Fourier Neural Operator-Based Models for Solving the Shallow Water Equations in Spherical Coordinates

Melissa Ulsøe Jessen (s194322)

Supervisor: DTU Compute
Co-supervisor: DTU Compute

Master Thesis
Mathematical Modelling and Computation

DTU Compute
Technical University of Denmark
February 2, 2025

Preface

This thesis has been prepared over five months at the Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, DTU, in partial fulfillment of the requirements for the degree of Master of Science in Mathematical Modelling and Computation. The project was carried out from September 2024 to February 2025 and is equivalent to 30 ECTS points.

Melissa Ulsøe Jessen (s194322)

Abstract

This master thesis focuses on the numerical and data-driven solutions for the Shallow Water Equations (SWE), which are fundamental in computational fluid dynamics. The project aims to solve the SWE using the Finite Volume Method (FVM) in 2D and spherical coordinates and subsequently exploit machine learning techniques, particularly the Fourier Neural Operator (FNO), to improve solution accuracy and efficiency. By the end of the project, the goal is to have a robust and efficient computational toolkit for solving the SWE, which will also be useful for simulating important geophysical processes like Kelvin and Rossby waves.

Acknowledgements

Contents

Chapter 1

Introduction

This master thesis focuses on the numerical and data-driven solutions for the Shallow Water Equations (SWE), which are fundamental in computational fluid dynamics. The project aims to solve the SWE using the Finite Volume Method (FVM) in 2D and spherical coordinates and subsequently exploit machine learning techniques, particularly the Fourier Neural Operator (FNO), to improve solution accuracy and efficiency. By the end of the project, the goal is to have a robust and efficient computational toolkit for solving the SWE, which will also be useful for simulating important geophysical processes like Kelvin and Rossby waves.

The Shallow Water Equations (SWE) are a set of hyperbolic partial differential equations that describe the motion of a fluid in a shallow layer of water. In this project, we will derive the SWE in 1D, 2D and in spherical coordinates, to model the flow of water on the surface of the Earth. We will go through the Finite Volume Method (FVM) in 1D and 2D, which is a numerical method for solving partial differential equations by dividing the domain into small control volumes and integrating the equations over these volumes. The method is widely used in computational fluid dynamics to model the behavior of fluid flows. We will implement the FVM and use it to solve the SWE in 1D for two different dam break problems.

1.1 Motivation

1.2 Literature

When working in this area it inevitably 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* [1], *Riemann Solvers and Numerical Methods for Fluid Dynamics* [2] and the new book from 2024 *Computational Algorithms for Shallow Water Equations* [3] as references. The books have been especially useful when deriving the shallow water equations and the Riemann solvers used in this project, as well as understanding and describing the Finite Volume Method.

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 [4].

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. [5] and *Physics-informed neural networks for the shallow-water equations on the sphere* by Bihlo et al. [6] are important references. The lecture notes *Shallow water on a sphere* by Raymond from New Mexico Tech [7], have also been useful in the derivation of the shallow water

equations in spherical coordinates. For the spherical SWE the papers [8] and [9] are also important references.

In the field of Fourier Neural Operators, there is not a lot of literature on the topic, as the concept of Fourier Neural Operators is relatively new. However, the paper *Fourier Neural Operator for Parametric Partial Differential Equations* [10], written by several authors, is a key reference. 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 [11]. Another paper is [12].

1.3 Thesis overview

The thesis is structured as follows. The shallow water equations are derived in 1D, 2D and in spherical coordinates in chapter 2. In chapter 3 we present the Finite Volume Method (FVM), which is used to solve the shallow water equations. We also present the Riemann problem and the numerical fluxes used in the FVM. These chapter also constitue the theory and methodology for the numerical methods used in this project, and we move on t the data-driven methods. In chapter 4 we introduce the concept of Fourier Neural Operators (FNO) and Convolutional Neural Networks (CNN), and how these can be used to solve the shallow water equations. In chapter 5 we outline how the data needed for the data-driven methods is generateded, as it is a part of this project, as we are generating all the data used for the data-driven methods ourselves.

Chapter 6 is dedicated to the numerical results for the 1D and 2D shallow water equations using the FVM. To validate the numerical results, we compare the FVM results to test cases from the literature. It is important to validate the numerical results, as the data-driven methods will be trained on the FVM data. In chapter 7 we present the results for the SWE using the data-driven models, and compare the results in terms of run time and accuracy. We analyse the results and discuss the performance of the data-driven models, and also compare them to the numerical results.

Finally, in chapter 8 we discuss the results, and in chapter 9 we conclude the thesis and suggest further work.

Chapter 2

The shallow water equations

In this chapter we will derive the shallow water equations (SWE) in conservative form, and present the equations in vector form and in integral form for 1D and 2D in cartesian coordinates. We will also derive the SWE in spherical coordinates, which is relevant for geophysical applications, such as simulating water flow on the surface of the Earth. To solve the SWE, we will present the numerical method, called the Finite Volume Method (FVM) including the Riemann problem. Lastly, we will dive into the theory behind the data-driven methods, i.e., the neural networks and fourier neural operators.

2.1 Notation

Before deriving the shallow water equations (SWE), we will introduce the notation that will be used throughout this report. In both the 1D case and the 2D case of the SWE, we use cartesian coordinates (x, y, z) with time denoted by t. Given that linear algebra is a fundamental tool used in this report, we first establish the relevant notation. Lowercase bold letters represent vectors, while uppercase bold letters represent matrices. For instance, $\mathbf{a}$ is a vector of size $r \times 1, r \in \mathbb{R}$, and $\mathbf{A}$ is a matrix of size $m \times n, m, n \in \mathbb{R}$. The identity matrix, denoted by $\mathbf{I}$, is a square matrix with ones along the diagonal and zeros elsewhere. For example, the 3×3 identity matrix is given by:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In this project, differential calculus plays a significant role. We denote partial derivatives using the following notation:

$$f_x = \frac{\partial f}{\partial x}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_z = \frac{\partial f}{\partial z}. \tag{2.1.1}$$

The gradient operator, denoted by ∇, gives the gradient of a scalar function $f(x, y, z)$ as a vector:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \end{bmatrix}.$$

Given two vectors $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^\top$ and $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}^\top$, the dot product of $\mathbf{a}$ and $\mathbf{b}$ is given by:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3.$$

The dot product can also be written as a matrix product:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^\top \mathbf{b}.$$

The divergence operator, represented as $\nabla\cdot$, gives the divergence of a vector $\mathbf{a}$ as:

$$\nabla \cdot \mathbf{a} = \frac{\partial a_1}{\partial x} + \frac{\partial a_2}{\partial y} + \frac{\partial a_3}{\partial z} = a_{1x} + a_{2y} + a_{3z},$$

using the notation for partial derivatives introduced in (2.1.1). The tensor product of two vectors $\mathbf{a}$ and $\mathbf{b}$, denoted as $\mathbf{a} \otimes \mathbf{b}$, is a matrix where each element is the product of the elements of $\mathbf{a}$ and $\mathbf{b}$, i.e.,

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix}.$$

Establishing this relevant notation, we can now derive the shallow water equations.

In this section we will derive the shallow water equations (SWE) in conservative form.

2.2 Derivation of the SWE with conservative variables

In this section we will derive the shallow water equations (SWE) in conservative form. The derivation follows four steps: First we consider the conservation laws for mass and momentum, and then we consider the boundary conditions for a free surface problem. Afterwards we make some neccessary assumptions and finally we use the boundary conditions to integrate the conservation laws over depth. The derivation follows the methods outlined in [1] and [13].

Conservation laws

The SWE are derived from the conservation laws for mass and momentum, which are fundamental in fluid dynamics. The conservation laws for mass and momentum can be expressed generally as follows (source: eq. (2.1) and (2.2) in [1]):

$$\rho_t + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{2.2.1}$$

$$(\rho \mathbf{v})_t + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p\mathbf{I} - \mathbf{T}) = \rho \mathbf{g}, \tag{2.2.2}$$

where ρ is the fluid density, $\mathbf{v} = \begin{bmatrix} u & v & w \end{bmatrix}^\top$ is the fluid velocity in the x, y and $z-$direction respectively; p is the pressure, $\mathbf{I}$ is the identity matrix, and the vector $\mathbf{g} = \begin{bmatrix} g_1 & g_2 & g_3 \end{bmatrix}^\top$ represents body forces including gravity. In these equations, the density ρ and the pressure p are dependent of x, y, z and t, but later we will introduce some assumptions that simplify the equations. The matrix $\mathbf{T}$ is the viscous stress tensor, given by

$$\mathbf{T} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix},$$

which accounts for the viscous forces in the fluid. However, in this project the viscous stress tensor $\mathbf{T}$ is neglected, since we assume the function $\tau(x, y, z)$ is constant. The matrix $\mathbf{v} \otimes \mathbf{v}$ represents the tensor product of the velocity vector $\mathbf{v}$ with itself, i.e.,

$$\mathbf{v} \otimes \mathbf{v} = \begin{bmatrix} u^2 & uv & uw \\ vu & v^2 & vw \\ wu & wv & w^2 \end{bmatrix}.$$

Note that $\mathbf{v} \otimes \mathbf{v} = \mathbf{v}\mathbf{v}^\top$. Putting this together, we can rewrite the momentum equation (2.2.2) as

$$(\rho \mathbf{v})_t + \nabla \cdot (\rho \mathbf{v}\mathbf{v}^\top + p\mathbf{I}) - \rho \mathbf{g} = 0. \tag{2.2.3}$$

In this project we consider incompressible fluids, meaning that the fluid density ρ is independent of the pressure p. We also assume that the fluid density only depends on temperature and salinity, and thus is independent of t, x, y and z. Additionally, we assume ρ is nonzero. Rewriting the mass conservation equation (2.2.1) gives

$$\rho_t + \rho(u_x + v_y + w_z) + u\rho_x + v\rho_y + w\rho_z = 0,$$

using the given assumptions and the product rule for differentiation. Hence we obtain

$$u_x + v_y + w_z = 0, \tag{2.2.4}$$

also referred to as the mass conservation equation. Applying the divergence operator $\nabla\cdot$, the momentum conservation equation (2.2.3) can be written out as:

$$\rho_t \mathbf{v} + \rho \mathbf{v}_t + \rho \begin{bmatrix} (u^2+p)_x + (uv)_y + (uw)_z \\ (vu)_x + (v^2+p)_y + (vw)_z \\ (wu)_x + (wv)_y + (w^2+p)_z \end{bmatrix} - \rho \mathbf{g} = 0. \tag{2.2.5}$$

We neglect all body forces in $\mathbf{g}$, expect the gravitational force in the $z-$direction, i.e., $\mathbf{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}$, where g is the gravity acceleration, which we assume to be constant. Hence, by using the product rule in (2.2.5) and that $\rho_t = 0$ we obtain

$$\rho \begin{bmatrix} u_t \\ v_t \\ w_t \end{bmatrix} + \rho \begin{bmatrix} p_x + uu_x + vu_y + wu_z + u(u_x + v_y + w_z) \\ p_y + uv_x + vv_y + wv_z + v(u_x + v_y + w_z) \\ p_z + uw_x + vw_y + ww_z + w(u_x + v_y + w_z) \end{bmatrix} - \rho \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = 0. \tag{2.2.6}$$

We apply (2.2.4) to (2.2.6) to remove some terms, we move the pressure terms to the right hand side, and we divide by ρ. Putting it all together, the mass equation (2.2.1) and the momentum equation (2.2.3), split in x, y and $z-$directions, simplify to

$$\left.\begin{aligned} u_x + v_y + w_z &= 0, \\ u_t + uu_x + vu_y + wu_z &= -\frac{1}{\rho} p_x, \\ v_t + uv_x + vv_y + wv_z &= -\frac{1}{\rho} p_y, \\ w_t + uw_x + vw_y + ww_z &= -\frac{1}{\rho} p_z - g. \end{aligned}\right\} \tag{2.2.7}$$

Boundary conditions

In this project, we consider the flow of water with a free surface, meaning that the surface is not fixed and can move or change over time. To solve the SWE, it is essential to impose boundary conditions at both the bottom of the water column and at the free surface. We assumme the bottom b is defined by a function

$$z = b(x, y),$$

meaning that the bottom is dependent on x and y, but not on the time t. Since the bottom is not moving over time, we refer to it as fixed. The free surface is defined by

$$z = s(x, y, t) \equiv b(x, y) + h(x, y, t),$$

where $h(x, y, t)$ is the water depth at time t. The following illustration helps to visualize the setup:

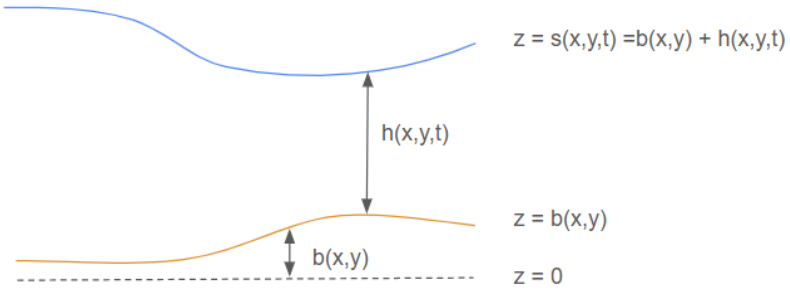


Figure 2.1: Illustration of a water column with a free surface.

We impose boundary conditions at the bottom and at the free surface, adressing both kinematic and dynamical conditions. To describe the boundaries mathematically, we introduce a boundary function $\psi(x, y, z, t)$ that is zero on the boundaries:

$$\psi(x, y, z, t) = 0.$$

For the free surface, this boundary is given by

$$\psi|_{z=s} = z - s(x, y, t) = 0, \tag{2.2.8}$$

and for the bottom, it is described by

$$\psi|_{z=b} = z - b(x, y) = 0. \tag{2.2.9}$$

In the kinematic condition, we assume that fluid particles on the boundary remain on the boundary over time. Mathematically this is expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi(x, y, z, t) = 0.$$

Recall, that $\frac{\partial\psi}{\partial t}$ is the partial derivative of ψ with respect to t, while the total derivate $\frac{\mathrm{d}\psi}{\mathrm{d}t}$ accounts for both the direct change of ψ with respect to t and the changes due to the movement of the fluid in the x, y and z directions. Hence, the total derivative of ψ wrt. t is given by

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{\partial\psi}{\partial t} + \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial\psi}{\partial x} + \frac{\mathrm{d}y}{\mathrm{d}t}\frac{\partial\psi}{\partial y} + \frac{\mathrm{d}z}{\mathrm{d}t}\frac{\partial\psi}{\partial z}.$$

We see that $\frac{\mathrm{d}x}{\mathrm{d}t}$ denotes the velocity in the $x-$direction, i.e., u, and correspondingly $\frac{\mathrm{d}y}{\mathrm{d}t}$ and $\frac{\mathrm{d}x}{\mathrm{d}t}$ denotes v and w respectively. Thus, the kinematic condition is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi = \psi_t + u\psi_x + v\psi_y + w\psi_z = 0. \tag{2.2.10}$$

Applying this to the free surface by substituting (2.2.8) into the kinematic condition (2.2.10) yields

$$(s_t + us_x + vs_y - w)|_{z=s} = 0. \tag{2.2.11}$$

Similarly, for the bottom, substitung (2.2.9) into the kinematic condition (2.2.10) gives

$$(ub_x + vb_y - w)|_{z=b} = 0. \tag{2.2.12}$$

The dynamical condition is related to the pressure distribution at the free surface. We assume that the pressure at the free surface is equal to the pressure in the air above the surface, that is, the atmospheric pressure. Since absolute pressure levels are irrelevant, as we are primarily concerned with pressure differences, we set the pressure at the free surface to zero. This leads to the following expression for the pressure at the free surface:

$$p(x, y, z, t)|_{z=s} = 0. \tag{2.2.13}$$

This condition, known as the dynamical condition, relates to the forces acting on the boundaries of the fluid.

Assumptions

To derive the SWE it is neccessary to make some assumptions. The shallow water equations are an approximation to the full free-surface problem and result from the assumption that the vertical compononent of the acceleration is negligible. Therefore, we begin by assuming that the vertical acceleration, represented by the total derivate of the vertical velocity component w with respect to time, is negligible. This assumption leads to the condition

$$\frac{\mathrm{d}w}{\mathrm{d}t} = w_t + uw_x + vw_y + ww_z = 0.$$

Applying $\frac{\mathrm{d}w}{\mathrm{d}t} = 0$ in the $z-$momentum conservation equation (2.2.7) simplifies it to

$$p_z = -\rho g.$$

By using properties of an integral, together with (2.2.13) we get

$$\int_{b(x,y)}^{z} -\rho g \ \mathrm{d}t + \int_{z}^{s(x,y,t)} -\rho g \ \mathrm{d}t = \int_{b(x,y)}^{s(x,y,t)} -\rho g \ \mathrm{d}t = p|_{z=s(x,y,t)} = 0.$$

This implies that the pressure distribution follows

$$p = \int_{b(x,y)}^{z} -\rho g \ \mathrm{d}t = \int_{z}^{s(x,y,t)} \rho g \ \mathrm{d}t = \rho g(s-z), \tag{2.2.14}$$

where s is the surface height. Differentiating (2.2.14) with respect to x and y yields

$$p_x = \rho g s_x, \quad p_y = \rho g s_y.$$

Substituting these expressions into the $x-$ and $y-$momentum conservation equations (2.2.7) leads to

$$\left.\begin{aligned} u_t + uu_x + vu_y + wu_z &= -gs_x, \\ v_t + uv_x + vv_y + wv_z &= -gs_y, \end{aligned}\right\} \tag{2.2.15}$$

which can be further simplified. We realize that both p_x and p_y are independent of z, implying that $\frac{\mathrm{d}u}{\mathrm{d}t}$ and $\frac{\mathrm{d}v}{\mathrm{d}t}$ are also independent of z. Hence $u_z = v_z = 0$, implying that (2.2.15) can be simplified to

$$\left.\begin{aligned} u_t + uu_x + vu_y &= -gs_x, \\ v_t + uv_x + vv_y &= -gs_y. \end{aligned}\right\} \tag{2.2.16}$$

These are the momentum equations for the shallow water equations in two dimensions.

