Chapter 1

Introduction

This chapter provides the motivation for the project and explains the background and key objectives of the thesis. It also includes a literature review, where we outline the most important sources relevant to this work. Finally, this chapter offers an overview of the thesis and its structure.

1.1 Motivation

Natural distasters such as floods and tsunamis are becoming increasingly frequent, causing devastating impacts on human life and property. Populations in coastal areas, river basins and flood-prone regions are particularly at risk. A recent example is the catastrophic flooding in Valencia, Spain, in late October 2024. On October 29, 2024, Valencia received a year's worth of rain in just eight hours, leading to flash floods that devastated the area, resulting in significant loss of life and property damage [1]. Satellite images highlight the extent of the flooding, comparing the region before the event on October 8, 2024, and after on October 30, 2024. These images are presented in Figure 1.1.

While it is impossible to prevent such disasters from occuring, trustworthy forecasts delivered in sufficiently short time can help in emergency response and disaster management. These events highlight the urgent need for efficient tools to simulate and understand flood behavior. The ability to rapidly simulate flood scenarios can significantly reduce the time needed for decision-making in critical situations, potentially saving lives and minimizing the impact of such disasters. During a catastrophe, fast simulatons are particularly valuable, even if some accuracy is sacrificed, as long as the results are accurate enough to make decisions. Moreover, running multiple simulations of different scenarios can provide a better understanding of water dynamics and a stronger data foundation.

For instance, the flooding in Germany during July 2021 was unexpected in several ways. Meteorological forecasts has predicted heavy rain in the affected regions, and there were warnings of the potential for flooding [2]. Despite these forecasts, the intensity and scale of the floods surpassed expectations, and climate scientist expressed shock of the extent of the flooding [3]. This is a good example of how a deeper understanding of water dynamics, supported by simulations, could have helped to predict the water flow and its spread.

This motivates the study of the shallow water equations (SWE), a set of hyperbolic partial differential equations that describe the motion of a fluid in a shallow layer of water. These equations are essential for understanding and simulating water dynamics in shallow water regions, such as coastal areas, rivers, and flood-prone regions. A wide range of problems can be modeled by the shallow water equations, such as flooding, tsunamis and dam break scenarios. By solving the shallow water equations, we can simulate and predict the behavior of water in these regions, providing valuable insights for disaster management and emergency response. In this work, we will derive

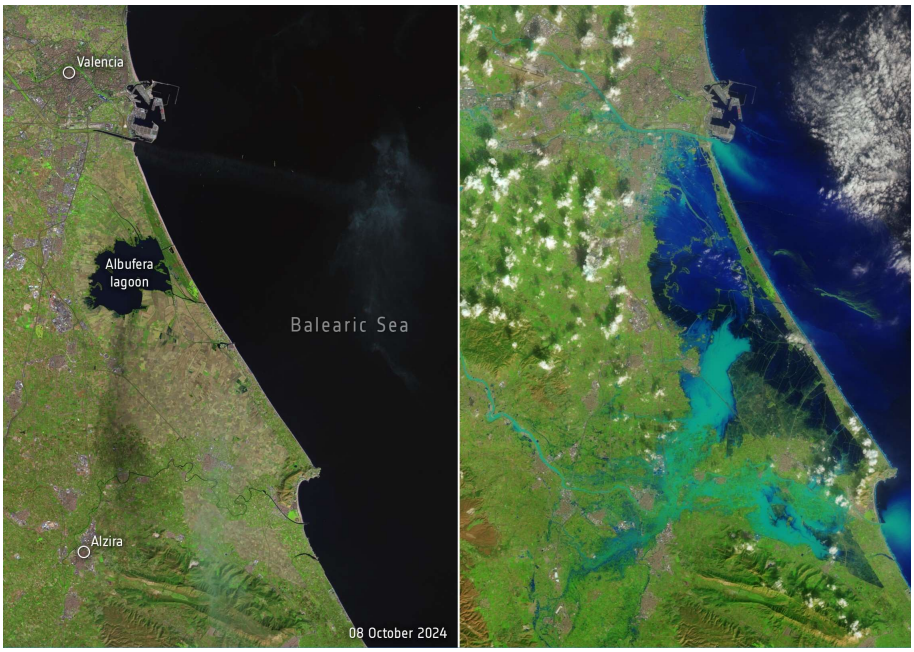


Figure 1.1: Before and after the floods in Valencia, Spain, October 2024.
Source: `https://www.esa.int/ESA_Multimedia/Images/2024/10/Valencia_flood_disaster`.

the SWE with one spatial dimension (1D), two spatial dimensions (2D), and in spherical coordinates, which are particularly useful for modeling water flow on the Earth's surface.

The SWE can be solved using numerical methods, such as the finite volume method (FVM), which is a numerical technique for solving partial differential equations, by discretizing the domain into small control volumes and integrating the equations over these volumes. This method is widely used in computational fluid dynamics to model fluid behavior. In this work, we will implement the FVM in 1D and use it to solve the SWE for several scenarios, including the dam break problem. We will extent the FVM to 2D and solve the idealized circular dam break problem. The FVM solvers will be validated against known test cases, as this validation is critical for future work. A finer grid resolution improves solution accuracy but comes at the cost of an increased computational run time, resulting in slower simulations. While high-resolution grids can provide more accurate results, they may not be practical for real-time simulations, such as during a flood event, where fast simulations are crucial for decision-making. This limitation motivates the investigation of data-driven approaches to solve the shallow water equations, which may offer a faster and more efficient way to simulate water dynamics while maintaining an acceptable level of accuracy.

