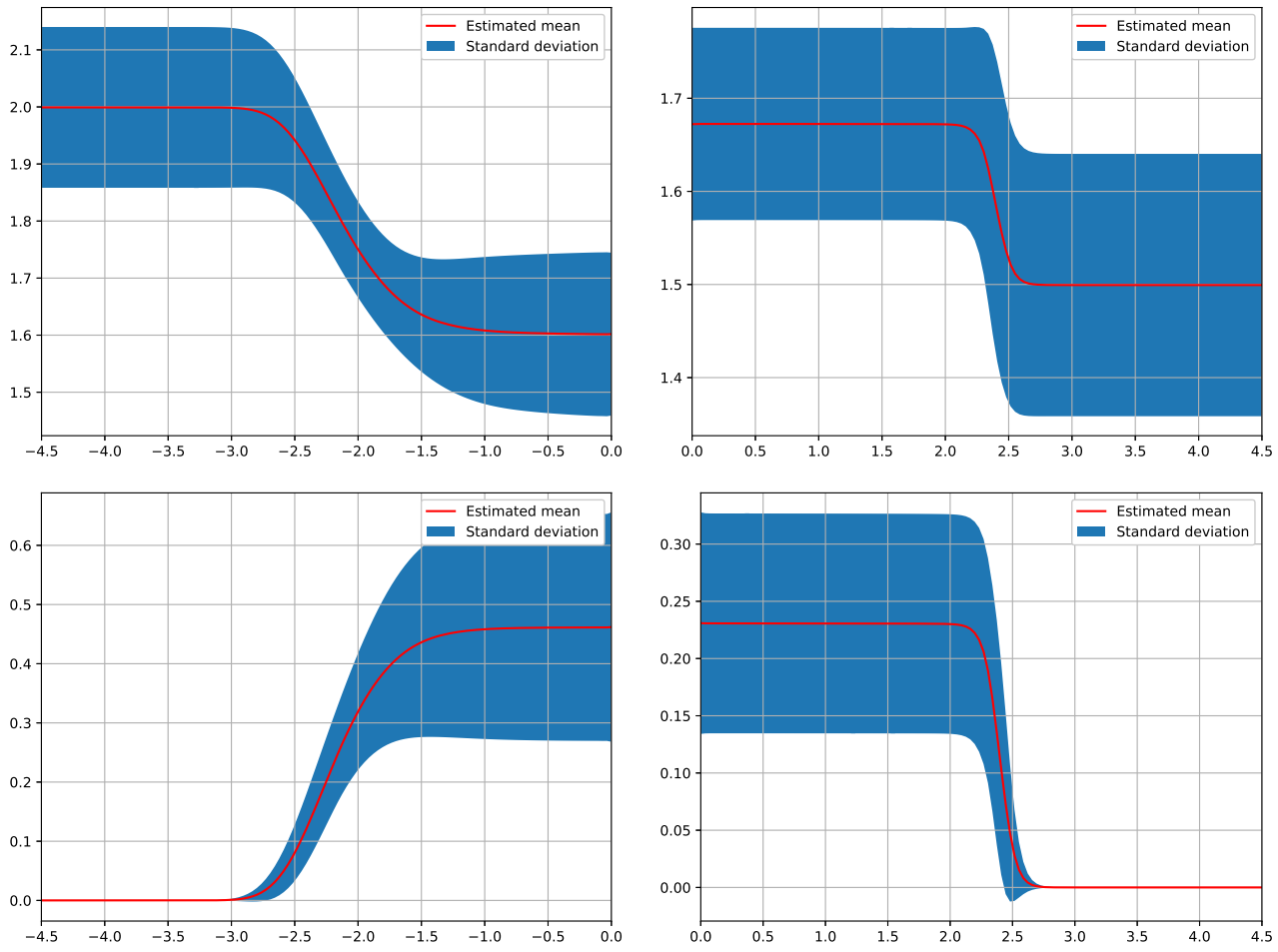


Figure 13.5: Estimated mean and standard deviation for a dam-break Riemann problem with varying initial water heights where the probability density functions are truncated normal distributions. The left images show Channel 1 while the right images show Channel 2 (which coincide with the solutions on Channel 3). The first row displays the water height, the second the momentum.

14 Conclusion and Future Work

In this thesis, we modeled water waves in networks by combining the TeCNO scheme with a nonlinear junction solver. This reduces the originally two-dimensional problem that can be solved by a two-dimensional solver to a one-dimensional problem coupled with a nonlinear system.