Integration over depth

The next step in deriving the SWE is to integrate the conservation equations over the vertical direction z. We integrate the mass conservation equation (2.2.4) and the momentum conservation equations in (2.2.16), from the bottom, $z = b(x, y)$ to the free surface, $z = s(x, y, t)$. Starting with the mass conservation equation (2.2.4), we have

$$\int_b^s u_x + v_y + w_z \ \mathrm{d}z = 0,$$

implying that, using linearity of the integral:

$$\int_b^s u_x \ \mathrm{d}z + \int_b^s v_y \ \mathrm{d}z + w|_{z=s} - w|_{z=b} = 0. \tag{2.2.17}$$

We will use Leibniz's integral rule [14], which is stated as follows:

$$\frac{\mathrm{d}}{\mathrm{d}x}\int_{a(x)}^{b(x)} f(x,t)\ \mathrm{d}t = \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)\ \mathrm{d}t + f(x,b(x))\frac{\mathrm{d}}{\mathrm{d}x}b(x) - f(x,a(x))\frac{\mathrm{d}}{\mathrm{d}x}a(x), \tag{2.2.18}$$

to integrate the first two terms in (2.2.17), which yields

$$\left.\begin{aligned} \int_b^s u_x\,\mathrm{d}z &= \frac{\mathrm{d}}{\mathrm{d}x}\int_b^s u\,\mathrm{d}z - u|_{z=s}\frac{\mathrm{d}s}{\mathrm{d}x} + u|_{z=b}\frac{\mathrm{d}b}{\mathrm{d}x},\\ \int_b^s v_y\,\mathrm{d}z &= \frac{\mathrm{d}}{\mathrm{d}y}\int_b^s v\,\mathrm{d}z - v|_{z=s}\frac{\mathrm{d}s}{\mathrm{d}y} + v|_{z=b}\frac{\mathrm{d}b}{\mathrm{d}y}. \end{aligned}\right\} \tag{2.2.19}$$

Note that since a change in x does not affect the y-component of the bottom or surface, we have that $\frac{\mathrm{d}s}{\mathrm{d}x} = s_x$ and $\frac{\mathrm{d}b}{\mathrm{d}x} = b_x$, and correspondingly for s_y and b_y. Likewise we can substitute $\frac{\mathrm{d}}{\mathrm{dx}}$ with $\frac{\partial}{\partial x}$ in the integrals, since the integrals are with respect to z, and u and v are independent of z. Inserting these results in (2.2.19) gives

$$\left.\begin{aligned} \int_b^s u_x\ \mathrm{d}z &= \frac{\partial}{\partial x}\int_b^s u\ \mathrm{d}z - u|_{z=s}s_x + u|_{z=b}b_x,\\ \int_b^s v_y\ \mathrm{d}z &= \frac{\partial}{\partial y}\int_b^s v\ \mathrm{d}z - v|_{z=s}s_y + v|_{z=b}b_y. \end{aligned}\right\} \tag{2.2.20}$$

We can now insert the integrals (2.2.20) into the integrated mass conservation equation (2.2.17) to get

$$\frac{\partial}{\partial x}\int_b^s u\ \mathrm{d}z - u|_{z=s}s_x + u|_{z=b}b_x + \frac{\partial}{\partial y}\int_b^s v\ \mathrm{d}z - v|_{z=s}s_y + v|_{z=b}b_y + w|_{z=s} - w|_{z=b} = 0. \tag{2.2.21}$$

To simplify this equation further, we consider the boundary conditions. From (2.2.12) we have

$$w|_{z=b} = (ub_x + vb_y)|_{z=b}, \tag{2.2.22}$$

and from (2.2.11) we have

$$w|_{z=s} = (s_t + us_x + vs_y)|_{z=s}. \tag{2.2.23}$$

We note that $s = b + h$ and hence $s_t = h_t$, as the bottom is fixed. Recall that u and v are independent of z, and the water depth is $h = s - b$, meaning we have

$$\int_b^s u\ \mathrm{d}z = u(s-b) = hu, \quad \int_b^s v\ \mathrm{d}z = v(s-b) = hv.$$

Putting it all together the equation (2.2.21) simplifies to

$$h_t + (hu)_x + (hv)_y = 0, \tag{2.2.24}$$

which is also the first equation in the SWE in conservative form. When integrating the momentum equations (2.2.16) over the vertical direction, we see that since the equations are independent of z, the resulting equations are simply

$$\left.\begin{aligned} h(u_t + uu_x + vu_y + gs_x) &= 0,\\ h(v_t + uv_x + vv_y + gs_y) &= 0. \end{aligned}\right\} \tag{2.2.25}$$

We multiply (2.2.24) with u and v respectively, and add the resulting two equations to (2.2.25). Recall that $s = h + b$. By using the product rule for differentiation and collecting terms, we obtain the momentum equations in conservative form:

$$\left.\begin{aligned} (hu)_t + (hu^2 + \frac{1}{2}gh^2)_x + (huv)_y &= -ghb_x,\\ (hv)_t + (huv)_x + (hv^2 + \frac{1}{2}gh^2)_y &= -ghb_y. \end{aligned}\right\} \tag{2.2.26}$$

The three partial differential equations in (2.2.24) and (2.2.26) are the shallow water equations in conservative form.

2.3 The SWE in vector form for 1D and 2D

The SWE can also be written in differential conservation law form as the vector equation

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = \mathbf{S}(\mathbf{U}), \tag{2.3.1}$$

where

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \quad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix} \quad \text{and} \quad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -ghb_x \\ -ghb_y \end{bmatrix}.$$

We call $\mathbf{U}$ the vector of conserved variables, $\mathbf{F}(\mathbf{U})$ and $\mathbf{G}(\mathbf{U})$ the flux vectors in the x and y direction, and $\mathbf{S}(\mathbf{U})$ the source term vector.

Homogeneous 1D case:
The most simple case is the homogeneous one-dimensional case, where the source term vector $\mathbf{S}(\mathbf{U}) = 0$. The vector form of the homogeneous SWE in 1D is then given by

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0,$$

where

$$\mathbf{U} = \begin{bmatrix} h \\ hu \end{bmatrix}, \quad \text{and} \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}.$$

As it is the 1D case, we only consider flow in the x-direction.

Inhomogeneous 1D case:
The inhomoegeneous one-dimensional case of the shallow water equations in vector form is given by

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}), \tag{2.3.2}$$

where

$$\mathbf{U} = \begin{bmatrix} h \\ hu \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} \quad \text{and} \quad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ -ghb_x \end{bmatrix}. \tag{2.3.3}$$

If $\mathbf{S}(\mathbf{U}) = 0$, the equation (2.3.2) is called a conservation law, and otherwise it is called a balance law.

2.4 The 1D SWE in integral form

It is often more convenient to work with the integral form of the SWE, since the integral form of equations of the form (2.3.2) and (2.3.3) allows discountinuous solutions. We derive the integral form of the inhomoegeneous one-dimensional case of the SWE in vector form (2.3.2). The integral form is obtained by integrating the vector form (2.3.2) over a control volume V in the x, t plane

$$V = [x_L, x_R] \times [t_1, t_2].$$

The control volume is illustrated in Figure 2.2.

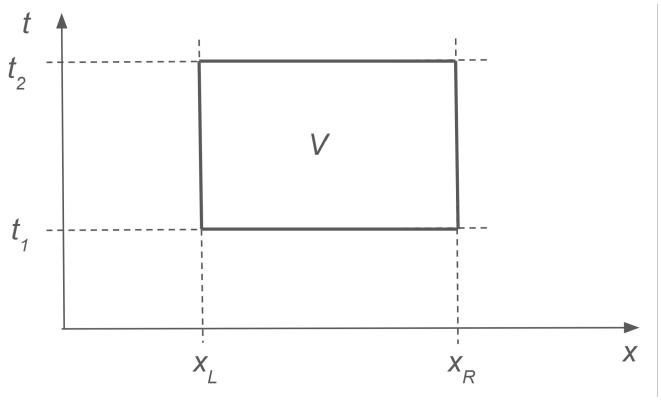


Figure 2.2: Illustration of a control volume V in the x, t plane. Illustration modified from [3].

First we integrate the vector form of the SWE (2.3.2) over x from x_L to x_R to obtain

$$\int_{x_L}^{x_R} \mathbf{U}_t \, \mathrm{d}x + \int_{x_L}^{x_R} \mathbf{F}(\mathbf{U})_x \, \mathrm{d}x = \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.4.1}$$

Using the fundamental theorem of calculus, we get that

$$\int_{x_L}^{x_R} \mathbf{F}(\mathbf{U}) \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_R, t)) - \mathbf{F}(\mathbf{U}(x_L, t)),$$

which we insert in (2.4.1):

$$\int_{x_L}^{x_R} \mathbf{U}_t \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)) + \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.4.2}$$

Then we integrate (2.4.2) over time from t_1 to t_2 to get

$$\int_{t_1}^{t_2} \int_{x_L}^{x_R} \mathbf{U}_t \, \mathrm{d}x\mathrm{d}t = \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_L, t)) \, \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_R, t)) \, \mathrm{d}t + \int_{t_1}^{t_2} \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x\mathrm{d}t.$$

Rewriting the left hand side using the fundamental theorem of calculus, we get

$$\int_{x_L}^{x_R} \mathbf{U}(x, t_2) \, \mathrm{d}x = \int_{x_L}^{x_R} \mathbf{U}(x, t_1) \, \mathrm{d}x + \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_L, t)) \, \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_R, t)) \, \mathrm{d}t + \int_{t_1}^{t_2} \int_{x_1}^{x_2} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x\mathrm{d}t, \tag{2.4.3}$$

which is the integral form of the conservation laws for the SWE in 1D. An alternative integral form of (2.3.2) is stated in [3]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{x_L}^{x_R} \mathbf{U}(x, t) \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)) + \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.4.4}$$

From (2.4.4) we get that the integral form of the homogeneous SWE in 1D is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{x_L}^{x_R} \mathbf{U}(x, t) \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)), \tag{2.4.5}$$

meaning that the rate of change of the integral over a domain is equal to the flux through the boundaries of the domain.

2.5 SWE in Spherical Coordinates

Until now we have derived the shallow water equations in cartesian coordinates. In this section, we will derive the shallow water equations in spherical coordinates. We will follow the methods used in [5] [6], [7] and [15]. To illustrate the spherical coordinates, we will use the latitude and longitude system, visualized in Figure 2.3.

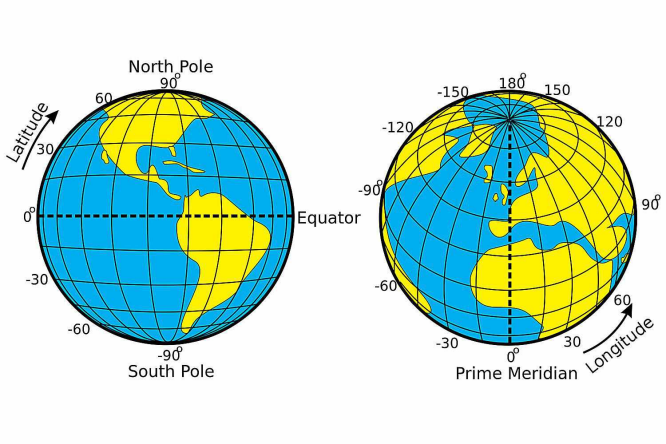


Figure 2.3: Illustration of latitude and longitude on the planet earth. Illustration from [16].

We see that the latitude direction is the north-south component, whereas the longitude direction is the east-west component. The latitude angle, denoted by ϕ, goes from $-\frac{\pi}{2}$ at the south pole to $\frac{\pi}{2}$ at the north pole, and the longitude angle, denoted by θ, goes from 0 at the prime meridian, increasing to the east, to 2π. The spherical coordinates we use are (r, θ, ϕ), where r is the radius from the center of the sphere, θ is the longitude angle, and ϕ is the latitude angle. This also means, that any point on the surface of the sphere can be represented by the coordinates (θ, ϕ). We consider a small domain of the sphere, as illustrated in Figure 2.4.

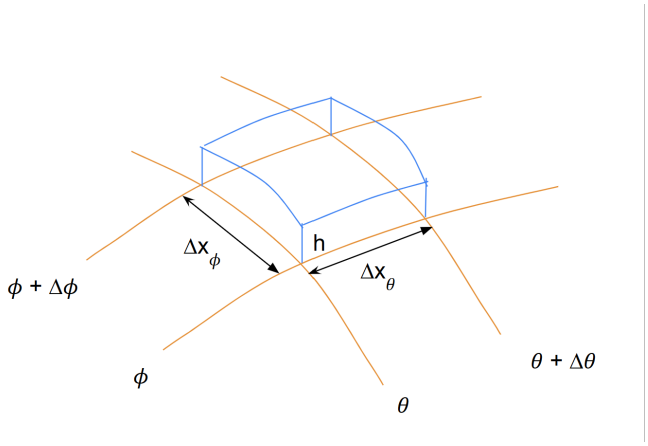


Figure 2.4: Illustrations of a small domain of the surface of the sphere.

We want to find expressions for Δx_ϕ and Δx_θ, the distances in the ϕ and θ directions, respectively, as illustrated in Figure 2.4. We can find these distances by using the arc length formula. Recall that the circumreference of a full circle is $2\pi r$, where r is the radius of the circle. The arc length is a fraction of the full circumreference, and it is given by the formula $l = rv$, where l is the arc length, r is the radius, and v is the angle in radians. Assuming

Earth's latitude side is a circle, we can find the distance Δx_ϕ by using the arc length formula, as:

$$\Delta x_\phi = r\Delta\phi,$$

where $\Delta\phi$ is the change in the latitude angle and r is the radius of the sphere. We assume that Earth is a perfect sphere, meaning that the radius is constant. Considering the longitude dimension Δx_θ, we need to make some adjustments, as we can see that the circumreference at equator is larger than at the poles. That is, we need to consider the radius of the circle at the given latitude ϕ. To illustrate this, we consider Figure 2.5.

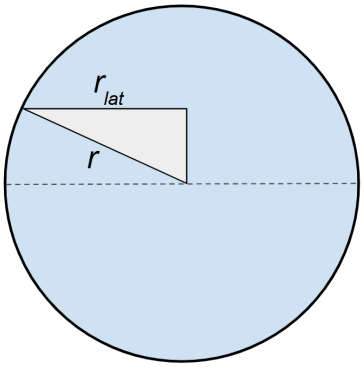


Figure 2.5: Illustration of the radius of the circle at the given latitude ϕ.

In Figure 2.5, we consider a right triangle with the hypotenuse as the radius of the sphere, and the adjacent side as the radius of the circle at the given latitude ϕ. Thus, we can express the radius of the circle at the given latitude ϕ as

$$r_{lat} = r\cos(\phi).$$

Using that, together with the formula for the arc length, we can find the distance Δx_θ as:

$$\Delta x_\theta = r\cos(\phi)\Delta\theta.$$

The volume of a small domain of the sphere, as shown in Figure 2.4, is given by

$$\begin{aligned} V &= \Delta x_\phi \Delta x_\theta h \\ &= r^2 h\cos(\phi)\Delta\phi\Delta\theta, \end{aligned}$$

assuming that the height of the domain is h, and that the domain is rectangular. This is a fair assumption for small values of Δx_ϕ and Δx_θ. We also assume that ϕ is not too close to the poles, as $\cos(\phi)$ will go to zero at the poles. We are interested in rate of change of the volume with respect to time, and we can find this by taking the time derivative of the volume. That is, we consider the partial derivative with respect to time of the volume V:

$$\frac{\partial V}{\partial t} = r^2\cos(\phi)\Delta\phi\Delta\theta\frac{\partial h}{\partial t}, \tag{2.5.1}$$

where we have utilized that r is constant, and that $\cos(\phi), \Delta\phi$ and $\Delta\theta$ are independent of the time t. We use u_θ and u_ϕ to denote the velocities in the θ and ϕ increasing directions, respectively. We are interested in the rate at which fluid volume enters the region from the sides. We can find this rate by considering the flux of fluid volume through the sides of the domain. That is, we consider how much fluid volume enters the domain from the θ direction, and how much fluid volume enters the domain from the ϕ direction. We calculate the influx at the θ line and the outflux at $\theta + \Delta\theta$ line, see Figure 2.4, to find the net flux. The influx is the area of the θ line times the velocity in the θ direction at the θ line. The influx is

$$u_\theta(\theta)h(\theta)r\Delta\phi,$$

where $h(\theta)$ is the height of the water at the θ line, assumed to be constant along the line. This way we can compute how much the volume changes due to the influx. For the outflux we do the same just for the $\theta+\Delta\theta$ line, introducing the notation $\theta' = \theta + \Delta\theta$ meaning the outflux is

$$u_\theta(\theta + \Delta\theta)h(\theta + \Delta\theta)r\Delta\phi = u_\theta(\theta')h(\theta')r\Delta\phi.$$

The net flux in the θ direction is the difference between the influx and the outflux, and is given by

$$\left(u_\theta(\theta)h(\theta) - u_\theta(\theta')h(\theta')\right) r\Delta\phi.$$

We can do the same for the ϕ direction, also using the notation $\phi' = \phi + \Delta\phi$. Using (2.5.1) and the net fluxes for both directions we can write:

$$r^2 h\cos(\phi)\Delta\phi\Delta\theta = \left(u_\theta(\theta)h(\theta) - u_\theta(\theta')h(\theta')\right) r\Delta\phi + \left(u_\phi(\phi)h(\phi)\cos(\phi) - u_\phi(\phi')h(\phi')\cos(\phi')\right) r\Delta\theta. \tag{2.5.2}$$