This project investigates whether data-driven methods can provide a faster and more efficient alternative to traditional numerical methods for solving the shallow water equations. Specifically, we will explore training a convolutional neural network (CNN) and a Fourier neural operator (FNO) to solve the SWE. The data-driven models will be trained on the data generated from the FVM solvers, and evaluated on performance metrics such as run time, accuracy, long-term prediction capabilities, grid transferability and their response to new initial conditions. These initial conditions could include varying water heights, velocities, and other environmental factors that may change in real-world flood scenarios. We will compare the advantages and limitations of the data-driven approaches against numerical methdos, focusing on their potential to improve disaster responser and flood prediction.This comparison will allow us to assess whether data-driven approaches offer a practical solution to efficiently solve the shallow water equations. Data-driven models may be preferred for applications requiring fast simulations, while numerical methods may be more suitable for scenarios demanding high accuracy. An interesting aspect is to investigate how much more accurate and efficient the data-driven methods may be compared to the traditional

numerical methods.

Additionally, we will examine the different models abilities to make long-term predictions. Numerical methods, including the FVM, solve the SWE one time step at a time, which can be computationally expensive, particularly for long-term predictions. In constrast, data-driven models also predict one time step at a time, but they may not need to be retrained for each time step. Once trained, these models can make predictions for all time steps, potentially making them faster than numerical methods.

By analyszing and comparing these approaches, this project aims to identify the contextx in which data-driven models can complement or even replace traditional numerical methods for solving the shallow water equations.

1.2 Literature

When working in this area it is inevitable to mention the work of E. F. Toro, who has written several books on the topic of Riemann solvers and the finite volume method, specifically for the shallow water equations. In this project, we will use the books *Shock-Capturing Methods for Free-Surface Shallow Flows* [4], *Riemann Solvers and Numerical Methods for Fluid Dynamics* [5] and the rather new book from 2024 *Computational Algorithms for Shallow Water Equations* [6] as references. The books have been especially useful when deriving the shallow water equations as well as understanding and describing the finite volume method, inlcuding the Riemann solvers used in this project. The course *Advanced Numerical Methods for Environmental Models* at the University of Trento, has provided a good foundation for the numerical methods used in this project, both in terms of lecture notes and exercises [7].

Working with the shallow water equations in spherical coordinates, the papers *Well-balanced methods for the shallow water equations in spherical coordinates* by Castro et al. [8] and *Physics-informed neural networks for the shallow-water equations on the sphere* by Bihlo et al. [9] are references for deriving the SWE in spherical coordinates. Additionally, the lecture notes *Shallow water on a sphere* by Raymond from New Mexico Tech [10] and the notes from Geophysical Fluid Dynamics Laboratory [11] have been valuable in this derivation. Furthermore, the papers by Gavete [12] and Galewsky [13] also provide important insights into the spherical shallow water equations. Implementing the FVM to solve the shallow water equations in spherical coordinates is a challenging task, as exact literature on the topic is limited. However, some sources discussing the discontinuous Galerkin scheme [14] for solving the spherical shallow water equations have been useful in this context.

The field of Fourier neural operators (FNO) is relatively new, and as a result, there is limited literature on the topic. However, the paper *Fourier Neural Operator for Parametric Partial Differential Equations* [15], written by several authors, is a key reference. The paper suggests that FNOs has shown great potential in being an efficient and resolution independent operator in the emerging field of scientific machine learning to solve partial differential equations. A key reason of their success is their ability to generalize to unseen data. For more on the topic of scientific machine learning, consider the tech report [16]. In the last years the company Nvidia has done some very interesting work on the topic of FNO, and they have published several blog posts on the topic. One of the posts consider the use of spherical Fourier neural operators (SFNO) to generate weather forecasts around the globe [17]. Another paper regarding SFNOs is [18], which generalizes FNOs on the sphere. Convolutional neural networks (CNNs) are a well-researched area, with numerous sources available on the topic. For this project, the sources [19] and [20] have been particularly helpful in explaining the theoretical foundations of the methods.

1.3 Thesis overview

The rest of tge thesis is structured as follows. In chapter 2, we derive the shallow water equations (SWE) in 1D, 2D, and spherical coordinates. In chapter 3, we present the finite volume method (FVM) used to solve the shallow water equations, including the Riemann problem and the numerical fluxes essential for the FVM. These chapters

where r is the radius, (θ, ϕ) are the longitude and latitude angles, h is the height of the water, u_θ and u_ϕ are the velocities in the $\theta-$ and $\phi-$directions and g is the gravitational constant. We can also write the spherical SWE (2.5.17) in vector form as

$$\mathbf{U}_t + \frac{1}{r\cos(\phi)}\mathbf{F}(\mathbf{U})_\theta + \frac{1}{r}\mathbf{G}(\mathbf{U})_\phi = \mathbf{S}, \tag{2.5.18}$$

where $\frac{1}{r\cos(\phi)}\mathbf{F}(\mathbf{U})_\theta$ and $\frac{1}{r}\mathbf{G}(\mathbf{U})_\phi$ are the flux terms in the $\theta-$ and $\phi-$directions and $\mathbf{S}$ is the source term.

2.6 The Linearized Shallow Water Equations in Spherical Coordinates

In this section, we derive the linearized shallow water equations (LSWE) in spherical coordinates, focusing on one spatial dimension, namely the longitude θ. This approach assumes a constant latitude ϕ, simplifying the equations to a one-dimensional framework. The LSWE are employed for their simplicity, providing a foundation for understanding wave dynamics in a spherical geometry. Later in this project, we will apply the finite volume method to solve these equations numerically. The derivation and methodology presented here are based on the sources [26] and [27]. We neglect all body forces other than gravity. We linearize by assuming that the height of the water h and the velocities u are small perturbations from a state of rest. We set $h = h_0 + h'$ and $u = u_0 + u'$, where h_0 and u_0 are the equilibrium height and velocity, h' and u' are the perturbations. We also assume that the equilibrium height h_0 is constant and that the equilibrium velocity u_0 is zero. Let the perturbations in water height and velocity be represented by:

$$h = h_0 + h', \quad u = u_0 + u', \tag{2.6.1}$$

where h_0 is the mean water height, and h' and u' represent the perturbations in the water height and velocity components, respectively. Since we consider the LSWE for one spatial dimension, we neglect the latitude component ϕ and the velocity component u_ϕ. The third equation in (2.5.17) is neglected, as we consider the LSWE for one spatial dimension. We also neglect all forces, except for the gravitational force. Thus we have the following SWE in spherical coordinates for one spatial dimension:

$$\left.\begin{aligned} h_t + \frac{1}{r\cos(\phi)}(hu)_\theta = 0, \\ u_t + \frac{u}{r\cos(\phi)}u_\theta + \frac{g}{r\cos(\phi)}h_\theta = 0. \end{aligned}\right\} \tag{2.6.2}$$