First, we presented the TeCNO scheme, a high-order accurate, entropy stable method for solving conservation laws. The main idea for constructing entropy stable fluxes is the combination of entropy conservative fluxes with diffusion terms. We extended them to higher-order methods by coupling linear combinations of second-order entropy conservative methods with particular diffusion terms based on the reconstructed entropy variables. Since the reconstruction procedure needs to fulfill the sign property, we chose the ENO reconstruction.

The numerical examples in Chapter 10 verified the numerical rate of convergence for scalar equations and systems and demonstrated the non-oscillatory property of the TeCNO scheme in comparison to the ELW scheme, which is also high-order accurate and entropy stable.

At the junction, we update the states using a nonlinear system. The equations are derived from Riemann problems across the junction and conservation of mass and momentum in the "junction triangle". However, when modeling waves in a network, the junction solver requires a very specific setting. First, we only consider channels with parallel walls such that the problems there can be interpreted as one-dimensional. The channel width, however, influences the solution of the junction solver, so it does play a role in the solution and cannot be neglected. Furthermore, we assume one junction connects exactly three channels. Nonetheless, the angles the channels form with each other are mutable and also incorporated into the nonlinear system. In our numerical experiments, we investigated the influence of different angles on the resulting wave and found that especially the reflected wave is affected by varying angles. In real-world applications, it is important to consider how different compositions of channel angles or channel widths influence the waves traversing the junction.

In future work, one could extend this simplified problem setting. We did not include a bathymetry term in our equations, i.e. we assumed that the bottom topography is straight and without any elevations or discontinuities since this is a reasonable assumption in artificially built channels. It would be interesting to investigate the influence of a varying ground on the solutions in the network and how that interacts with different channel angles. Furthermore, we assumed that the channel walls are parallel, i.e. each channel has a fixed width since this was necessary to apply a one-dimensional solver in each channel. However, in reality, e.g. in harbours this is not necessarily the case. Apparently, this changes the solution throughout the channels and one has to account for that. Hence, in order to apply this work to a real-world scenario, one has to be aware of the simplifications that have been made and potentially adapt them accordingly.

References

[1] Ulrik Skre Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Arbitrarily high order accurate entropy stable essentially non-oscillatory schemes for systems of conservation laws. *SIAM Journal on Numerical Analysis*, 50(2):544–573, 2012.

[2] Maya Briani, Gabriella Puppo, and Magali Ribot. Angle dependence in coupling conditions for shallow water equations at channel junctions. *Computers and Mathematics with Applications*, 108:49–65, 2022.

[3] David L. George. *Finite Volume Methods and Adaptive Refinement for Tsunami Propagation and Inundation*. PhD thesis, University of Washington, 2006.

[4] Gnenakantanhan Coulibaly, Babacar Leye, Fowe Tazen, Lawani Adjadi Mounirou, and Harouna Karambiri. Urban flood modeling using 2d shallow-water equations in ouagadougou, burkina faso. *Water*, 12(8), 2020.

[5] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press,, 2004.

[6] Jonas Philipp Berberich. *Fluids in Gravitational Fields – Well-Balanced Modifications for Astrophysical Finite-Volume Codes*. PhD thesis, Julius-Maximilians-Universität Würzburg, 2020.

[7] Ulrik Skre Fjordholm. *High-order accurate entropy stable numerical schemes for hyperbolic conservation laws*. PhD thesis, ETH Zurich, 2013.

[8] Randall J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag, 1992.

[9] Helge Holden and Nils Henrik Risebro. *Front Tracking for Hyperbolic Conservation Laws*. Springer Verlag, 2 edition, 2015.

[10] Dongbin Xiu and Jan S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

[11] Akil Narayan, Claude Gittelson, and Dongbin Xiu. A stochastic collocation algorithm with multifidelity models. *SIAM Journal on Scientific Computing*, 36(2):A495–A521, 2014.

[12] S. N. Kružkov. First order quasilinear equations in several independent variables. *Mathematics of the USSR-Sbornik*, 10(2):217, feb 1970.

[13] Peter D. Lax. Hyperbolic systems of conservation laws ii. *Communications on Pure and Applied Mathematics*, 10:537–566, 1957.

[14] James Glimm. Solutions in the large for nonlinear hyperbolic systems of equations. *Communications on Pure and Applied Mathematics*, 18:697–715, 1965.