Since we are interested in the rate of change in the water height h with respect to time, we divide (2.5.2) by the area of the element $r^2\cos(\phi)\Delta\phi\Delta\theta$. Hence we get

$$\frac{\partial h}{\partial t} = \frac{u_\theta(\theta)h(\theta) - u_\theta(\theta')h(\theta')}{r\cos(\phi)\Delta\theta} + \frac{u_\phi(\phi)h(\phi)\cos(\phi) - u_\phi(\phi')h(\phi')\cos(\phi')}{r\cos(\phi)\Delta\phi}, \tag{2.5.3}$$

where $\phi \neq \pm\frac{\phi}{2}$. By collecting terms to the left hand side and changing the order of the numerators, we can rewrite (2.5.3) as

$$\frac{\partial h}{\partial t} + \frac{u_\theta(\theta')h(\theta') - u_\theta(\theta)h(\theta)}{r\cos(\phi)\Delta\theta} + \frac{u_\phi(\phi')h(\phi')\cos(\phi') - u_\phi(\phi)h(\phi)\cos(\phi)}{r\cos(\phi)\Delta\phi} = 0. \tag{2.5.4}$$

Next step is to investigate the limit values, as $\Delta\theta$ and $\Delta\phi$ goes to zero. We can find the limit values by using the definition of the derivative. The derivative of a function f with respect to x is defined as

$$f'(x) = \lim_{\Delta x\to 0}\frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

We can use this definition to find the limit values in (2.5.4).

$$\lim_{\Delta\theta\to 0}\frac{u_\theta(\theta')h(\theta') - u_\theta(\theta)h(\theta)}{\Delta\theta} = \frac{\partial}{\partial\theta}(hu_\theta),$$

and

$$\lim_{\Delta\phi\to 0}\frac{u_\phi(\phi')h(\phi')\cos(\phi') - u_\phi(\phi)h(\phi)\cos(\phi)}{\Delta\phi} = \frac{\partial}{\partial\phi}(hu_\phi\cos(\phi)).$$

Inserting these results in (2.5.4) yields

$$h_t + \frac{1}{r\cos(\phi)}\left((hu_\theta)_\theta + (hu_\phi\cos(\phi))_\phi\right) = 0,$$

which is the mass conservation equation in spherical coordinates and is the first equation in the shallow water equations in spherical coordinates. The next step is to derive the momentum equations in spherical coordinates. In this case, we focus on the horizontal velocity components, specifically the velocity tangential to the surface of the sphere, i.e., the θ and ϕ velocities. The vertical velocity is neglected, as the key assumption in the shallow water equations is that the vertical component of the acceleration is negligible. Additionally, when considering Earth, the water layer is thin compared to the radius of the Earth, referred to as a thin-layer approximation. We need to express the horizontal velocity u_h, which is dependent on the variables θ, ϕ and t. Since θ and ϕ are angles, we introduce the unit vectors $\mathbf{e}_\theta$ and $\mathbf{e}_\phi$ on the surface in the θ and ϕ directions, respectively. The unit vectors are illustrated in Figure 2.6.

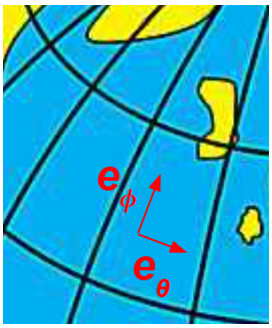


Figure 2.6: Illustration of the unit vectors $\mathbf{e}_\theta$ and $\mathbf{e}_\phi$.

We can express the horizontal velocity u_h in terms of the unit vectors as

$$u_h(\theta, \phi, t) = u_\theta \mathbf{e}_\theta + u_\phi \mathbf{e}_\phi. \tag{2.5.5}$$

We are then interested in the total derivative of the horizontal velocity u_h in (2.5.5) with respect to time. The total derivative is given by

$$\frac{\mathrm{d}u_h}{\mathrm{d}t} = \frac{\partial u_h}{\partial t} + \frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\partial u_h}{\partial \theta} + \frac{\mathrm{d}\phi}{\mathrm{d}t}\frac{\partial u_h}{\partial \phi}. \tag{2.5.6}$$

If we differentiate the longitude angle θ with respect to time, we get the angular velocity ω_θ in the θ direction, i.e.,

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \omega_\theta,$$

meaning that if $\omega_\theta > 0$, the point is moving eastwards, and if $\omega_\theta < 0$, the point is moving westwards. Similarly, if we differentiate the latitude angle ϕ with respect to time, we get the angular velocity ω_ϕ in the ϕ direction, i.e.,

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \omega_\phi,$$

meaning that if $\omega_\phi > 0$, the point is moving northwards, and if $\omega_\phi < 0$, the point is moving southwards. By using the arc length formula, we get that

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{u_\theta}{r\cos(\phi)}, \quad \frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{u_\phi}{r}. \tag{2.5.7}$$

We can now insert (2.5.7) into (2.5.6) to find the total derivative of the horizontal velocity split into the θ and ϕ directions:

$$\begin{aligned} \frac{\mathrm{d}u_\theta}{\mathrm{d}t} &= \frac{\partial u_\theta}{\partial t} + \frac{u_\theta}{r\cos(\phi)}\frac{\partial u_\theta}{\partial \theta} + \frac{u_\phi}{r}\frac{\partial u_\theta}{\partial \phi}, \\ \frac{\mathrm{d}u_\phi}{\mathrm{d}t} &= \frac{\partial u_\phi}{\partial t} + \frac{u_\theta}{r\cos(\phi)}\frac{\partial u_\phi}{\partial \theta} + \frac{u_\phi}{r}\frac{\partial u_\phi}{\partial \phi}. \end{aligned} \tag{2.5.8}$$

We know that the right hand side of (2.5.8) are the given physical forces acting on the fluid. Earlier in this project, we focused on the shallow water equations in Cartesian coordinates, accounting solely for gravitational forces. However, in spherical coordinates, additional physical forces must be considered. These include the Coriolis force, centripetal acceleration, and the effects of Earth's curvature. First we consider the gravitational force acting on the fluid, descibed as $-g\Delta h$, where g is the gravitational constant, and Δh is the gradient of the height h. We consider the gradient of $h(\theta, \phi)$:

$$\Delta h = \frac{1}{r\cos(\phi)} h_\theta \mathbf{e}_\theta + \frac{1}{r} h_\phi \mathbf{e}_\phi, \tag{2.5.9}$$

meaning that the gravity force acting on the fluid in the θ and ϕ directions are given by:

$$\begin{aligned}\theta - \text{direction:} \quad & -\frac{g}{r\cos(\phi)}h_\theta, \\ \phi - \text{direction:} \quad & -\frac{g}{r}h_\phi.\end{aligned}$$

Hence, we obtain the two momentum equations in spherical coordinates as

$$\begin{aligned}(u_\theta)_t + \frac{u_\theta}{r\cos(\phi)}(u_\theta)_\theta + \frac{u_\phi}{r}(u_\theta)_\phi &= -\frac{g}{r\cos(\phi)}h_\theta + \text{other forces}, \\ (u_\phi)_t + \frac{u_\theta}{r\cos(\phi)}(u_\phi)_\theta + \frac{u_\phi}{r}(u_\phi)_\phi &= -\frac{g}{r}h_\phi + \text{other forces}.\end{aligned} \tag{2.5.10}$$

The next force we consider is the Coriolis force, which is a force that acts on moving objects on the surface of the earth [17]. The Coriolis force is given by $f = 2\Omega\sin(\phi)$, where Ω is the angular velocity of the earth. The Coriolis force in the θ and ϕ directions are then given by:

$$\begin{aligned}\theta - \text{direction:} \quad & fu_\phi, \\ \phi - \text{direction:} \quad & -fu_\theta.\end{aligned}$$

The last thing we need to take into account when working in the spherical domain is the curvature of the earth. This adds the following terms:

$$\begin{aligned}\theta - \text{direction:} \quad & \frac{u_\theta u_\phi}{r}\tan(\phi), \\ \phi - \text{direction:} \quad & -\frac{u_\theta^2}{r}\tan(\phi).\end{aligned}$$

Inserting these forces into the momentum equations, we get the shallow water equations in spherical coordinates:

$$\left.\begin{aligned} h_t + \frac{1}{r\cos(\phi)}\left((hu_\theta)_\theta + (hu_\phi\cos(\phi))_\phi\right) &= 0, \\ (u_\theta)_t + \frac{u_\theta}{r\cos(\phi)}(u_\theta)_\theta + \frac{u_\phi}{r}(u_\theta)_\phi - \frac{u_\theta u_\phi}{r}\tan(\phi) + \frac{g}{r\cos(\phi)}h_\theta - fu_\phi &= 0, \\ (u_\phi)_t + \frac{u_\theta}{r\cos(\phi)}(u_\phi)_\theta + \frac{u_\phi}{r}(u_\phi)_\phi + \frac{u_\theta^2}{r}\tan(\phi) + \frac{g}{r}h_\phi + fu_\theta &= 0,\end{aligned}\right\} \tag{2.5.11}$$

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 θ and ϕ directions, g is the gravitational constant.

Obtain in vector form: Multiply with $\cos\phi$ and something else. Consider the paper.

Vector form:

$$\cos(\phi)\mathbf{U}_t + \mathbf{F}(\mathbf{U})_\theta + \cos(\phi)\mathbf{G}(\mathbf{U})_\phi = 0, \tag{2.5.12}$$

where

$$\mathbf{U} = \begin{bmatrix} h \\ hu_\theta \\ hu_\phi \end{bmatrix}, \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu_\theta \\ hu_\theta^2 + gh^2 \\ hu_\theta u_\phi \end{bmatrix}, \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hu_\phi \\ hu_\theta u_\phi \\ hu_\phi^2 + gh^2 \end{bmatrix}.$$

2.6 Linearized SWE in Spherical Coordinates

We consider the linearized shallow water equations in cartesian coordinates with one spatial dimension (x) and time (t).

$$\left.\begin{aligned} h_t + Hu_x &= 0, \\ u_t + gh_x &= 0,\end{aligned}\right\} \tag{2.6.1}$$

where h is the height of the water, u is the velocity of the water, H is the average height of the water, and g is the acceleration due to gravity. We now consider the linearized shallow water equations in spherical coordinates with one spatial dimension, the longitude (θ), and time (t). That is, we assume constant latitude ϕ.

In this section, we will derive the linearized shallow water equations in spherical coordinates. Later in the project, we will use the finite volume method to solve the linearized shallow water equations in spherical coordinates. We use the linearized SWE for simplicity. This section use the sources [18] and [19]. The linearized shallow water equations in spherical coordinates with one spatial dimension (θ) and time (t) are given by

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

where r is the radius of the Earth, ϕ is the latitude, and f is the Coriolis parameter, given by $f = 2\Omega\sin(\phi)$, where Ω is the angular velocity of the Earth. We still refer to the equations as the mass and momentum equations, respectively.

We can also write the linearized shallow water equations in spherical coordinates in vector form, without the coriolis force, as

$$\mathbf{W}_t + \mathbf{A}\mathbf{W}_\theta = 0, \tag{2.6.3}$$

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

Chapter 3

The Finite Volume Method

In this section, we present the finite volume method (FVM) for solving nonlinear systems of balance laws, specifically focusing on the shallow water equations (SWE). Nonlinear problems are more challenging than linear problems, as stability and convergence theory are more difficult. Our focus is on discountinuous solutions, that can accurately capture shock waves and other discontinuities. The approach described here is based on the work of LeVeque [20].

In finite volume methods, the computational domain is discretized into cells or control volumes. At the interfaces between these cells, we solve the local Riemann problem to compute the fluxes. These fluxes are then used to update the solution in each cell. By solving the Riemann problem at cell interfaces, the FVM can handle discontinuous solutions, making it particularly well suited for hyperbolic balance laws, such as the shallow water equations.

3.1 Finite Volume Methods for the 1D SWE

We begin by considering finite volume methods for the SWE in one space dimension. In the FVM, we discretize the domain into finite control volumes or cells:

$$V_i = [x_{i-1/2}, x_{i+1/2}] \times [t_n, t_{n+1}],$$

where $\Delta x = x_{i+1/2} - x_{i-1/2}$ is the length of the cell and $\Delta t = t_{n+1} - t_n$ is the time step. The cell V_i^n is illustrated in Figure 3.1.

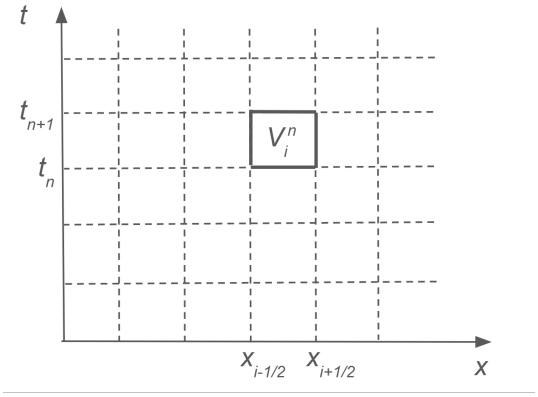


Figure 3.1: Illustration of the control volume V_i^n in the x, t plane.

For now, we will assume a uniform grid for simplicity. The finite volume formula is derived from the integral form (2.4.3). By dividing the integral form of the 1D inhomoegeneous SWE by the cell length Δx, we espress it in terms of the newly defined cells:

$$\begin{aligned}\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t_{n+1})\ \mathrm{d}x = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t_n)\ \mathrm{d}x \\ - \frac{\Delta t}{\Delta x}\left[\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{F}(\mathbf{U}(x_{i+1/2},t))\ \mathrm{d}t - \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{F}(\mathbf{U}(x_{i-1/2},t))\ \mathrm{d}t\right] \\ + \frac{\Delta t}{\Delta x \Delta t}\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{t_n}^{t_{n+1}} \mathbf{S}(\mathbf{U})(x,t)\mathrm{d}x\mathrm{d}t.\end{aligned}$$

For a finite volume V_i^n, averaging the terms over the volume yields the explicit conservative form

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n\right) + \Delta t \mathbf{S}_i. \tag{3.1.1}$$

The formula (3.1.1) is referred to as a finite volume scheme. The value $\mathbf{U}_i^n$ is the average value over the i-th cell at time t_n:

$$\mathbf{U}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t_n)\ \mathrm{d}x, \tag{3.1.2}$$

also known as the cell average. The flux $\mathbf{F}_{i-1/2}^n$ is the average flux across the line $x = x_{i-1/2}$ from time t_n to t_{n+1}:

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

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

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

The source term $\mathbf{S}_i$ is the average source term over the i-th cell at time t_n:

$$\mathbf{S}_i = \frac{1}{\Delta t \Delta x}\int_{t_n}^{t_{n+1}}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{S}(x,t)\ \mathrm{d}x\mathrm{d}t.$$

The values are illustrated in Figure 3.2.

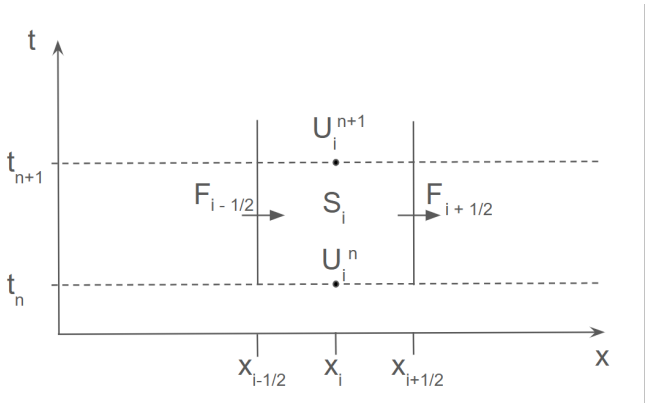


Figure 3.2: Illustration of the grid for the 1D SWE.

The central idea of the FVM is to define the numerical flux $\mathbf{F}^n_{i+1/2}$, at the cell interface, as a function of the cell averages $\mathbf{U}^n_i$ and $\mathbf{U}^n_{i+1}$, since the solution is known only in terms of these cell averages. Consequently, the FVM does not provide pointwise values of the solution, i.e., $\mathbf{U}(x,t)$, but instead gives cell-averaged values, $\mathbf{U}^n_i$, over the control volume. One of the main challenges in the FVM is to determine appropiate numerical flux functions that, based on the available cell averages, can reasonably approximate the fluxes at the cell interfaces. Later in the thesis, we will consider several numerical flux functions that can be used to solve the local Riemann problem at the cell interfaces.

Finite volume methods are closely related to Finite Difference Methods (FDM), but they differs as they are based on the integral form of the conservation laws. Where finite difference methods tend to break down near discontinuities in the solution, finite volume methods are more suited, since they are based on the integral form of the conservation laws. The key distinction between the FVM and the FDM lies in their formulation: while the FVM is based on the integral conservation over finite volumes, the FDM is based on the differential conservation over finite differences.

3.2 Finite Volume Method for the 2D SWE

We now extend the FVM to two space dimensions. Consider the 2D SWE in vector form (2.3.1) with $\mathbf{S}(\mathbf{U}) = 0$:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = 0. \tag{3.2.1}$$

Following the methods outlined in [2], an explicit finite volume scheme to solve (3.2.1) is given by

$$\mathbf{U}^{n+1}_{i,j} = \mathbf{U}^n_{i,j} + \frac{\Delta t}{\Delta x}(\mathbf{F}_{i-1/2,j} - \mathbf{F}_{i+1/2,j}) + \frac{\Delta t}{\Delta y}(\mathbf{G}_{i,j-1/2} - \mathbf{G}_{i,j+1/2}). \tag{3.2.2}$$

This is the unsplit finite volume method, meaning that, in a single step, the cell average $\mathbf{U}^n_{i,j}$ is updated using the fluxes from all intercell boundaries.