Where u denotes the velocity in the θ direction. We substitute the perturbed variables (2.6.1) into the 1D spherical SWE (2.6.2) and utilize that $u_0 = 0$ to obtain

$$\left.\begin{aligned} (h_0 + h')_t + \frac{1}{r\cos(\phi)}\left((h_0 + h')(u')_\theta + (h_0 + h')_\theta(u')\right) = 0, \\ (u')_t + \frac{u'}{r\cos(\phi)}(u')_\theta + \frac{g}{r\cos(\phi)}h'_\theta = 0. \end{aligned}\right\} \tag{2.6.3}$$

We neglect the terms involving products of pertubations arising in (2.6.3), as they are second-order terms. Hence, we obtain the linearized shallow water equations in spherical coordinates with one spatial dimension θ and time t as:

$$\left.\begin{aligned} h'_t + \frac{h_0}{r\cos(\phi)}u'_\theta = 0, \\ u'_t + \frac{g}{r\cos(\phi)}h'_\theta = 0. \end{aligned}\right\} \tag{2.6.4}$$

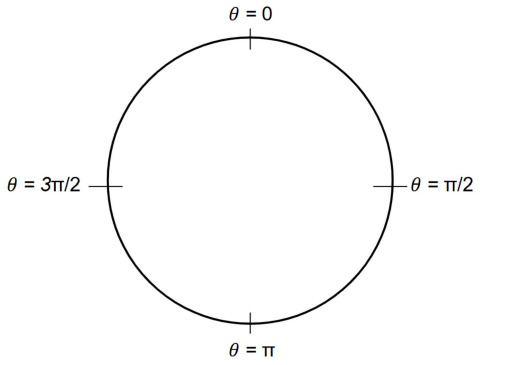
These equations describe the evolution of small perturbations in water height and velocity on a spherical surface. We can also write the linearized shallow water equations in spherical coordinates in vector form as

$$\mathbf{W}_t + \mathbf{A}\mathbf{W}_\theta = 0, \tag{2.6.5}$$

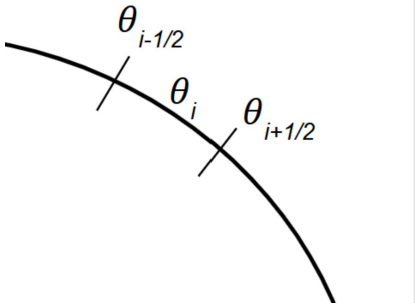
where $\mathbf{W} = \begin{bmatrix} h' \\ u' \end{bmatrix}$ and the coefficient matrix $\mathbf{A}$ is constant and given as: $\mathbf{A} = \begin{bmatrix} 0 & \frac{h_0}{r\cos(\phi)} \\ \frac{g}{r\cos(\phi)} & 0 \end{bmatrix}$.

Integral form of the 1D spherical LSWE

As we consider the longitude θ as the spatial dimension, the domain is the circle $[0, 2\pi]$, which is divided into N cells or control volumes, each of length $\Delta\theta = \frac{2\pi}{N}$, expressend in terms of longitude angle. Each cell i has a cell center at θ_i and cell boundaries/interfaces at $\theta_{i-1/2}$ and $\theta_{i+1/2}$.



(a) Illustration of the grid for the 1D SWE with small cells.



(b) Illustration of the grid for the 1D SWE with a small domain.

Figure 2.7: Grid illustrations for the 1D SWE in spherical coordinates.

As when deriving the finite volume scheme for the 1D SWE in cartesian coordinates, we start by integrating the linearized shallow water equations over the control volume, and then divide by the cell length to obtain the finite volume scheme. We integrate the vector form of the 1D SWE in spherical coordinates without the coriolis force over θ from $\theta_L := \theta_i - \frac{1}{2}\Delta\theta$ to $\theta_R := \theta_i + \frac{1}{2}\Delta\theta$ to obtain

$$\int_{\theta_L}^{\theta_R} \mathbf{W}_t \, \mathrm{d}\theta + \int_{\theta_L}^{\theta_R} \mathbf{A}\mathbf{W}_\theta \, \mathrm{d}\theta = 0.$$

Since the matrix $\mathbf{A}$ is constant, we can take it out of the integral:

$$\int_{\theta_L}^{\theta_R} \mathbf{W}_t \, \mathrm{d}\theta + \mathbf{A} \int_{\theta_L}^{\theta_R} \mathbf{W}_\theta \, \mathrm{d}\theta = 0.$$

We also use the fundamental theorem of calculus to rewrite the integral of the derivative as the difference of the function at the boundaries of the control volume:

$$\int_{\theta_L}^{\theta_R} \mathbf{W}_t \, \mathrm{d}\theta = \mathbf{A} \left(\mathbf{W}(\theta_L, t) - \mathbf{W}(\theta_R, t) \right). \tag{2.6.6}$$