[15] Alberto Bressan, Tai-Ping Liua, and Tong Yang. L1 stability estimates for n×n conservation laws. *Archive for Rational Mechanics and Analysis*, 1999.

[16] Alberto Bressan, Graziano Crasta, and Benedetto Piccoli. Well-posedness of the cauchy problem for *ntimesn* systems of conservation laws. *Mem. Amer. Math. Soc.*, 146:viii+134, 01 2000.

[17] Peter Lax. Shock waves and entropy. In Eduardo H. Zarantonello, editor, *Contributions to Nonlinear Functional Analysis*, pages 603–634. Academic Press, 1971.

[18] Amiram Harten, James Hyman, Peter Lax, and Barbara Keyfitz. On finite-difference approximations and entropy conditions for shocks. *Communications on Pure and Applied Mathematics*, 29:297–322, 05 1976.

[19] Stanley Osher. Riemann solvers, the entropy condition, and difference approximations. *SIAM Journal on Numerical Analysis*, 21(2):217–235, 1984.

[20] Eitan Tadmor. Numerical viscosity and the entropy condition for conservative difference schemes. *Mathematics of Computation*, 43(168):369–381, 1984.

[21] Eitan Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. i. *Mathematics of Computation*, 49(179):91–103, 1987.

[22] P. G. LeFloch, J. M. Mercier, and C. Rohde. Fully discrete, entropy conservative schemes of arbitrary order. *SIAM Journal on Numerical Analysis*, 40(5):1968–1992, 2002.

[23] Ulrik Skre Fjordholm. Structure preserving finite volume methods for the shallow water equations. Master's thesis, University of Oslo, Norway, 2009.

[24] Ulrik Fjordholm, Siddhartha Mishra, and Eitan Tadmor. *Energy Preserving and Energy Stable Schemes for the Shallow Water Equations*, volume 363 of *London Mathematical Society Lecture Note Series*, page 93–139. Cambridge University Press, 2009.

[25] Timothy J. Barth. *Numerical Methods for Gasdynamic Systems on Unstructured Meshes*, pages 195–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[26] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. *Journal of Computational Physics*, 131(1):3–47, 1997.

[27] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM REVIEW*, 43(1):89–112, 2001.

[28] Neelabja Chatterjee. *Numerical analysis of conservation laws involving non-local terms*. PhD thesis, University of Oslo, 2019.

[29] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag, 5 edition, 1998.

[30] P.J. Davis. *Interpolation and Approximation*. Dover Books on Mathematics. Dover Publications, 1975.

[31] G.G. Lorentz. *Approximation of Functions*. Approximation of Functions. Holt, Rinehart and Winston, 1966.

[32] Erich Novak and Klaus Ritter. Simple cubature formulas with high polynomial exactness. *Constructive Approximation*, 15:499–522, 12 1999.

[33] Erich Novak and Klaus Ritter. High dimensional integration of smooth functions over cubes. *Numerische Mathematik*, 75:79–97, 10 1996.

[34] Ion Mihoc and Cristina Fătu. Fisher's information measure and truncated normal distributions (ii). *Revue d'Analyse Numérique et de Théorie de l'Approximation*, 32, 01 2003.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Versicherung zur selbständigen Leistungserbringung

Titel der Masterarbeit :

Simulating Water Waves in Networks Using a High-Order Accurate Entropy Stable Finite Volume Scheme

Thema bereitgestellt von (Titel, Vorname, Nachname, Lehrstuhl):

Prof. Dr. Christian Klingenberg, Lehrstuhl für Mathematik VI

Eingereicht durch (Vorname, Nachname, Matrikel):

Veronika Mayerhofer, 2300311

Ich versichere, dass ich die vorstehende schriftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die benutzte Literatur sowie sonstige Hilfsquellen sind vollständig angegeben. Wörtlich oder dem Sinne nach dem Schrifttum oder dem Internet entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

☐ Mit dem Prüfungsleiter bzw. der Prüfungsleiterin wurde abgestimmt, dass für die Erstellung der vorgelegten schriftlichen Arbeit Chatbots (insbesondere ChatGPT) bzw. allgemein solche Programme, die anstelle meiner Person die Aufgabenstellung der Prüfung bzw. Teile derselben bearbeiten könnten, eingesetzt wurden. Die mittels Chatbots erstellten Passagen sind als solche gekennzeichnet.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Würzburg, 04.10.23 V. Mayerhofer
Ort, Datum, Unterschrift