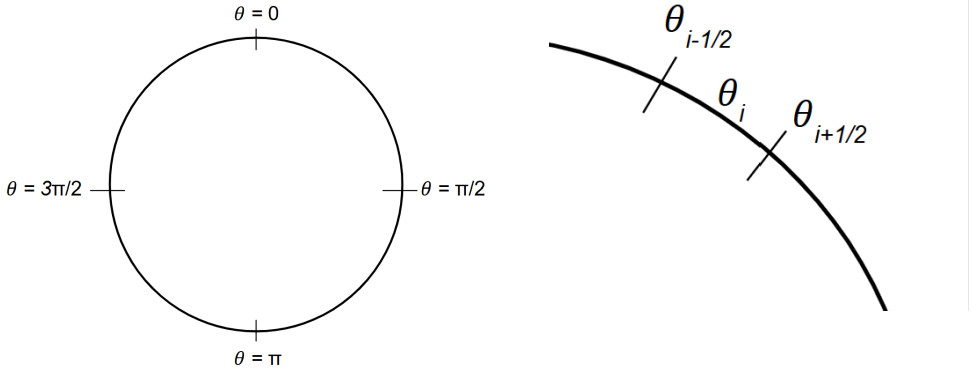
3.2.1 the MUSCL scheme

The MUSCL (Monotone Upwind Scheme for Conservation Laws), second order accurate in space and time. High-order total variation diminishing (TVD) scheme. We use the finite volume update:

$$\mathbf{U}^{n+1}_{i,j} = \mathbf{U}^n_{i,j} - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+1,j} - \mathbf{F}_{i,j}) - \frac{\Delta t}{\Delta y}(\mathbf{G}_{i,j-1} - \mathbf{G}_{i,j}). \tag{3.2.3}$$

3.3 Finite Volume Method for the 1D SWE in spherical coordinates

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. 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 $\Delta\theta = \frac{2\pi}{N}$. 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 3.3: 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:

$$\int_{\theta_L}^{\theta_R} \mathbf{W}_t \, \mathrm{d}\theta = \mathbf{A}\left(\mathbf{W}(\theta_L, t) - \mathbf{W}(\theta_R, t)\right). \tag{3.3.1}$$

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

$$\left.\begin{aligned} \frac{\partial}{\partial t}\int_{\theta_L}^{\theta_R} h \, \mathrm{d}\theta + \frac{H}{r\cos(\phi)}(u_R - u_L) = 0, \\ \frac{\partial}{\partial t}\int_{\theta_L}^{\theta_R} u \, \mathrm{d}\theta + g(h_R - h_L) + f u_i = 0. \end{aligned}\right\} \tag{3.3.2}$$

Then we integrate (3.3.1) 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

$$\int_{\theta_L}^{\theta_R} \mathbf{W}(\theta, t_2) \, \mathrm{d}\theta = \int_{\theta_L}^{\theta_R} \mathbf{W}(\theta, t_1) \, \mathrm{d}\theta - \mathbf{A}\left(\int_{t_1}^{t_2} \mathbf{W}(\theta_R, t) \, \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{W}(\theta_L, t) \, \mathrm{d}t\right), \tag{3.3.3}$$

which we refer to as the integral form of the linearized shallow water equations in spherical coordinates with one spatial dimension and time. We divide (3.3.3) with the cell length $\Delta\theta$ to obtain

$$\frac{1}{\Delta\theta}\int_{\theta_L}^{\theta_R}\mathbf{W}(\theta,t_2)\ \mathrm{d}\theta = \frac{1}{\Delta\theta}\int_{\theta_L}^{\theta_R}\mathbf{W}(\theta,t_1)\ \mathrm{d}\theta - \frac{\Delta t}{\Delta\theta}\mathbf{A}\left(\frac{1}{\Delta t}\int_{t_1}^{t_2}\mathbf{W}(\theta_L,t)\ \mathrm{d}t - \frac{1}{\Delta t}\int_{t_1}^{t_2}\mathbf{W}(\theta_R,t)\ \mathrm{d}t\right). \tag{3.3.4}$$

Averaging over the terms for a finite volume V_i^n gives the finite volume scheme for the linearized shallow water equations in spherical coordinates with one spatial dimension and time:

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \frac{\Delta t}{\Delta\theta}(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n), \tag{3.3.5}$$

The FVM scheme can be written as

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \frac{\Delta t}{\Delta\theta}(\mathbf{A}_{i+1/2}^n - \mathbf{A}_{i-1/2}^n) + \Delta t\mathbf{S}_i,$$

where $\mathbf{W} = \begin{bmatrix} h \\ u \end{bmatrix}$, $\mathbf{A}_{i+1/2}^n$ is the numerical flux at the cell interface $\theta_{i+1/2}$, and $\mathbf{S} = \begin{bmatrix} 0 \\ fu \end{bmatrix}$. With the cell averages:

$$\mathbf{W}_i^n = \frac{1}{\Delta\theta}\int_{\theta_L}^{\theta_R}\mathbf{W}(\theta,t_1)\ \mathrm{d}\theta$$

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}:

$$\mathbf{F}_{i-1/2}^n = \frac{1}{\Delta t}\mathbf{A}\int_{t_1}^{t_2}(\mathbf{W}(\theta_L,t))\ \mathrm{d}t,$$

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}:

$$\mathbf{F}_{i+1/2}^n = \frac{1}{\Delta t}\mathbf{A}\int_{t_1}^{t_2}(\mathbf{W}(\theta_R,t))\ \mathrm{d}t.$$

3.4 FVM 1D Linearized SWE spherical

The following scheme numerically solves the linearized 1D shallow water equations in spherical coordinates:

$$\frac{\partial h'}{\partial t} = -\frac{h_0}{a\cos\phi}\frac{\partial v}{\partial\theta}, \tag{3.4.1}$$

$$\frac{\partial v}{\partial t} = -g\frac{\partial h'}{\partial\theta} - fv, \tag{3.4.2}$$

where h' is the perturbation in water height, v is the velocity, h_0 is the mean water depth, g is the gravitational acceleration, f is the Coriolis parameter, a is the Earth's radius, and ϕ is the fixed latitude. The method employs a finite volume approach with a four-stage Runge-Kutta (RK4) time-stepping scheme to update h' and v.

At each time step, the fluxes of h' and v between neighboring cells are computed using:

$$\text{Flux} = \frac{1}{2}(q_{\text{left}} + q_{\text{right}}),$$

where q_{left} and q_{right} represent the state variables at adjacent cell edges.

For time integration, we use Runge-Kutta 4th order method, which is a numerical method to solve ordinary differential equations. The RK4 scheme approximates the time derivatives of h' and v using four intermediate stages:

Stage 1 (Initial Flux Evaluation):

$$k_{h1} = -\frac{h_0}{a\cos\phi}\frac{v_{\text{flux}} - \text{circshift}(v_{\text{flux}}, 1)}{\Delta\theta}, \tag{3.4.3}$$

$$k_{v1} = -g\frac{h'_{\text{flux}} - \text{circshift}(h'_{\text{flux}}, 1)}{\Delta\theta} - fv. \tag{3.4.4}$$

Stages 2–4 (Predictor Steps): Each stage recalculates the fluxes and updates the intermediate derivatives: We then update the state variables. The final values of h' and v are computed by combining the contributions from all four stages:

$$h' \leftarrow h' + \Delta t\left(\frac{1}{6}k_{h1} + \frac{1}{3}k_{h2} + \frac{1}{3}k_{h3} + \frac{1}{6}k_{h4}\right), \tag{3.4.5}$$

$$v \leftarrow v + \Delta t\left(\frac{1}{6}k_{v1} + \frac{1}{3}k_{v2} + \frac{1}{3}k_{v3} + \frac{1}{6}k_{v4}\right). \tag{3.4.6}$$

The perturbation in water height (h') is stored at each time step and visualized as a wave propagating around a great circle on a sphere. This scheme effectively combines finite volume flux computation with the high accuracy of the RK4 method to solve the 1D shallow water equations on a spherical domain.

3.5 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) [3]:

$$\begin{aligned} &\text{PDEs: } \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \\ &\text{ICs: } \mathbf{U}(x,0) = \begin{cases} \mathbf{U_L}, & \text{if } x < 0, \\ \mathbf{U_R}, & \text{if } x > 0. \end{cases} \end{aligned} \tag{3.5.1}$$

The vectors $\mathbf{U}$ and $\mathbf{F}(\mathbf{U})$ in (3.5.1) are given by

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ hvu \end{bmatrix}, \tag{3.5.2}$$

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 $t = 0$ in the left and right states of $x = 0$, respectively. The function $\mathbf{U}$ is piecewise constant, with a discontinuity at $x = 0$. The Riemann problem can be solved either exactly or approximately. Various approximate Riemann solvers exist, based on finding an approximate solution to the Riemann problem. Some of these solvers will be considered later in the thesis.

3.5.1 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 (3.5.1). 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.

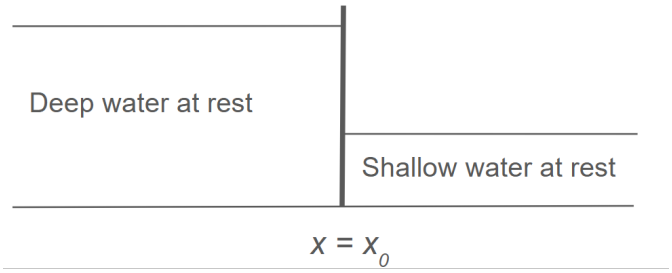


Figure 3.4: Initial conditions for the dam-break problem. An infinitely thin wall at $x = x_0$ divides two sections of water with different water levels.

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 $t = 0$.

3.5.2 Waves in the Riemann problem

To get a better understanding of the flow in shallow water, we provide some very short background information about waves. In particular the wave structure in the solution of the Riemann problem (3.5.1), which consists of a combination of waves, including shock waves and rarefaction waves. In the solution of the Riemann problem (3.5.1) there are four possible wave patterns outcomes, which are combinations of shock waves and rarefaction waves. In each case there are three waves, the left and right waves correspond to the one-dimensional SWE, and the middle wave aries from the $y-$momentum equation in (3.5.1) and is always a shear wave. 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, and (d) Both left and right shock. The region between the left and right waves is called the star region. 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.

3.5.3 Exact Riemann solver

The exact Riemann solver is a method that solves the Riemann problem exactly, and it is based on the solution of the Riemann problem (3.5.1). There exist exact Riemann solvers which are very efficient and leads to Gudonov methods, that are only slightly more expensive than those based on approximate Riemann solvers [1]. However, in this project we will focus on approximate Riemann solvers, which are able to solve the Riemann problem with high accuracy and efficiency.

3.6 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 (3.5.1) 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.

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 [3]. 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

$$\mathbf{F}_{1+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } S_L \geq 0, \\ \mathbf{F}^{HLL} \equiv \frac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_LS_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & \text{if } S_L \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R \leq 0. \end{cases} \tag{3.6.1}$$

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

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

Here $\hat{h}$ is an estimate for the water depth in the star region, h_*. In the two-rarefaction Riemann Solver, the water depth h in the star region is given by

$$h_* = \frac{1}{g}\left(\frac{1}{2}(a_L + a_R) + \frac{1}{4}(u_L - u_R)\right)^2, \tag{3.6.2}$$

which is what we use in this project for $\hat{h}$ in the HLL solver. Since this is a two-wave model, it is complete for one dimensional problems, but for the augmented system of equations in two dimensions, the HLL solver is not complete, as it ignores the middle wave, the shear wave. This motivates the use of the HLLC solver, which is a modification of the HLL solver.

HLLC

The HLLC (Harten, Lax, van Leer, Contact) solver is an extension of the HLL solver, which includes the middle wave, i.e., it is a three-wave model. In addition to the wave speeds S_L and S_R, the HLLC solver also requires the speed of the middle wave S^*. We can write the HLLC numerical flux as

$$\mathbf{F}_{i+\frac{1}{2}}^{HLLC} = \begin{cases} \mathbf{F}_L & \text{if } 0 \leq S_L, \\ \mathbf{F}_{*L} & \text{if } S_L \leq 0 \leq S^*, \\ \mathbf{F}_{*R} & \text{if } S^* \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R \leq 0. \end{cases}$$

The fluxes $\mathbf{F}_{*L}$ and $\mathbf{F}_{*R}$ are given by

$$\mathbf{F}_{*L} = \mathbf{F}_L + S_L(\mathbf{U}_L - \mathbf{U}_{*L}),$$
$$\mathbf{F}_{*R} = \mathbf{F}_R + S_R(\mathbf{U}_R - \mathbf{U}_{*R}),$$

and the middle states $\mathbf{U}_{*L}$ and $\mathbf{U}_{*R}$ are given by

$$U_{*K} = h_K \left(\frac{S_K - u_K}{S_K - S_*} \right) \begin{bmatrix} 1 \\ S_* \\ \psi_K \end{bmatrix}.$$

The function ψ_K can represent either a passive scalar $\psi(x,t)$ or the velocity component $v(x,t)$ if we consider the two-dimensional shallow water equations. Mathematically $\psi(x,t)$ and $v(x,t)$ behave identically. An estimate for the middle wave speed S^* can be calculated as

$$S^* = \frac{S_L h_R(u_R - S_R) - S_R h_L(u_L - S_L)}{h_R(u_R - S_R) - h_L(u_L - S_L)},$$

where S_L and S_R are the wave speeds of the left and right waves, respectively.

Rusanov

The Rusanov flux uses the HLL framework, but with a different choice of wave speeds. To obtain the flux, we assume an estimate S^+ for the positive wave speed is available. Then we set

$$S_L = -S^+, \quad S_R = S^+. \tag{3.6.3}$$

By substituting (3.6.3) into the $\mathbf{F}^{HLL}$ in (3.6.1), we obtain the Rusanov flux as

$$\mathbf{F}_{i+\frac{1}{2}}^{Rus} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) - \frac{1}{2}S^+\left(\mathbf{U}_R - \mathbf{U}_L\right), \tag{3.6.4}$$

where a simple estimate for the wave speed S^+ is given by

$$S^+ = \max(|S_L|, |S_R|).$$

There are some requirement for S^+ in (3.6.4) to ensure stability. It must hold that

$$S^+ \leq \frac{\Delta x}{\Delta t},$$

where $\frac{\Delta x}{\Delta t}$ is called the mesh speed. This Rusanov sheme is upwind and based on a one-wave model. Therefore it is an incomplete Riemann solver.

Lax-Friedrichs

In the Lax-Friedrichs method, we use the Rusanov flux, but with a different choice of wave speed. That is, we set the wave speed S^+ as the largest possible speed, while still ensuring stability, i.e.,

$$S^+ = \frac{\Delta x}{\Delta t}. \tag{3.6.5}$$

By inserting the wave speed (3.6.5) into the Rusanov flux (3.6.4), we obtain the Lax-Friedrichs flux as

$$\mathbf{F}^{LF}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) - \frac{1}{2}\frac{\Delta x}{\Delta t}\left(\mathbf{U}_R - \mathbf{U}_L\right),$$

where $\mathbf{F}_L = \mathbf{F}(\mathbf{U}_L)$ and $\mathbf{F}_R = \mathbf{F}(\mathbf{U}_R)$. The Lax-Friedrichs method is a centred method, which is first-order accurate.

Lax-Wendroff

There are several versions of the Lax-Wendroff flux, but in this thesis we will use the following flux:

$$\begin{aligned} \mathbf{U}^{LW}_{i+\frac{1}{2}} &= \frac{1}{2}\left(\mathbf{U}_L + \mathbf{U}_R\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{F}_R - \mathbf{F}_L\right), \\ \mathbf{F}^{LW}_{i+\frac{1}{2}} &= \mathbf{F}(\mathbf{U})^{LW}_{i+\frac{1}{2}}. \end{aligned} \tag{3.6.6}$$

The Lax-Wendroff method is a centred method, which is second-order accurate in space and time.

FORCE

The FORCE scheme (First-Order Centred) is a combination of Lax-Friedrichs and Lax-Wendroffs fluxes. The numerical flux is given by

$$\mathbf{F}^{FO}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}^{LF}_{i+\frac{1}{2}} + \mathbf{F}^{LW}_{i+\frac{1}{2}}\right),$$

where $\mathbf{F}^{LF}_{i+\frac{1}{2}}$ is the Lax-Friedrichs flux and $\mathbf{F}^{LW}_{i+\frac{1}{2}}$ is the Lax-Wendroff flux. The FORCE scheme is first-order accurate. It is possible to extend the FORCE scheme to multiple dimensions on structured meshes by using dimensional splitting.

Chapter 4

Fourier Neural Operators and Neural Networks

The FVM, together with other numerical solvers such as the FDM and FEM (Finite Element Method), solves PDEs by discretizing the domain into a grid. The finer the grid, the more accurate the solution, but also the more computationally expensive the solution. This introduces a trade-off between accuracy and computational cost. Complex PDEs often require a fine grid to capture the solution accurately, which can be computationally expensive. The hope for data-driven methods is that, by learning the dynamics of the solution, we can reduce the computational cost of solving PDEs and still maintain a high level of accuracy. A classical neural network (NN), is able to learn a map from input to output, i.e., a map between finite-dimensional spaces. Fourier Neural Operators stands out at they are able to learn mappings between function spaces, meaning they are also grid-independent. The neural operator requires data only, and not the PDE itself. This is obiviously a great advantage, if we are aiming to describe systems where the PDE is unknown. The neural operator is also able to transfer solutions between meshes, meaning that we can train the model on a coarse grid and transfer the solution to a fine grid, depending on the desired accuracy. This is also referred to as the zero-shot super-resolution property. We make the neural operator to a fourier neural operator, by using a fourier integral operator in the neural network. We utilize the fact that differentation with respect to time is equivalent to multiplication in the Fourier domain. (Not to be confused with the convolution theorem).