We can also write the equations out to get a better understanding of the terms:

$$\left.\begin{aligned} \frac{\partial}{\partial t}\int_{\theta_L}^{\theta_R} h \ \mathrm{d}\theta + \frac{h_0}{r\cos(\phi)}(u_R - u_L) = 0, \\ \frac{\partial}{\partial t}\int_{\theta_L}^{\theta_R} u \ \mathrm{d}\theta + g(h_R - h_L) + f u_i = 0. \end{aligned}\right\} \tag{2.6.7}$$

Then we integrate (2.6.6) over time from $t_1 := t_n$ to $t_2 := t_{n+1}$:

$$\int_{t_1}^{t_2}\int_{\theta_L}^{\theta_R} \mathbf{W}_t \ \mathrm{d}\theta \ \mathrm{d}t = \mathbf{A}\left(\int_{t_1}^{t_2} \mathbf{W}(\theta_L, t) \ \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{W}(\theta_R, t) \ \mathrm{d}t\right).$$

Rewriting, using the fundamental theorem of calculus, gives

$$\int_{\theta_L}^{\theta_R} \mathbf{W}(\theta, t_2) \ \mathrm{d}\theta = \int_{\theta_L}^{\theta_R} \mathbf{W}(\theta, t_1) \ \mathrm{d}\theta - \mathbf{A}\left(\int_{t_1}^{t_2} \mathbf{W}(\theta_R, t) \ \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{W}(\theta_L, t) \ \mathrm{d}t\right), \tag{2.6.8}$$

which we refer to as the integral form of the linearized shallow water equations in spherical coordinates with one spatial dimension and time. The integral form (2.6.8) is the foundation for the finite volume method, which we will use to solve the linearized shallow water equations in spherical coordinates numerically.

piecewise first-degree polynomials. This reconstruction provides a higher-order representation of the solution within each cell. In this project, we use the MUSCL scheme with the TVD criteria to ensure smooth and stable solutions. A numerical method $U_{i,j}^{n+1}$ is called TVD if it satisfies the following condition [29]:

$$TV(U^{n+1}) \leq TV(U^n), \tag{3.2.3}$$

for all grid functions U^n. For a discrete function $U_{i,j}$ defined on a grid $i = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, N$, the total variation $TV(U)$ is defined as

$$TV(U) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |U_{i+1,j} - U_{i,j}| + |U_{i,j+1} - U_{i,j}|. \tag{3.2.4}$$

The total variation measures the sum of the absolute differences between neighboring grid points. A TVD scheme ensures that the total variation of the solution does not increase in time, which is crucial for maintaining stability and preventing oscillations. The TVD property is used in shock-capturing schemes to prevent nonphysical oscillations near discontinuities, helping to accurately capture the shock location.

3.3 FVM 1D Linearized SWE spherical

In this section, we derive the finite volume method for the 1D shallow water equations in spherical coordinates, where we consider the linearized shallow water equations on a circle. Since we now work on a circle, we must impose periodic boundary conditions. We follow the same method as in section 3.1. We begin by stating the integral form of the 1D LSWE in spherical coordinates on a global domain. Each cell has length $\Delta\theta = \theta_{i+1/2} - \theta_{i-1/2}$. That is, we rewrite (2.6.8) to be in global variables:

$$\int_{\theta_{i-1/2}}^{\theta_{i+1/2}} \mathbf{W}(\theta, t_{n+1}) \, \mathrm{d}\theta = \int_{\theta_{i-1/2}}^{\theta_{i+1/2}} \mathbf{W}(\theta, t_n) \, \mathrm{d}\theta - \mathbf{A} \left(\int_{t_n}^{t_{n+1}} \mathbf{W}(\theta_{i+1/2}, t) \, \mathrm{d}t - \int_{t_n}^{t_{n+1}} \mathbf{W}(\theta_{i-1/2}, t) \, \mathrm{d}t \right). \tag{3.3.1}$$

We divide the integral form (3.3.1) with the cell length $\Delta\theta$ to obtain

$$\begin{aligned} \frac{1}{\Delta\theta} \int_{\theta_{i-1/2}}^{\theta_{i+1/2}} \mathbf{W}(\theta, t_{n+1}) \, \mathrm{d}\theta = \frac{1}{\Delta\theta} \int_{\theta_{i-1/2}}^{\theta_{i+1/2}} \mathbf{W}(\theta, t_n) \, \mathrm{d}\theta \\ - \frac{\Delta t}{\Delta\theta} \mathbf{A} \left(\frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{W}(\theta_{i-1/2}, t) \, \mathrm{d}t - \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{W}(\theta_{i+1/2}, t) \, \mathrm{d}t \right). \end{aligned}$$

Averaging over the terms for a finite volume gives the first-order explicit time-stepping finite volume scheme:

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \frac{\Delta t}{\Delta\theta} (\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n). \tag{3.3.2}$$

The scheme uses the cell averages:

$$\mathbf{W}_i^n = \frac{1}{\Delta\theta} \int_{\theta_{i-1/2}}^{\theta_{i+1/2}} \mathbf{W}(\theta, t_n) \, \mathrm{d}\theta$$

The flux $\mathbf{F}_{i-1/2}^n$ is the average flux across the line $\theta = \theta_{i-1/2}$ from time t_n to t_{n+1}:

$$\mathbf{F}_{i-1/2}^n = \frac{1}{\Delta t} \mathbf{A} \int_{t_n}^{t_{n+1}} (\mathbf{W}(\theta_{i-1/2}, t)) \, \mathrm{d}t,$$

and correspondingly the flux $\mathbf{F}_{i+1/2}^n$ is the average flux across the line $\theta = \theta_{i+1/2}$ from time t_n to t_{n+1}:

$$\mathbf{F}_{i+1/2}^n = \frac{1}{\Delta t} \mathbf{A} \int_{t_n}^{t_{n+1}} (\mathbf{W}(\theta_{i+1/2}, t)) \, \mathrm{d}t.$$