Another issue we are facing in this project is non-linearities. Standard neural networks uses combinations of linear multiplications and non-linear activation functions to approximate non-linear functions. Whereas, neural operators approximate non-linear operators, by combining linear functions and non-linear activation functions with global integral operators.

4.1 Convolutional Neural Networks

Some text about CNNs.

4.2 Fourier Neural Operators

In this section, we will introduce the concept of Fourier Neural Operators (FNO). The theory and method described here is based on the paper [10].

We consider the operator $G : A \to U$, that maps from a 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 can access the data by point wise evaluations of the functions, i.e., we have access to the observations $\{a_j, u_j\}_{j=1}^N$, in a domain $D \subset \mathbb{R}^d$, which is bounded open set. The neural operator is iterative, where the 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 $W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function. 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. We aim for a multi-step prediction model, which can predict a given number of time steps.

The network for the FNO model is illustrated in Figure 4.1.

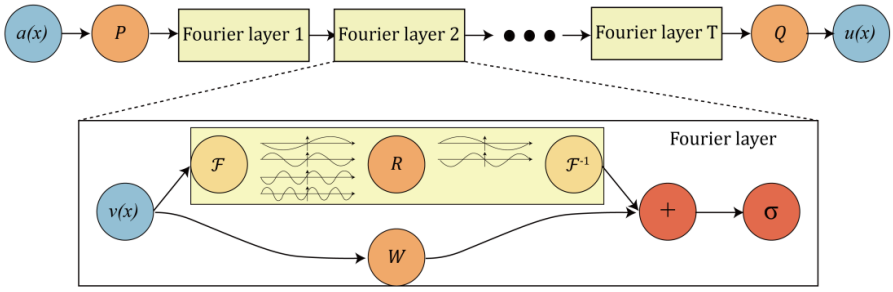


Figure 4.1: An overview of the network architecture with several Fourier layers. Illustration from [10].

From the figure we see that the network consists of a input layer P, several Fourier layers and a output layer Q. A Fourier layer consists of two parallel paths. The top path consists of a Fourier transformation $\mathcal{F}$, a linear transformation R to filter out the higher Fourier modes, and a inverse Fourier transformation $\mathcal{F}^{-1}$. The bottom path consists of a linear transformation W. The paths meet and there are applied an activation function σ.

Input-output mapping:

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

Chapter 5

Data generation

In this chapter we will outline how the data needed for the data-driven methods is generated. We will also present the mesh generation for the sphere, which is used to solve the SWE in spherical coordinates. The data generation is done by solving the SWE in 1D using the FVM with the Godunov scheme and the exact Riemann solver. We use Gaussian functions with parametric extension [21] to generate the initial conditions.

5.1 True solution of the SWE

In this section, we present how the so-called true solution is found in the code by solving the Riemann problem exactly. The true solution is found by solving the Riemann problem exact, with 5000 cells, and distinguishing between the wetbed or drybed case, and also identifying the shock and rarefaction waves. First we caculate the wave speeds for the left and right states, respectively, as

$$c_L = \sqrt{gh_L}, \quad c_R = \sqrt{gh_R},$$

which are used to determine the critical water height h_{crit} as

$$h_{\text{crit}} = (u_R - u_L) - 2(c_L + c_R).$$

If either $h_L \leq 0$ or $h_R \leq 0$, we are in a drybed case. If $h_{\text{crit}} \geq 0$ it means the water depth is somehow critical, and we are in a drybed case, If none of the above conditions are met, we are in a wetbed case. Summarized:

$$\begin{cases} \text{Dry-bed case} & \text{if } h_L \leq 0, \quad h_R \leq 0 \text{ or } h_{\text{crit}} \geq 0, \\ \text{Wet-bed case} & \text{otherwise.} \end{cases}$$

For a dry bed case, we then identify where the dry is located, i.e., if the left side is dry, the right side is dry, or the middle is dry, and calculate the wave speeds accordingly. For a wet bed case, we compute the characteristics h_* and u_* for the star region. We then identify the shock and rarefaction waves, and calculate the wave speeds for the left and right states, respectively.

FLOW-DIAGRAM.

5.2 Data generation

We use the Matlab script: SWE1D-data-generation to generate the data for the training and testing of the neural network, and the Python script: data-generation to load and visualize the data. The data generation is done by

solving the SWE in 1D using the FVM with the Godunov scheme and the exact Riemann solver. We use Gaussian functions with parametric extension [21] to generate the initial conditions, that is, functions on the form

$$h(x,0) = a \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right),$$

where a is the amplitude of the Gaussian, μ is the mean value, and σ is the standard deviation. We solve the 1D SWE with the following parameters:

- $N = 200$ cells,
- From $t = 0.0$ to $t_{\text{end}} = 1.0$,
- $x \in [0, 1]$,
- $u(x, 0) = 0$,
- $b(x) = 0.0$,
- $g = 9.81$ and
- $\sigma = 0.1$.

The value of μ is varied to generate different initial conditions, as seen in Figure 5.1.

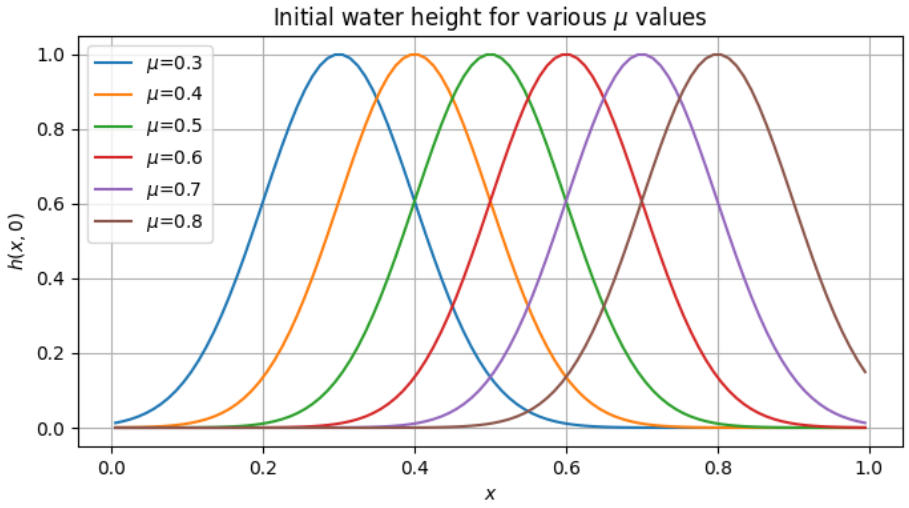


Figure 5.1: Initial conditions for the data generation.

5.3 Data Copernicus

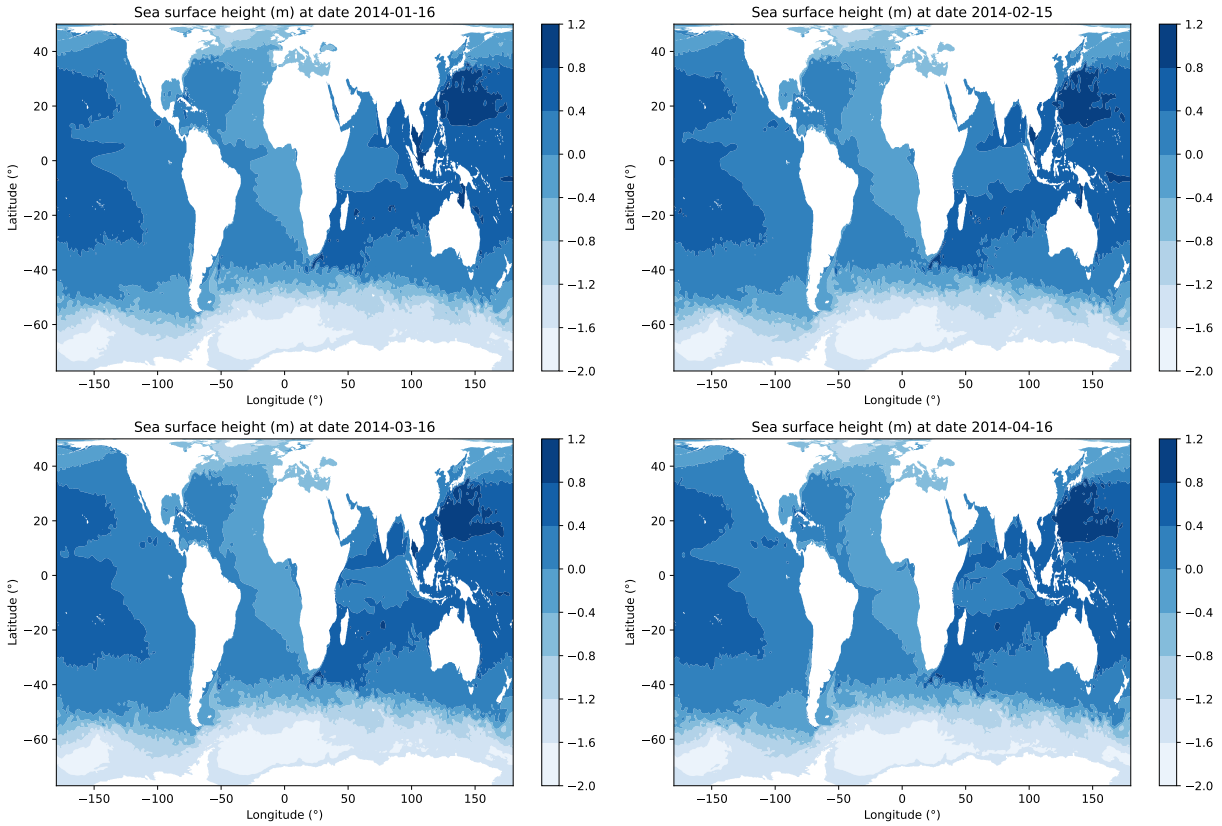


Figure 5.2: Sea surfae height as the diffference from reference sea surface height for the months jan-apr 2014.

5.4 Mesh generation for the sphere

To solve the SWE on the sphere, we must use another grid, than the regular grid used in the 2D case. Hence, we use the icosahedral grid, which is a grid that approximates the sphere with triangles. The grid can be generated in different generations, depending on the level of detail we need. For each time we refine to a higher level, the number of triangles increases by a factor of four, i.e., we split each triangle into four smaller triangles. Meaning that the number of triangles is increasing drastically with each level of refinement. The grid is generated by the Github: https://github.com/siddharth-maddali/SphereMesh/tree/master and then rewritten to Python. The grid for the first 4 levels of refinement is shown in Figure 5.3. For simplicity, we will begin by considering the first level of refinement, which consists of 20 triangles. The matrix tri is a 20×3 matrix, where each row represents a triangle and the three columns represent the three vertices of the triangle. The vertices are stored in the matrix P, which is a 12×3 matrix, where each row represents a vertex and the three columns represent the x, y, and z coordinates of the vertex. The vertices are normalized to the unit sphere, i.e., the radius of the sphere is 1. The idea is now, that similar to the case in cartesian coordinates, we loop through each cell, in this case triangles, and calculate the fluxes between the cells. We must be aware of which cells/triangles are neighbors, and we must also be aware of the orientation of the triangles, i.e., the normal vector of the triangle. The normal vector is calculated by the cross product of the two vectors that span the triangle. The normal vector is then normalized to the unit sphere. The matrix tri is my Element to Vertex matrix. I also need an Element-to-Face table and a Element-to-Element table. The Element-to-Face table is used to define which edges or faces belong to each triangle. Each face of a

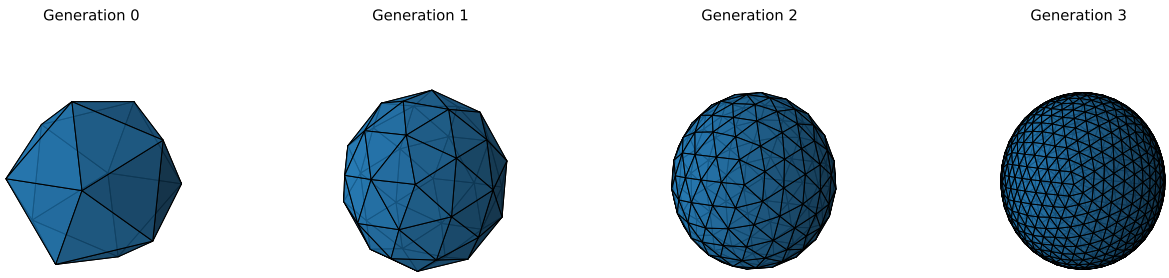


Figure 5.3: Icosahedral grid for the first 4 levels of refinement.

triangle is an edge shared between two triangles. The Element-to-Element table is used to define which triangles are neighbors. That is, it indicates which triangles share an edge. This is important when calculating the fluxes between the triangles. To construct this table we loop through each triangle and check if the edge of the triangle is shared with another triangle.

We make the FVM to solve SWE for a triangular grid on the sphere. For each triangle, we consider each face (edge). For each face we define the normal vector in terms of the spherical unit vectors $\mathbf{e}_\theta$ and $\mathbf{e}_\phi$. The spherical unit vectors at a point are:

$$\mathbf{e}_\theta = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ 0 \end{bmatrix},$$
$$\mathbf{e}_\phi = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta) \\ -\sin(\phi) \end{bmatrix}.$$

Where $\mathbf{e}_r$ is the radial direction, going outward from the origin. $\mathbf{e}_\theta$ is the longitude direction, and $\mathbf{e}_\phi$ is the latitude direction. The unit vectors are tangensial to the sphere. We do not need to consider the radial direction, as the SWE is only in the tangential directions.

Now we have the cartesian edge vectors in tangential directions, $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$.

The projection of an edge vector $\mathbf{e}_i = (e_{ix}, e_{iy}, e_{iz})$ onto $\mathbf{e}_\theta$ is:

$$\mathbf{e}_i \cdot \mathbf{e}_\theta = e_{ix}\left(-\sin\theta\cos\phi\right) + e_{iy}\left(-\sin\theta\sin\phi\right) + e_{iz}\cos\theta$$

Similarly, the projection onto $\mathbf{e}_\phi$ is:

$$\mathbf{e}_i \cdot \mathbf{e}_\phi = e_{ix}\left(-\sin\phi\right) + e_{iy}\cos\phi$$

Recall that the cross product between two vectors in 3D produces a third vector that is orthogonal to the two input vectors, i.e., the plane spanned by the two input vectors.

For a given face f of a triangle, we first calculate the Cartesian face normal $\mathbf{n}_f$ as the cross product of the two

vectors that span the face. For a given face f of a triangle, we define the normal vector $\mathbf{n}_f$ is decomposed into the spherical unit vectors as

Chapter 6

Numerical results

In this section we present the results of the numerical experiments, together with the results from the data-driven methods, including neural networks and Fourier neural operators. In chapter 6 we present the results from the finite volume method (FVM), where we have implemented the method for solving the shallow water equations in 1D and 2D and tested it on several problems. We also address the scalability issues of the finite volume method. In chapter 7 we present the results from the data-driven methods, where we have implemented neural networks and Fourier neural operators to solve the shallow water equations in 1D. We compare the results from the data-driven methods with the results from the FVM, and discuss the advantages and disadvantages of the different methods. We expand and in section 7.3 we present the results from the data-driven methods for solving the shallow water equations in 2D.

In this chapter we present the results of the numerical experiments, where we have implemented the finite volume method for solving the shallow water equations in 1D and tested it on several problems. [1] A key focus is to validate the implementation, as it will generate data for the data-driven methods, including neural networks and Fourier neural operators. To test the implementation, we have solved the 1D dam break problem, and the five test cases from Toro's book [1]. These problems are all discontinuous in either the water height h or the fluid velocity u. The idea is, that if the numerical solution can capture the discontinuities, it should be well-suited to handle smoother solutions as well. Finally, we have tested the implementation on the 2D idealised circular dam break problem, which is also from Toro's book [3]. The results from the 2D problem are compared to the results from the book to validate the implementation. Lastly, we have tested the scalability of the FVM to solve the 2D SWE, by running the 2D problem with a Gaussian initial condition for different values of N, i.e., the number of cells in each direction.

6.1 The 1D Dam Break Problem

First we solve the 1D dam break problem, with the following initial conditions:

$$h(x,0) = \begin{cases} h_L, & \text{if } x < x_0, \\ h_R, & \text{if } x > x_0, \end{cases}$$

where $x \in [0, 50], h_L = 3.5$ m, $h_R = 1.25$ m and $x_0 = 20$ m. Since it is a dam break problem the initial fluid velocity is zero, i.e., $u(x,0) = 0$. We solve the problem starting at $t = 0$ and ending at $t = 2.5$ seconds. The numerical solution to the 1D Dam Break Problem using the FVM, together with the true solution, provided from the course [22], can be seen in Figure 6.1.

[1] Code and small animations can be found at github, visit `https://github.com/MelissaJessen/Shallow-Water-Equations`.

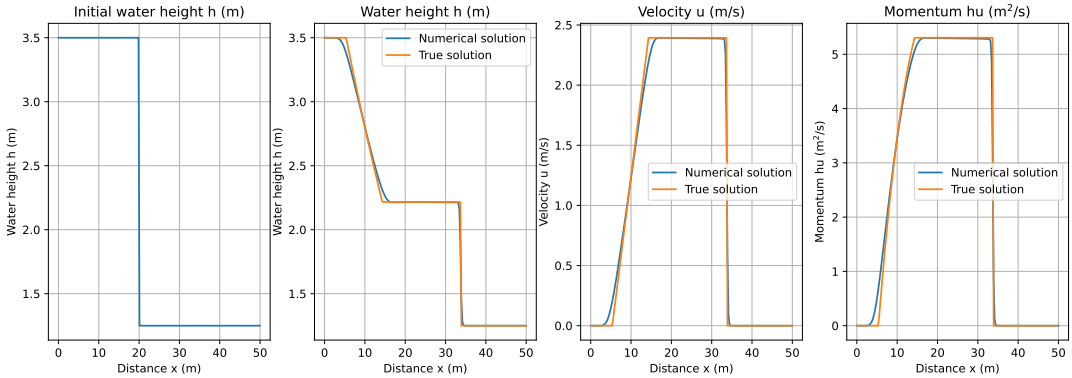


Figure 6.1: The initial water height h at $t = 0$, together with the water height, the fluid velocity u and the momentum hu after $t = 2.5$ seconds.

From Figure 6.1 we see that the numerical solution aligns well with the true solution, and successfully captures the discontinuity.

6.2 Toro test cases

We have tested the method on the five test cases for Riemann problems from Toros book [1]. The initial conditions for the five test cases are given in Table 6.1.

Test case	h_L	u_L	h_R	u_R	x_0	t_{end}
1	1.0	2.5	0.1	0.0	10.0	7.0
2	1.0	-5.0	1.0	5.0	25.0	2.5
3	1.0	0.0	0.0	0.0	20.0	4.0
4	0.0	0.0	1.0	0.0	30.0	4.0
5	0.1	-3.0	0.1	3.0	25.0	5.0

Table 6.1: Initial conditions for the five test cases.

The domain is $x \in [0, 50]$ for all test cases. The Riemann problems are chosen to test the method on different types of waves, such as shock waves and rarefaction waves. To solve the test cases we have used the following fluxes:

1. Godunov method with exact Riemann solver,
2. Lax-Friedrich flux,
3. Lax-Wendroff flux,
4. FORCE flux,
5. HLL flux,

Test case 1

The initial conditions for test case 1 are given in Figure 6.2 and the final solutions after $t = 7.0$ seconds are given in Figure 6.3. In test case 1, we observe a right shock wave and a left rarefaction wave.

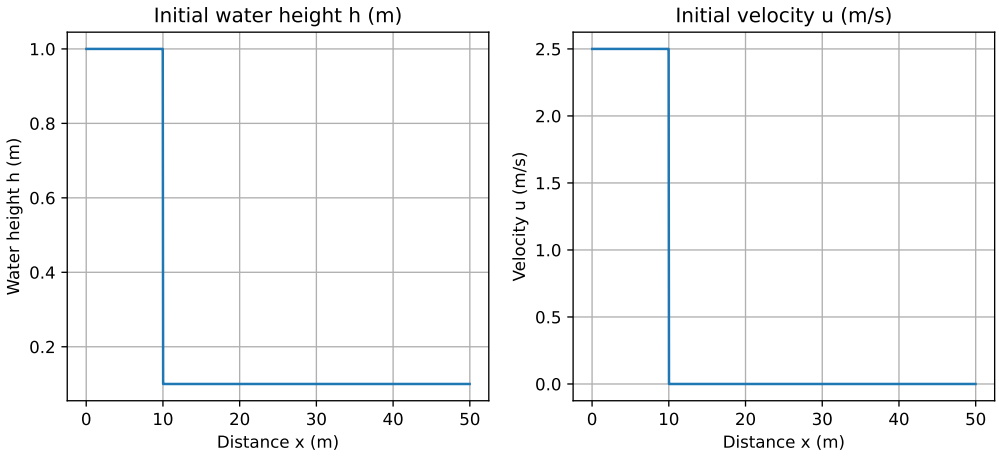


Figure 6.2: Initial conditions for the test case.

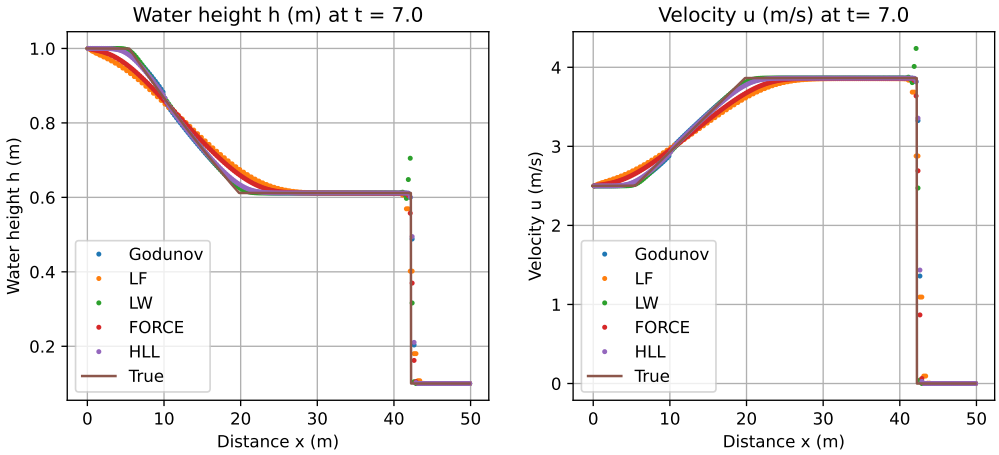


Figure 6.3: Final solution for the test case.

For this test case all the fluxes work well, but there are minor differences in the solution, which can be seen in Figure 6.3. We also see that Lax-Wendroff flux has some oscillations in the solution, which is not present in the other fluxes.

Test case 2

The initial conditions for test case 2 are illustrated in Figure 6.4.

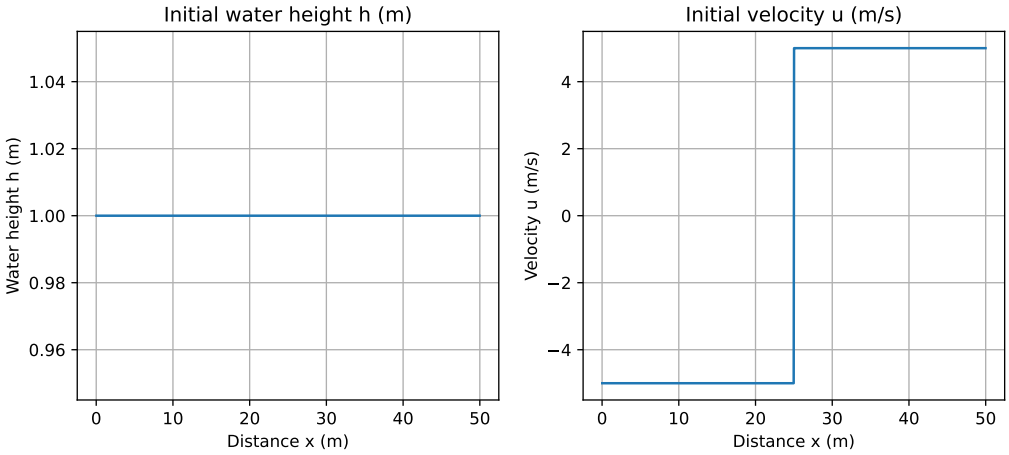


Figure 6.4: Initial conditions for the test case.

In test case 2 we have two rarefaction waves, one on the left side and one on the right side. As they are travelling in opposite directions (away from each other), there will be created a nearly dry bed in the middle of the domain. Many methods have difficulties with this test case as they may compute a negative water height. For these experiments we were able to get close to the true solution, using Lax-Friedrich flux, FORCE flux and HLL flux. The final solutions after $t = 2.5$ seconds are illustrated in Figure 6.5.

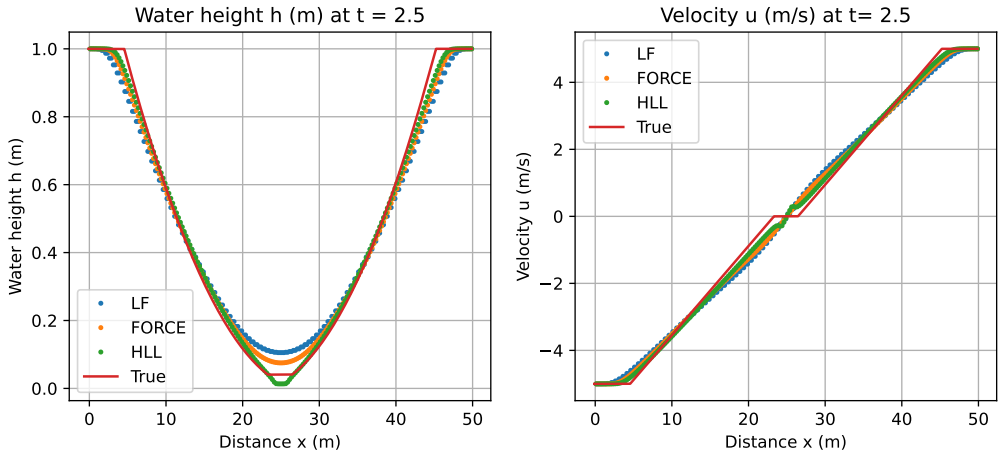


Figure 6.5: Final solution for the test case.

For the fluxes, Godunov method with exact Riemann solver, and Lax-Wendroff, it was not possible to get an acceptable solution.

Test case 3

The initial conditions for test case 3 are given in Figure 6.6, and the final solutions after $t = 4.0$ seconds are given in Figure 6.7.

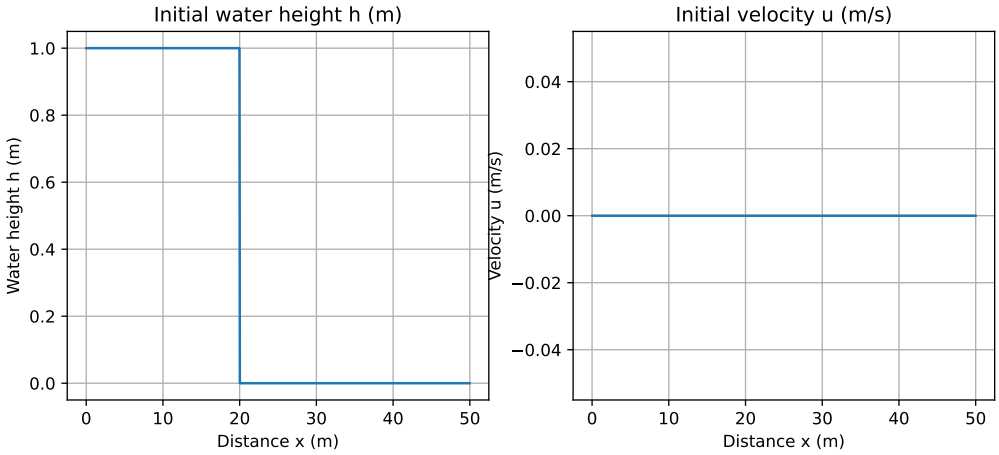


Figure 6.6: Initial conditions for the test case.

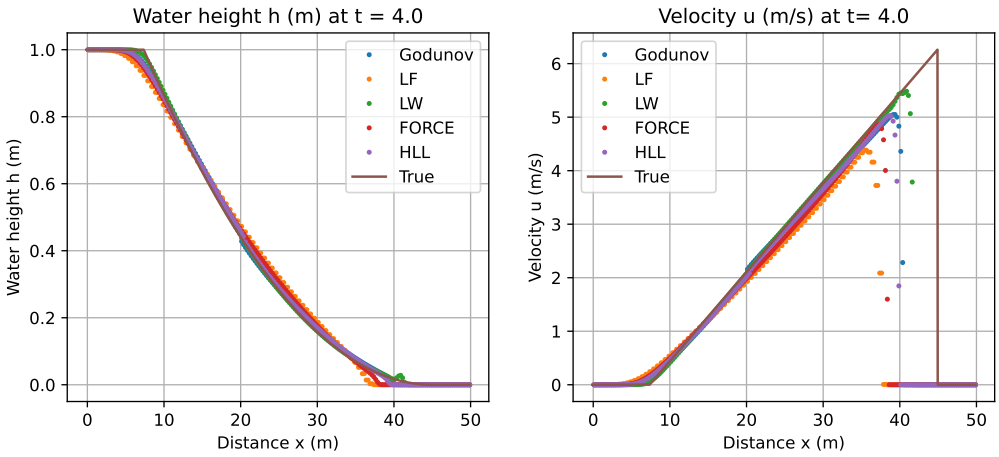


Figure 6.7: Final solution for the test case.

To solve case 3, with the FVM we must add a small amount to h_R, since the code does not handle $h_R = 0$ well. We set $h_R = 0.00005$ to solve it numerically, but the true solution is for $h_R = 0$. By running experiments with different values of h_R, we see that the solution converges to the true solution as h_R approaches 0. The solution consists of a left rarefaction wave. From Figure 6.7 we see that when it comes to predicting the velocity, there are some differences in how the different fluxes perform.

Test case 4

The initial conditions for test case 4 are given in Figure 6.8, and the final solutions after $t = 4.0$ seconds are given in Figure 6.9.

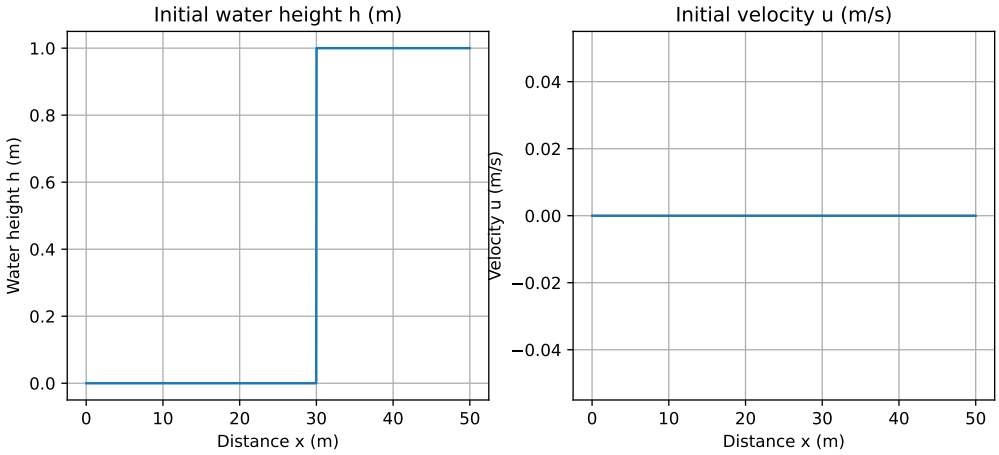


Figure 6.8: Initial conditions for the test case.

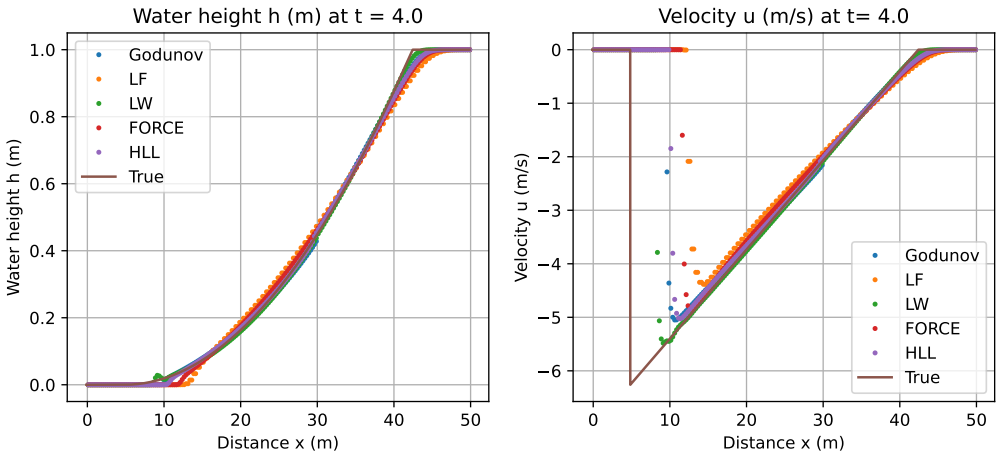


Figure 6.9: Final solution for the test case.

In case 4 we face the same challenges as in case 3. We set $h_L = 0.00005$, and the solution converges to the true solution as h_L approaches 0. This test case is symmetric to test case 3, and the solution consists of a right rarefaction wave. The case is included to test if the results are as expected. As in test case 4, we observe differences in the fluxes performance.

Test case 5

The initial conditions for test case 5 are given in Figure 6.10, and the final solutions after $t = 5.0$ seconds are given in Figure 6.11.

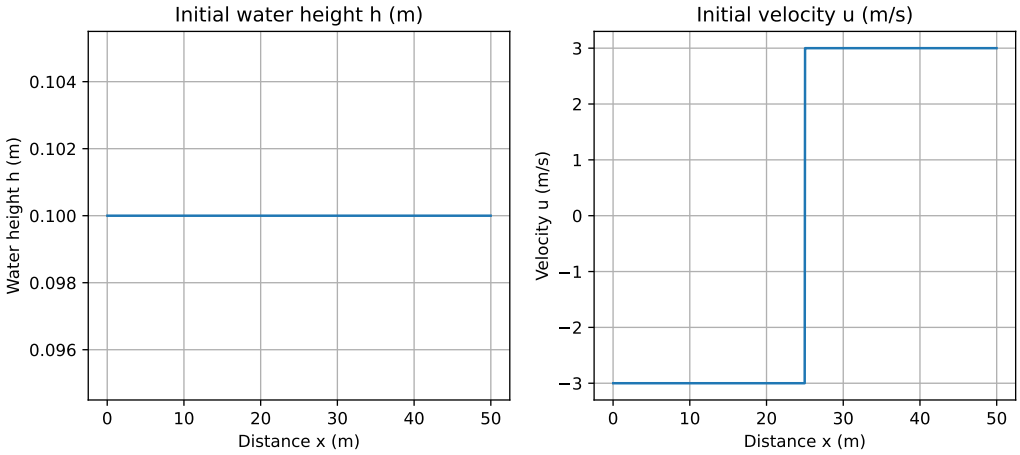


Figure 6.10: Initial conditions for the test case.