The explicit Runge-Kutta 4th order method (ERK4) is a numerical technique used to solve ordinary differential equations (ODEs). This method is particularly effective for time-stepping and is an alternative approach to traditional time integration schemes. RK4 provides higher accuracy in time while relying on the same spatial discretization, such as the finite volume method (FVM). In this context, the RK4 method is applied to numerically solve the linearized 1D shallow water equations in spherical coordinates:

$$\frac{\partial h'}{\partial t} = -\frac{h_0}{r\cos\phi}\frac{\partial u}{\partial \theta}, \tag{3.3.3a}$$

$$\frac{\partial u}{\partial t} = -g\frac{\partial h'}{\partial \theta} - fu, \tag{3.3.3b}$$

where h' is the perturbation in water height, u is the velocity, h_0 is the mean water depth, g is the gravitational acceleration, f is the Coriolis parameter, r is the Earth's radius, and ϕ is the fixed latitude. The ERK4 method employs a four-stage time-stepping scheme to update the solution for h' and u. The general update formula is:

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n + \frac{\Delta t}{\Delta\theta}\frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

where the state variable $\mathbf{W}_i = \begin{bmatrix} h_i' \\ u_i \end{bmatrix}$ represents the pertubation height and velocity at the $i-$th cell. The intermediate stages k_1, k_2, k_3 and k_4 are computed at each time step. The scheme integrates the equations in time while relying on flux differences to update the solution. At each time step, the fluxes $\mathbf{F}_i$ between neighboring cells are computed as

$$\mathbf{F}_i = \frac{1}{2}(\mathbf{F}_{i-1/2} + \mathbf{F}_{i+1/2}),$$

where $\mathbf{F}_{i-1/2}$ and $\mathbf{F}_{i+1/2}$ are the fluxes at the edges of the adjacent cells. These fluxes depend on h' and u. At each ERK4 stage, the fluxes and intermediate derivatives are calculated to ensure accurate time integration. The final values of h' and u are obtained by combining the contributions from all four stages. This approach allows the method to achieve high temporal accuracy while maintaining the spatial resolution provided by the finite volume discretization.

3.4 The Riemann problem

We will now define the Riemann problem, since it plays a crucial role in the finite volume method. In the Riemann problem we distinguish between what we call a wet bed and a dry bed. A wet bed is the case where the water depth is positive everywhere, whereas a dry bed is the case where the water depth is zero in some cells. The special Riemann problem where parts of the bed are dry is dealing with the so-called dry fronts or wet/dry fronts, which are challenging to handle numerically. We will leave these cases for now, and only consider wet bed problems. The Riemann problem for the shallow water equations in 1D with a zero source term is defined as the initial-value problem (IVP) [6]:

$$\begin{aligned} &\text{PDEs: } \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \\ &\text{ICs: } \mathbf{U}(x,0) = \begin{cases} \mathbf{U_L}, & \text{if } x < 0, \\ \mathbf{U_R}, & \text{if } x > 0. \end{cases} \end{aligned} \tag{3.4.1}$$

The vectors $\mathbf{U}$ and $\mathbf{F}(\mathbf{U})$ in (3.4.1) are given by

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ hvu \end{bmatrix}, \tag{3.4.2}$$

and the initial conditions $\mathbf{U_L}$ and $\mathbf{U_R}$ are

$$\mathbf{U_L} = \begin{bmatrix} h_L \\ h_L u_L \\ h_L v_L \end{bmatrix}, \quad \mathbf{U_R} = \begin{bmatrix} h_R \\ h_R u_R \\ h_R v_R \end{bmatrix},$$

which represents the conditions at time $t = 0$ s in the left and right states of $x = 0$ m. The function $\mathbf{U}$ is piecewise constant, with a discontinuity at $x = 0$ m. The Riemann problem can be solved either exactly or approximately. However, in this project we will focus on approximate Riemann solvers, which are able to solve the Riemann problem with high accuracy and efficiency. Various approximate Riemann solvers exist, based on finding an approximate solution to the Riemann problem. Some of these solvers will be considered in the next section. An interesting example of a Riemann problem is the so-called dam break problem, which is presented next.

3.4.1 The Dam-Break problem

We now introduce the dam-break problem, a scenario of significant physical interest. This problem models the sudden release of water following the collapse of a dam, making it highly relevant for studying natural disasters such as floods and tsunamis. As a classic test case for numerical methods, the dam-break problem is commonly used to test the ability of a method to capture discontinuities in the solution. The dam-break problem is a special case of the Riemann problem (3.4.1). The difference is that in the dam-break problem, the initial velocity components, u_L, u_R, v_L and v_R, are zero, whereas in the Riemann problem they are allowed to be distinct from zero. The initial setup is visualized in Figure 3.4.

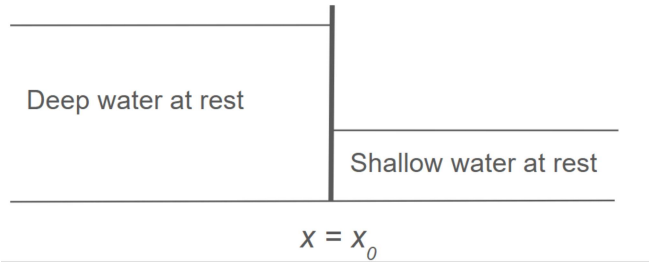


Figure 3.4: Initial conditions for the dam-break problem. An infinitely thin wall at $x = x_0$ divides two sections of water with different water levels.

We can use the shallow water equations to model the flow of water in the dam-break problem, approximately, if we assume that the wall collapses instantaneously at $t = 0$ s. In this project we solve both the dam-break problem and cases of the Riemann problem, where the inital fluid velocity is nonzero. The results are presented in chapter 6.