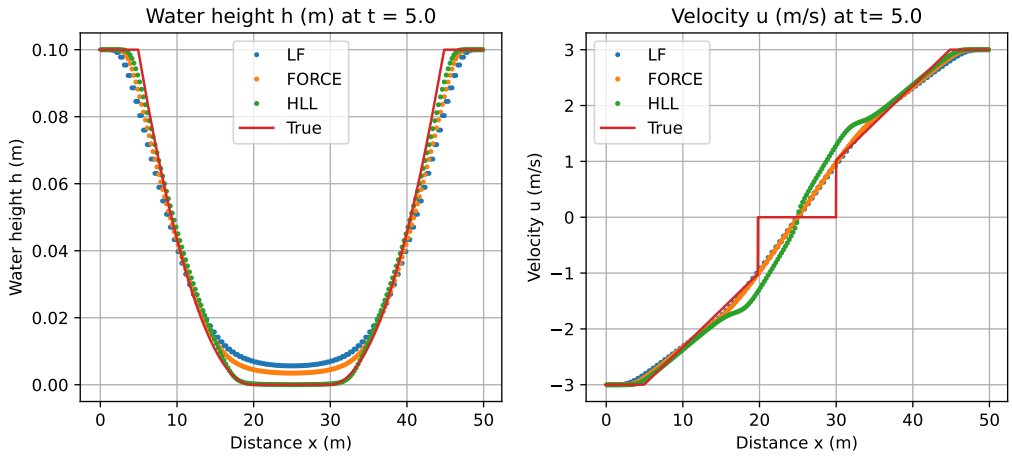


Figure 6.11: Final solution for the test case.

From Figure 6.11 we see that the numerical solutions for the velocity v at $t = 5.0$ are smooth, where the true solution is discontinuous. In this test case there are also challenges with some of the fluxes due to the generation of a dry-bed region. The fluxes that are not able to solve this case are Godunov method with exact Riemann solver and Lax-Wendroff flux, the same as in test case 2. The solution consists of two rarefaction waves, one on the left side and one on the right side, and a dry-bed region in the middle.

To get an overview of which fluxes that were able to produce solutions for the test cases, consider Table 6.2. The table shows which fluxes that were able to produce a solution for the test cases.

Test case	Godunov	LF	LW	FORCE	HLL
1	✓	✓	✓	✓	✓
2	×	✓	×	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓
5	×	✓	×	✓	✓

Table 6.2: Overview of which fluxes that were able to produce solutions for the test cases.

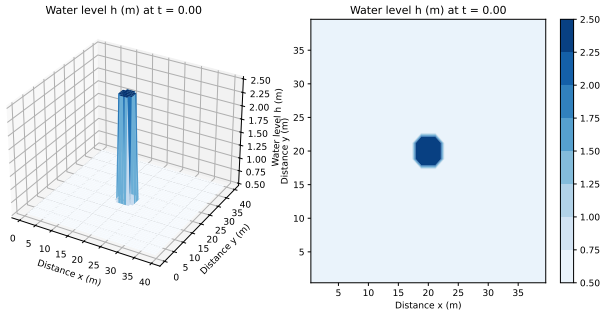
Note, that as we see in the results, there are still differences in the solutions accuracy between the fluxes that were able to solve the test cases.

6.3 2D idealised Circular Dam Break Problem

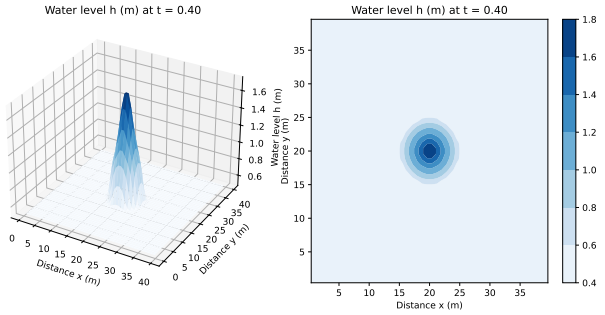
We now proceed to the 2D case, focusing on an idealised circular dam break problem over a horizontal bottom. This problem is also from Toro's book [3]. We assume there is an infinitely thin circular wall at radius $R = 2.5$ m is a square domain of size 40×40 with centre at $(x_c, y_c) = (20, 20)$. The initial conditions are

$$h(x, y, 0) = \begin{cases} 2.5\ m, & \text{if } \sqrt{(x - x_c)^2 + (y - y_c)^2} \leq R, \\ 0.5\ m, & \text{otherwise,} \end{cases}$$
$$u(x, y, 0) = 0,$$
$$v(x, y, 0) = 0.$$

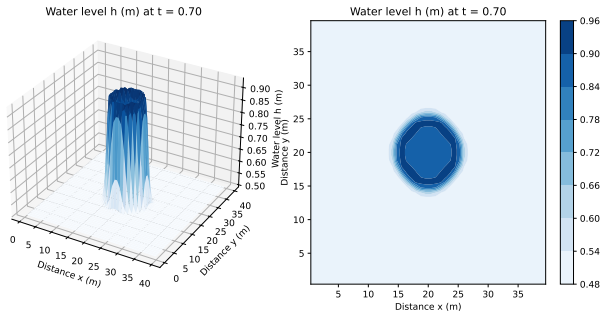
We use a mesh of size 200×200. The results after $t = 0.0, 0.4, 0.7$ and 1.4 seconds are given in Figure 6.12.



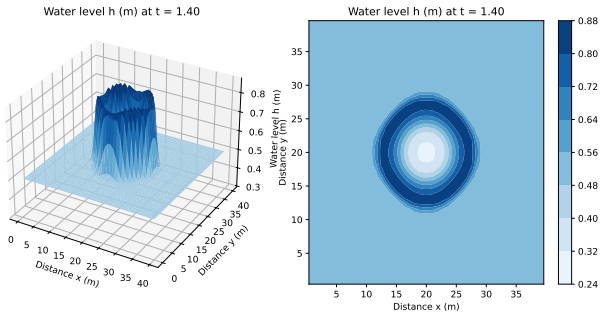
(a) 2D dam break problem after $t = 0$.



(b) 2D dam break problem after $t = 0.4$.



(c) 2D dam break problem after $t = 0.7$.



(d) 2D dam break problem after $t = 1.4$.

Figure 6.12: Snapshots of the 2D dam break problem at different times.

By comparing Figure 6.12 with the results from the book by Toro [3], and see that the numerical solution aligns well with the true solution from the book.

6.4 Scalability

To test the scalability of the FVM to solve the 2D SWE, we have run the 2D problem with a Gaussian initial condition, for different values of N, i.e., the number of cells in each direction. Numerical methods are good as they can be more or less as accurate as we want them to be, but the computational cost increases with the number of cells.

N	16	32	64	128	256
Time (s)	0.37	2.73	19.90	170.08	4205.45

Table 6.3: Running time for the FVM to solve 2D SWE for different values of N.

The run time dependent of the number of cells N is illustrated in Table 6.3.

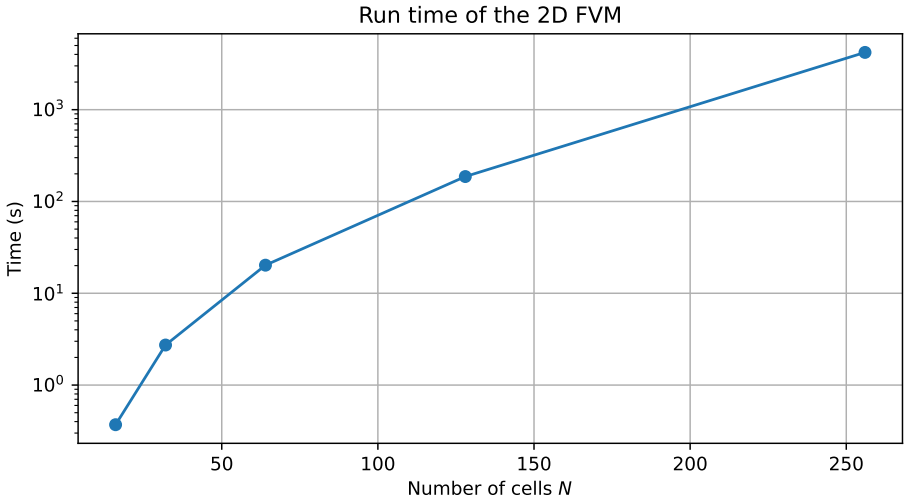


Figure 6.13: Scalability of the FVM to solve the 2D SWE.

From Table 6.3 and Figure 6.13 we see that the run time increases drastically with the number of cells N. This is expected, as the number of cells increases, the number of computations increases as well. This also means that the computational cost increases with the number of cells, and the method is not scalable for large values of N. Ultimately, if we want to model floods or tsunamis for the real world, we need a scalable method. This is also some of the motivation for using data-driven methods, as we will investigate if they can be more scalable.

Chapter 7

Data-driven results

In this section we present the results of the data-driven models. We go through two main cases. The first case is the 1D shallow water equations with Gaussian initial conditions, where we compare the performance of the CNN and FNO models. The second case is the 1D linearized shallow water equations in spherical coordinates, where we compare the performance of the CNN and FNO models. We also compare the performance of the models for different values of σ, i.e., the standard deviation of the Gaussian function. We do this to see how the models perform for different types of initial conditions, depending how smooth or discontinuous the solution is.

7.1 1D SWE with Gaussian initial conditions

We start by showing the numerical solution of the shallow water equations, then we present the predictions of the NN and FNO models. Until now, we have mostly considered discountinuous initial conditions, but we will also consider smooth initial conditions in this section. We solve the SWE with the following initial conditions:

$$\begin{aligned} h(x,0) &= h_0 \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \\ u(x,0) &= 0, \end{aligned} \tag{7.1.1}$$

where $h_0 = 1, \mu = 0.5, \sigma = 0.1$. The function $h(x,0)$ is a Gaussian function, illustrated in Figure 7.1.

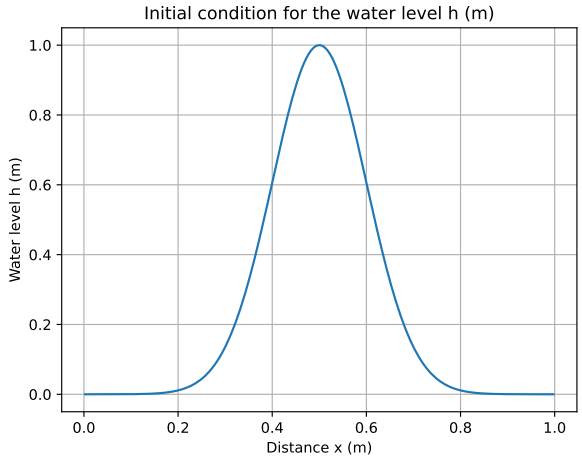


Figure 7.1: Initial conditions (7.1.1).

The domain is $x \in [0, 1]$ with $N = 200$ points and the final time is $t = 1.0$. We use a CFL number of 0.9 and variable time steps. The numerical solution is shown in Figure 7.2, in both a contour plot and a 3D plot.

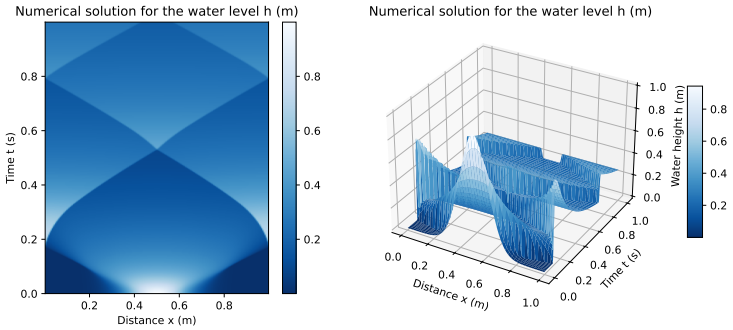


Figure 7.2: Numerical solution of the shallow water equations with initial conditions (7.1.1).

CNN Model

In the convolutional neural network, we train the model using the data generated by the numerical solution of the shallow water equations. The model uses the data from the numerical solution to predict the solution at the next time step. Meaning that the input and output data are the same, but shifted one time step. This way, the model is supposed to learn the flowmap. The model has been trained using the Adam optimizer with a learning rate of 0.001, a batch size of 32.

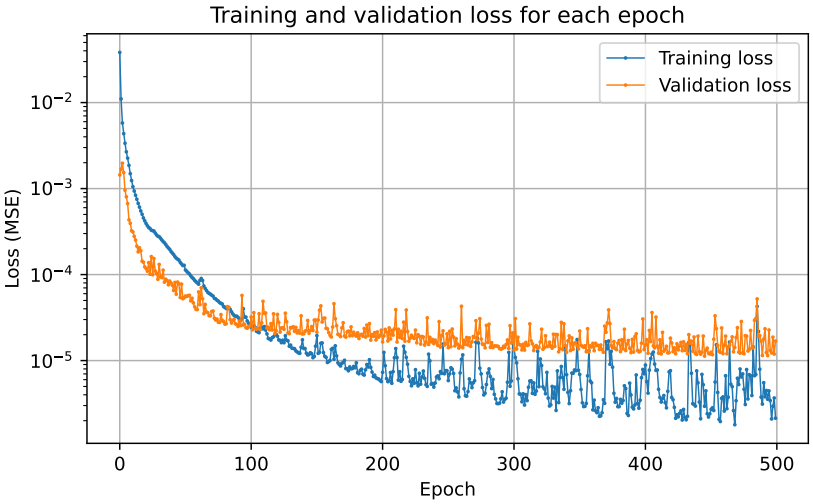


Figure 7.3: Training and validation loss for the CNN model.

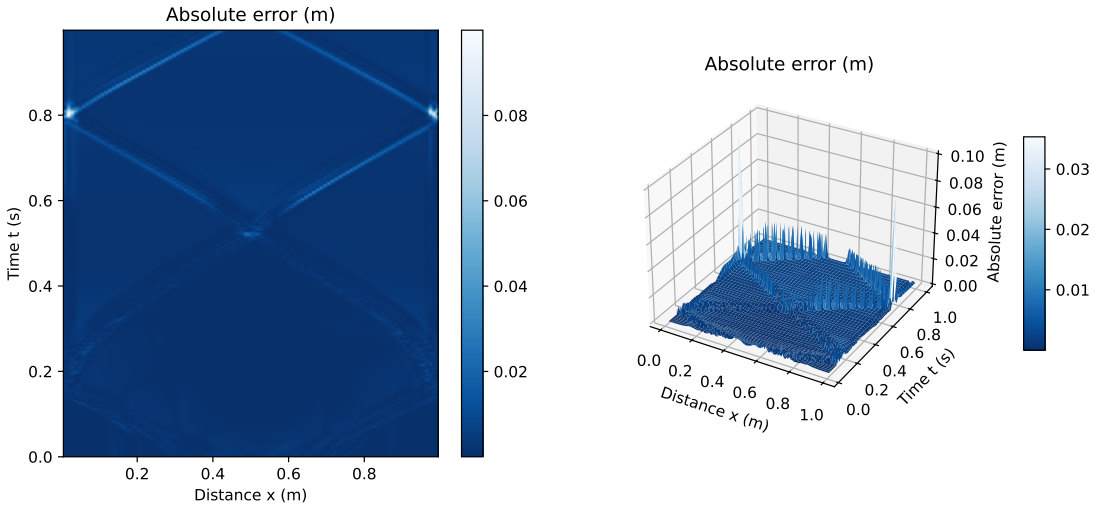


Figure 7.4: Error plot for the predictions for the CNN model.

To understand the performance of the model, we consider the predictions for some given time steps, shown in Figure 7.5.

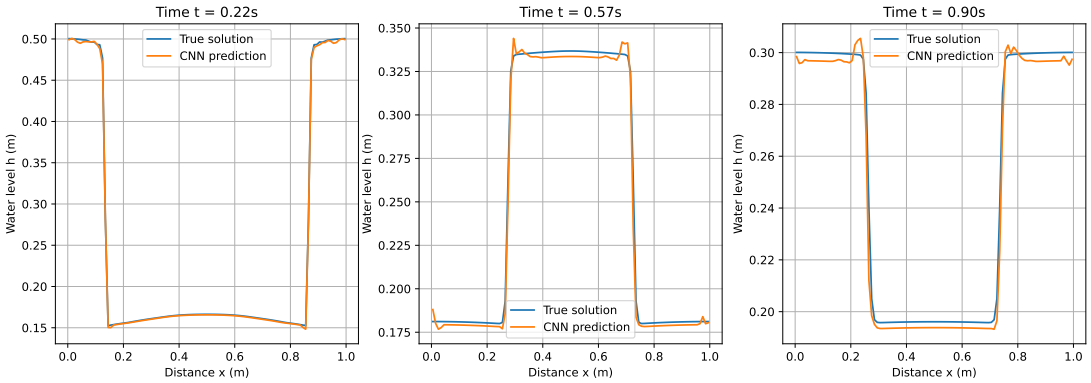


Figure 7.5: Predictions for the CNN model for some given time steps.

We also see that the highest absolute errors are located at the edges of the solution, which is expected, as the solution tends to be discontinuous. We also see that the further we get in time, the worse the predictions become. This is also somehow expected, since we are outside the training data.

Based on the training and validation loss, the model is overfitting the data. We see that as the training loss is decreasing, the validation loss is increasing. This also means that the model is not able to generalize the data. A possible solution could be to use more data.

Again, we see that the model finds some of the dynamics, but has many oscillations. We have also trained a FNN model and a LSTM model, but the results are not shown here, as the performance is worse than the RNN model.

FNO Model

One of the main goals in this thesis is to use Fourier Neural Operators to solve the shallow water equations. We define a FNO model, which consists of an input channel, 64 hidden channels and an output channel. We use a Fourier basis with 16 modes and a batch size of 32. The model is trained using the Adam optimizer with a learning rate of 0.001, a total of 1000 epochs and the critera is to minimize the mean squared error (MSE). The model is trained on the same data as the CNN model, and tested on the same data.