3.4.2 Solving the Riemann problem exactly by wave decomposition

To get a better understanding of the flow in shallow water, we provide some very short background information about the wave structures in the solution of the Riemann problem. In general, the wave structure in the solution of the Riemann problem (3.4.1) consists of three wave families separating four regions. The wave families are denoted W_1, W_2, W_3 and the regions are the spaces between the wave families, denoted by R_0, R_1, R_2 and R_3. The wave structure is illustrated in Figure 3.5.

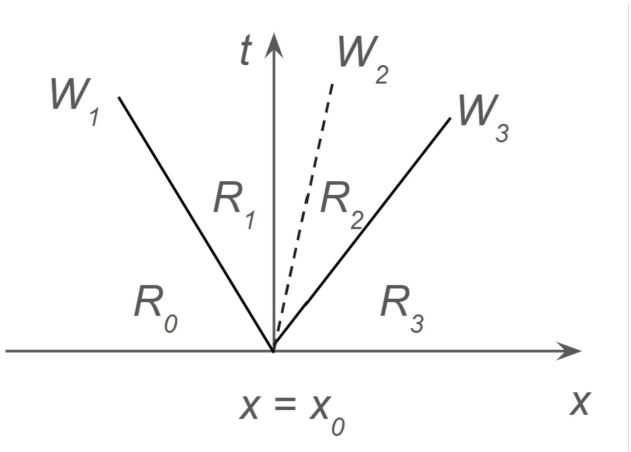


Figure 3.5: General wave structure in the solution of the Riemann problem.

From Figure 3.5 we see how the solution consists of three waves. The left and right waves are either shock waves or rarefaction waves, and correspond to the one-dimensional shallow water equations. The middle wave arises from the $y-$momentum equation in (3.4.1) and is always a shear wave. The region between the left and right wave is called the star region and is interesting, since we do not know the solution in this region. The star region is divided into two subregions R_1 and R_2. The states in the regions are (from left to right) U_L, U_{*L}, U_{*R} and U_R, where U_L and U_R are known, as these are the initial conditions. The states U_{*L} and U_{*R} are in the regions R_1 and R_2, i.e., the star region and are unknown. In the star region, we use h_* to denote the water depth and u_* to denote the velocity. We can use h_* to determine whether the left and right waves are shock waves or rarefaction waves. Since we are considering the wet bed case, we can use the following characteristic:

$$\begin{cases} \text{The left wave is a shock wave} & \text{if } h_* > h_L, \\ \text{The left wave is a rarefaction wave} & \text{if } h_* \leq h_L, \end{cases}$$

and similarly for the right wave:

$$\begin{cases} \text{The right wave is a shock wave} & \text{if } h_* > h_R, \\ \text{The right wave is a rarefaction wave} & \text{if } h_* \leq h_R. \end{cases}$$

A shock wave is characterized by a discontinuity in the solution. On each side of the shock wave, the water properties, such as height and velocity, differ significantly. In contrast, a rarefaction wave represents a smooth transition in the solution. The water height and velocity change gradually across the wave, and the properties are more similar on each side of the wave, without the sharp edges seen in a shock wave. In the solution of the Riemann problem (3.4.1) there are four possible wave patterns outcomes, which are combinations of shock waves and rarefaction waves. The left and right waves are either shock waves or rarefaction waves. The four possible wave patterns are as follows:

(a) Left rarefaction, right shock,

(b) Left shock, right rarefaction,

(c) Both left and right rarefaction,

(d) Both left and right shock.

In the example of the dam-break problem, with inital conditions as in Figure 3.4, the solution consists of a left rarefaction wave and a right shock wave. This means that the shock wave moves to the right and is characterized by

a discontinuity in the solution and a high speed. Whereas, the rarefaction wave moves to the left and is characterized by a more smooth transition in the solution and a lower speed. The Riemann solver computes the flux across the interface between two cells, considering the left and right states. The numerical fluxes calculated by the Riemann solver allow for the determination of the state in the star region, which represents the solution at the interface between the two regions. This is essential for determining whether the wave is a shock or a rarefaction wave, ensuring both the stability and accuracy of the numerical scheme. In summary, Riemann solvers and numerical fluxes are crucial for solving for the states in the star region, which are then used to update the solution at each timestep.

3.5 Numerical fluxes

In this section we will study the numerical fluxes used to solve the SWE in 1D. At each cell interface, we need to solve the Riemann problem (3.4.1) to find the numerical flux. There are several numerical fluxes that can be used to solve the local Riemann problem, and we will consider some of them in this section. The fluxes we will consider are the Godunov method with an exact Riemann solver, the HLL, HLLC, Rusanov, Lax-Friedrichs, Lax-Wendroff and FORCE fluxes. All of them are later implemented and tested in the numerical experiments in chapter 6.

Godunov method with exact Riemann solver

We consider the Godunov Upwind method, which is a first-order accurate metod to solve non-linear systems of hyperbolic conservation laws [6]. In the method we solve the non-linear Riemann problem at each cell interface. The Godunov flux is given by

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}(\mathbf{U}_{i+\frac{1}{2}}),$$

meaning that we solve the Riemann problem exactly to find h^* and u^*, and then use these values to compute the flux as

$$\mathbf{F}_{i+\frac{1}{2}} = \begin{bmatrix} h^*u^* \\ h^*(u^*)^2 + \frac{1}{2}g(h^*)^2 \end{bmatrix}.$$

HLL

The HLL (Harten, Lax and van Leer) approach assumes a two-wave structure of the Riemann problem. The solver is based on the data $\mathbf{U}_L := \mathbf{U}_i^n, \mathbf{U}_R := \mathbf{U}_{i+1}^n$ and fluxes $\mathbf{F}_L := \mathbf{F}(\mathbf{U}_L), \mathbf{F}_R := \mathbf{F}(\mathbf{U}_R)$. The HLL flux is given by

$$\mathbf{F}_{i+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } S_L \geq 0, \\ \mathbf{F}^{HLL} \equiv \frac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_LS_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & \text{if } S_L \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R \leq 0. \end{cases} \tag{3.5.1}$$

The wave speeds S_L and S_R must be estimated in some way, and one possibility is to use

$$S_L = u_L - a_Lq_L, \quad S_R = u_R + a_Rq_R,$$

where $a_L = \sqrt{gh_L}, a_R = \sqrt{gh_R}$ and $q_K (K = L, R)$ is given by

$$q_K = \begin{cases} \sqrt{\frac{1}{2}\left(\frac{(\hat{h}+h_K)\hat{h}}{h_K^2}\right)} & \text{if } \hat{h} > h_K, \\ 1 & \text{if } \hat{h} \leq h_K. \end{cases}$$

Chapter 4

Neural Networks and Fourier Neural Operators

So far, we have studied numerical methods for solving partial differential equations (PDEs). The finite volume method (FVM), along with other numerical solvers such as the finite difference method (FDM) and finite element method (FEM), solves PDEs by discretizing the domain into a grid. A finer grid improves the accuracy of the solution but also increases the computational cost, creating a trade-off between accuracy and efficiency. Complex PDEs often require a fine grid to accurately capture the solution, which can be computationally expensive.

In this chapter, we introduce the use of data-driven methods for solving PDEs. We will introduce the concepts of convolutional neural networks (CNNs) and Fourier neural operators (FNOs).

4.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized class of artificial neural networks designed to process and analyze data with a grid-like topology, such as images or time-series data represented by 2D grids. CNNs excel at extracting spatial featues from data through the use of convolutional layers, which apply learnable filters to detect patterns such as edges, shapes or textures. These layers are typically followed by pooling layers for dimensionality reduction and fully connected layers for classification or regression tasks. A key advantage of CNNs is their ability to reduce the number of parameters compared to fully connected networks by sharing weights across spatial regions, making them computationally efficient and less prone to overfitting when working with large inputs. Although CNNs are traditionally used for image recognition tasks, their architecture is adaptable for time-series analysis, especially when the data is structured spatially or sequentially.

In this project, CNNs are used to solve the shallow water equations, by training on the solution data generated by the FVM solver. We aim to learn the underlying dynamics of the system by training the CNN on sequences of input-output pairs, where the input is the state of the system at time t and the output is the state at time $t + \Delta t$. The network is trained to construct a flow map. The flow map $\Phi^t : \mathbb{R}^n \to \mathbb{R}^n$ is defined such that, for all $x \in X$ and all $t \in \mathbb{R}$:

$$\Phi^t(x_0) = x(t),$$

where $x(t)$ is the solution of the PDEs with initial condition $x(0) = x_0$. The flow map satisfies

$$\Phi(\Phi(x, t), s) = \Phi(x, s + t),$$

A CNN can approximate the flow map Φ by training on data that pairs an initial state x_0 with its state at later times $x(t)$. The goal is to learn the mapping:

$$x(t) = \Phi_{CNN}(x_0),$$

where Φ_{CNN} is the CNNS' approximation of the flow map. The CNN model is trained on input-output pairs of sequences of data, with a sequence length of N.

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Meaning the network takes sequences of input data, and predicts the corresponding output data. The output of the CNN is the solution at the next time step, effectively learning the dynamics of the SWE through the time-series data. This setup allows the model to capture both spatial and temporal dependencies in the data, leveraging the CNN's ability to learn localized featires while processing sequential information. A pro of CNN's is that they are efficient. By processing data in parallel using convolutional layers, the CNN efficiently handles large datasets without requiring excessive computational resources. Another pro is the adaptability. The model's ability to learn from sequential data makes it adaptable for time-series predictions in dynamic systems like the shallow water equations. A con is the data representation. Representing time-series data as sequences may require preprocessing, which can introduce complexity or a potential loss of information. There can also be temporal limitations, as CNNs lack explicit mechanisms to model long-term temporal dependencies (unless extended with additional architechtures like RNNs or Transformers). We will test and evaluate the performance of the CNN model in chapter 7.

4.2 Fourier Neural Operators

In this section, we introduce the concept of Fourier Neural Operators (FNOs), a method for approximating mappings between infinite-dimensional function spaces. The theory and method described here are based on the paper [15]. The goal is to learn a mapping between two infinite dimensional spaces from a finite collection of input-output pairs. Consider the operator $G : A \rightarrow U$, which maps from an infinite-dimensional function space A to another infinite-dimensional function space U. We aim to approximate the exact operator G by constructing the map

$$G_\theta : A \mapsto U, \quad \theta \in \Theta,$$

where Θ is a finite-dimensional parameter space. Consider the functions $a \in A$ and $u \in U$. We assume access to data in the form of pointwise evaluations of these functions, i.e., we have access to the observations $\{a_j, u_j\}_{j=0}^N$, in a domain $D \subset \mathbb{R}^d$, which is bounded open set. The first step is to create $v_0(x) = P(a(x))$ by the input layer P, which is typically a shallow fully-connected neural network. The neural operator is iterative, meaning we apply several iterations of updates to obtain $v_1, v_2, \ldots$ up to v_T. An update $v_t \mapsto v_{t+1}$ is defined as

$$v_{t+1}(x) := \sigma\left(W v_t(x) + \left(\mathcal{K}(a;\phi)v_t\right)(x)\right), \quad \forall x \in D, \tag{4.2.1}$$

where $W : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a non-linear activation function. The output $u(x) = Q(v_T(x))$ is the final update v_T transformed by the output layer Q. There are various types of operators, but the core of the Fourier neural operator lies in its kernel function $\mathcal{K}$. We define the Fourier integral operator $\mathcal{K}$ as

$$\left(\mathcal{K}(\phi)v_t\right)(x) := \mathcal{F}^{-1}\left(R_\phi \cdot (\mathcal{F}v_t)\right)(x), \quad \forall x \in D, \tag{4.2.2}$$

where $\mathcal{F}$ is the Fourier transform, $\mathcal{F}^{-1}$ is the inverse Fourier transform, and R is the linear transformation applied on the lower Fourier modes. By transforming the data into the Fourier domain, FNOs can take advantage of the

periodicity and smoothness properties of the Fourier basis, which simplifies the learning process for functions defined over continuous domains. This approach leverages the fact that differentiation with respect to time is equivalent to multiplication in the Fourier domain, distinct from the convolution theorem. CITE. The network architechture for the FNO model is illustrated in Figure 4.1.

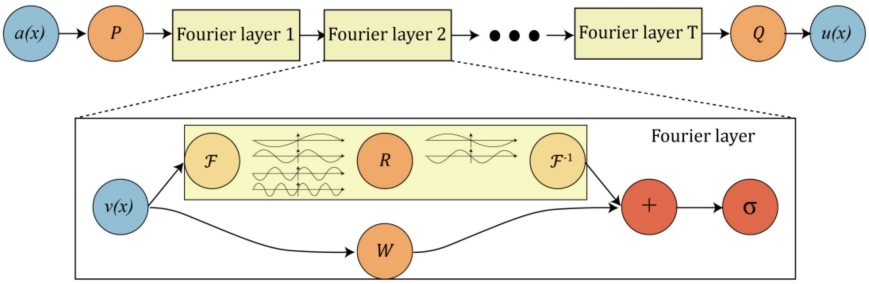


Figure 4.1: An overview of the network architecture with several Fourier layers. Illustration from [15]. Top: Overall structure with input function a, input layer P, several Fourier layers, output layer Q and output function u. Bottom: A Fourier layer consists of two parallel paths. Top is a Fourier transformation $\mathcal{F}$, a linear transformation R and an inverse Fourier transformation $\mathcal{F}^{-1}$. The bottom path consists of a linear transformation W. The parts meet and undertake an activation function σ.

From the top in Figure 4.1 we see that the network consists of an input function $a(x)$, an input layer P, several Fourier layers, an output layer Q and some output function $u(x)$. As we only have access to pointwise evaluations of $a(x)$ and $u(x)$ these are what we will use. That is, our goal is to approximate the input-output mapping:

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix} \rightarrow \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}.$$

In the context of this project we have that $a(x) = v_t(x)$ and $u(x) = v_{t+1}(x)$.

In the bottom of Figure 4.1 we see the structure of a Fourier layer, which consists of two parallel paths. In the top path, the data undergoes a Fourier transform $\mathcal{F}$, decomposing it into a sum of Fourier basis functions (sines and cosines) with varying frequencies, amplitudes and phases. A linear transformation R it the applied to filter out the higher Fourier modes, as illustrated in the figure, where high-frequency components are removed. When implementing the model, we choose the number of Fourier modes to retain, depending on how much information we want to preserve. Retaining more modes keeps more information, but may also introduce more noise and oscillations. After filtering, the inverse Fourier transform $\mathcal{F}^{-1}$ is applied to reconstruct the data in its original form. The bottom path involves a linear transformation W, and the two paths merge before applying a non-linear activation function σ.

A key advantage of FNOs is that since they learn mapping between function spaces, they are not constrained by a specific grid or mesh. This allows them to transfor solutions across different grids, a process known as zero-show super-resolution. The learned operator can generalize from the coarse grid to the fine grid without needing to be retrained, making it grid-independent. This is a major advantage because it reduces the computational cost associated with fine-grid simulations. In chapter 7, we will evaluate the performance of the implemented FNO model in this context, testing its ability to maintain high accuracy when making predictions on finer grids.

Multistep prediction

We aim to develop a multi-step prediction model that can predict a specified number of time steps ahead. The input-output pairs are given as $\{a_j, u_j\}_{j=0}^N$, where the points $\{u\}_{j=0}^N$ represent the values one time step after $\{a\}_{j=0}^N$. We collect state pairs $\{v_t, v_{t+1}\}_{j=0}^N$ for training a flowmap. Feeding this data into the model allows it to learn the flow map for the system. We are dealing with time series data, and the original form of the FNO can predict one time step ahead. However, for many applications, predicting multiple time steps ahead is more useful. To enable multi-step predictions, we organize the input data into sequences. The model is trained to predict the output state based on data from several previous time steps, with a specified sequence length. That is, the input data is given as

$$\{v_{t-n}, v_{t-n+1}, \ldots, v_{t+1}\}_{j=0}^N,$$

where n is the sequence length.

We will conduct experiments to determine the optimal sequence length for our model.

During prediction, the model's output is added to the input as the newest data point, replacing the oldest data. This process is repeated until predictions for the desired number of future time steps are made. The process is illustrated in Figure 4.2.

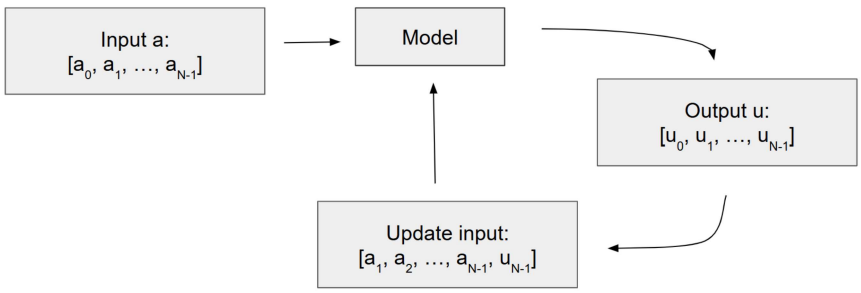


Figure 4.2: Flowchart of the multi-step prediction process.

The results of the multi-step prediction model will be presented in chapter 7.