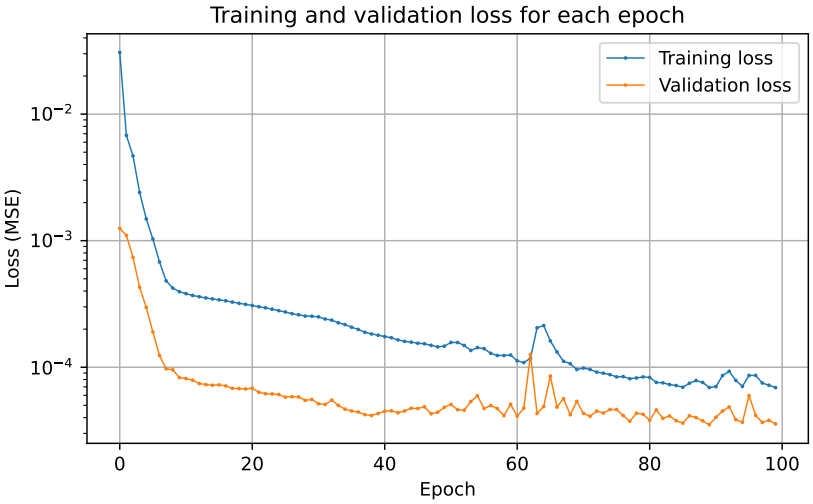


Figure 7.6: Training and validation loss for the FNO model.

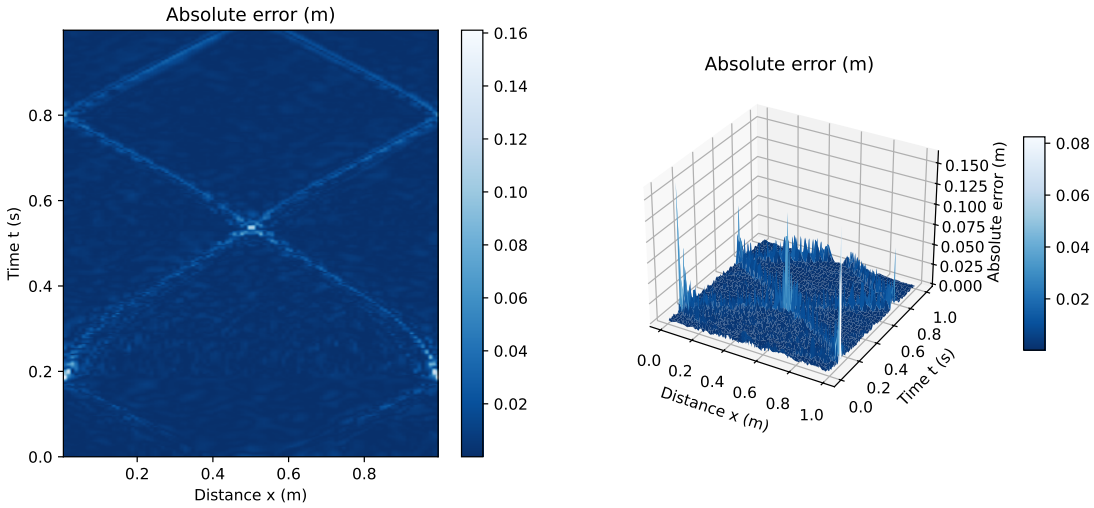


Figure 7.7: Error plot for the predictions for the FNO model.

To get an overview of the performance of the model, we consider the predictions for some given time steps, shown in Figure 7.8.

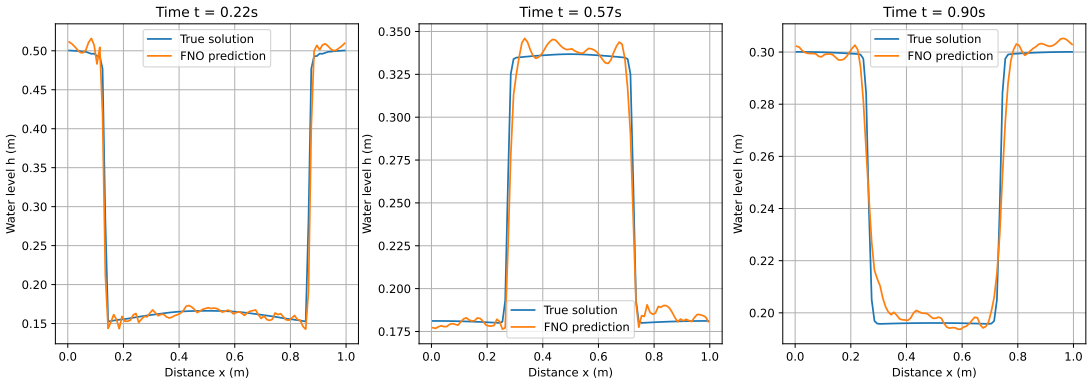


Figure 7.8: Predictions for the FNO model for some given time steps.

Comparison

Model	Gauss initial condition				New initial condition		
	Epochs	MSE	MAE	Training time (s)	MSE	MAE	Prediction time (s)
CNN	100	1.42e-05	1.30e-03	25.05	1.27e-05	8.88e-04	0.13
FNO	100	3.57e-05	5.18e-03	120.42	3.63e-05	5.11e-03	0.54

Table 7.1: Test loss in terms of MSE and MAE, and time for training the models for the 2D SWE.

Truncation errors

To consider the truncation error, we generate a close to exact solution by using a very fine gri, $N = 1000$. We plot the very accurate solution at time $t = 1$ together with the prediction of the CNN and FNO. This is to give us an idea of how the truncation error affects the predictions. This is one thing we must consider, when generating our own data. The difference between the exact solution and the predictions is shown in Figure 7.9.

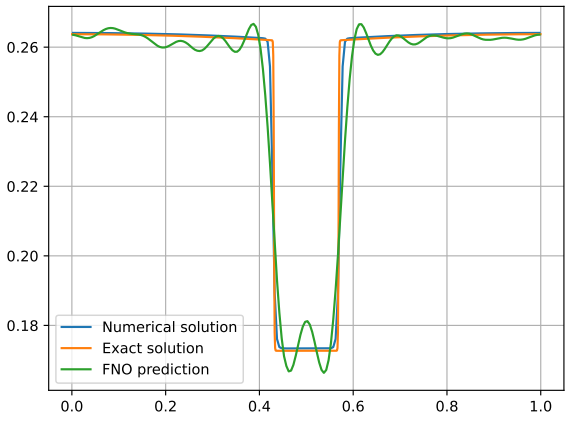


Figure 7.9: Truncation error for the FNO model.

7.2 1D linearized SWE in Spherical Coordinates

We also consider the spherical shallow water equations in a 1D setting, focusing on the linearized SWE on a circular domain. The length of the domain corresponds to the circumreference of the circle, $L = 2\pi$, and is discretized into $N = 500$ points. The initial conditions is specified as a Gaussian function wrapped around the circle, expressed as:

$$h(\theta, 0) = h_0 \exp\left(\frac{-(\theta - \mu)^2}{2\sigma^2}\right),$$

where the parameters are $h_0 = 1, \mu = \frac{\pi}{4}, \sigma = \frac{\pi}{16}$. The initial conditions can be seen in Figure 7.10.

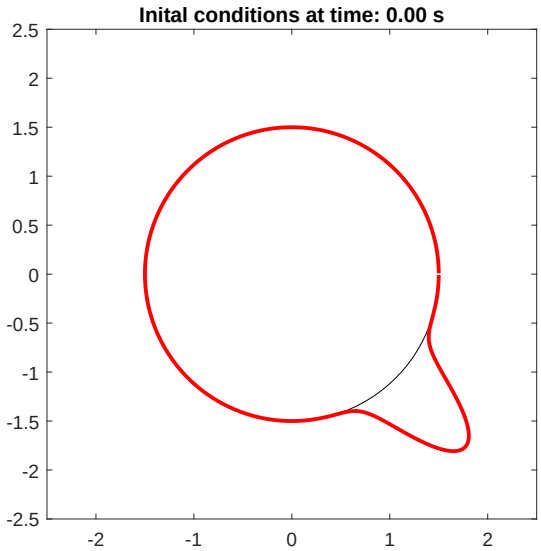


Figure 7.10: Initial conditions for the 1D linearized shallow water equations in spherical coordinates.

The numerical solution in the θ, t plane is shown in Figure 7.11, for $t = 0$ to $t = 1$.

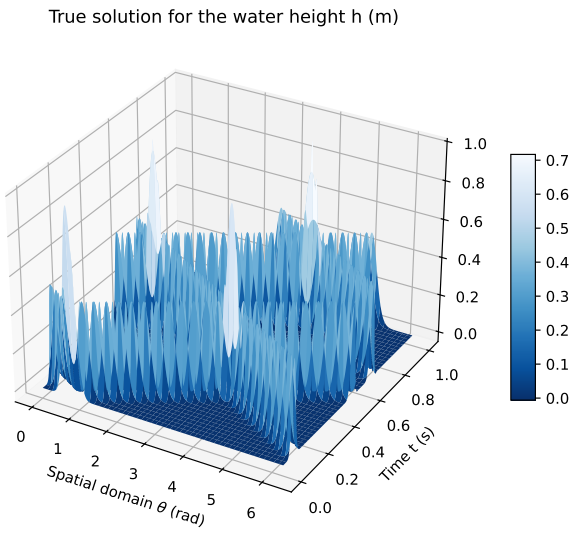


Figure 7.11: Numerical solution of the spherical shallow water equations in 1D in the $\theta, t-$space.

Based on the data generated by the numerical solution (we use time steps of $\Delta t = 0.0025$), we train a FNO model and a Convolutional Neural Network (CNN) model.

CNN Model

We have also trained a CNN model on the data generated by the numerical solution of the SWE in 1D. Like the FNO model, the CNN model is trained on the data from $t = 0$ to $t = 0.6$, validated on the data from $t = 0.6$ to $t = 0.8$, and tested on the data from $t = 0.8$ to $t = 1.0$. The training and validation loss is shown in Figure 7.12.

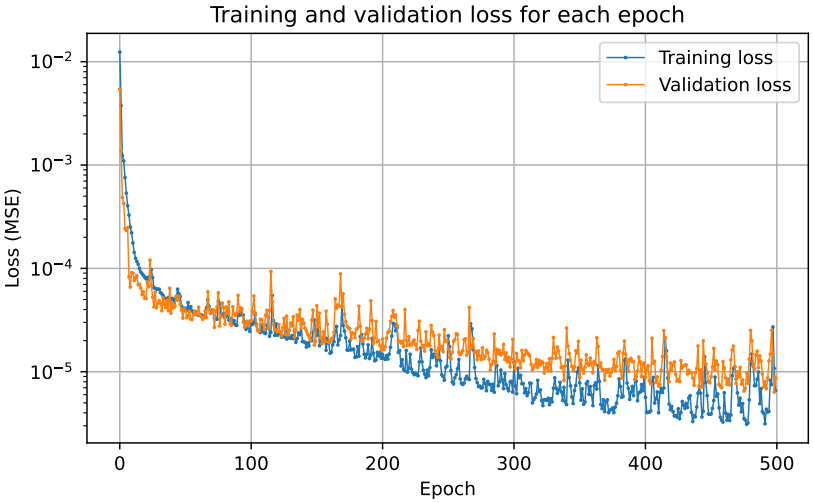


Figure 7.12: Training and validation loss for the CNN model for the spherical shallow water equations in 1D.

The error plots are shown in Figure 7.13.

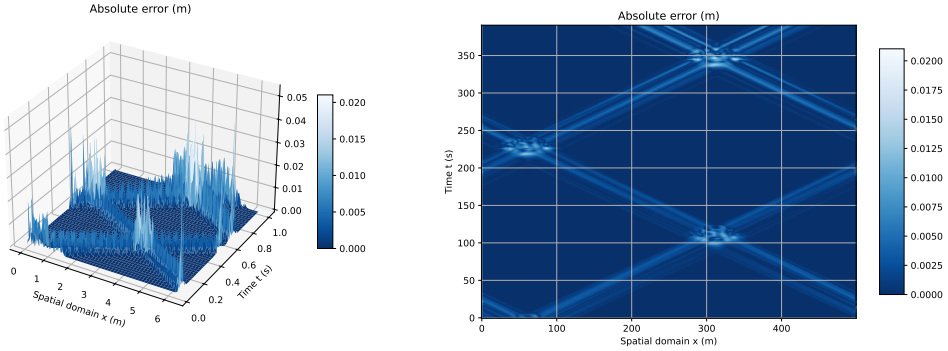


Figure 7.13: Error plots for the predictions of the CNN model for solving the 1D linearized spherical SWE.

The predictions for some given time steps are shown in Figure 7.14.

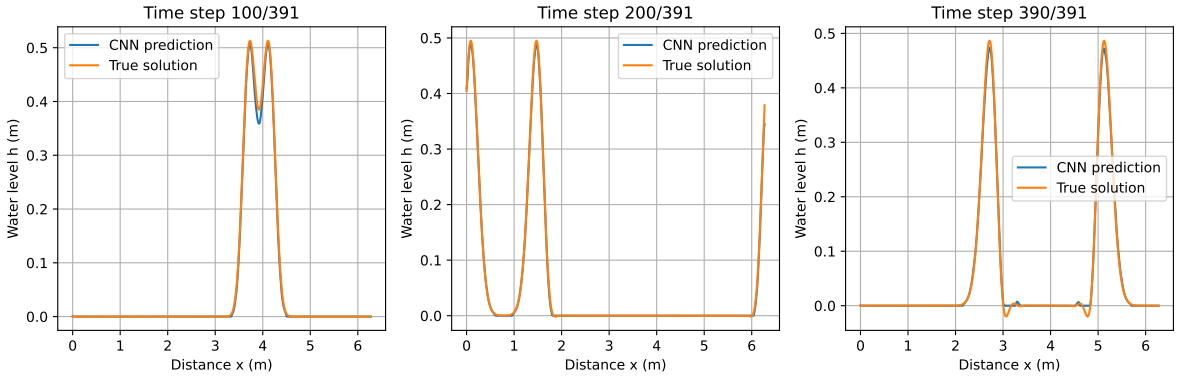


Figure 7.14: Predictions for the spherical shallow water equations in 1D using the CNN model for some given time steps.

From Figure 7.14, we also see that the predictions capture the waves, but are more noisy than the FNO predictions.

FNO model

The FNO model consists of an input channel, 64 hidden channels and an output channel. We use a Fourier basis with 16 modes and a batch size of 32. The model is trained using the Adam optimizer with a learning rate of 0.001, a total of 100 epochs and the critera is to minimize the mean squared error (MSE). The model is trained on the data from $t = 0$ to $t = 0.6$, validated on the data from $t = 0.6$ to $t = 0.8$, and tested on the data from $t = 0.8$ to $t = 1.0$. The training and validation loss is shown in Figure 7.15.

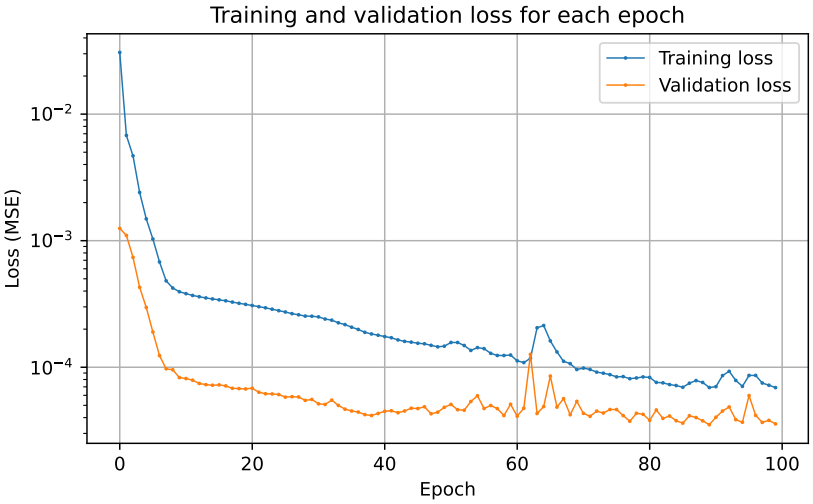


Figure 7.15: Training and validation loss for the FNO model for the spherical shallow water equations in 1D.

The error plots are shown in Figure 7.16.

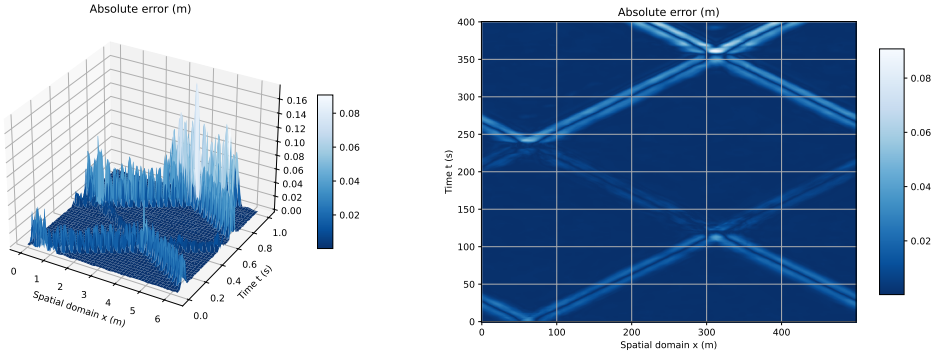


Figure 7.16: Error plots for the predictions of the 1D linearized spherical SWE.

From Figure 7.16 we see that the model is able to learn the dynamics of the solution. We also see that the absolute error is biggest at the edges of the solution, which is expected, as the solution tends to be discontinuous.

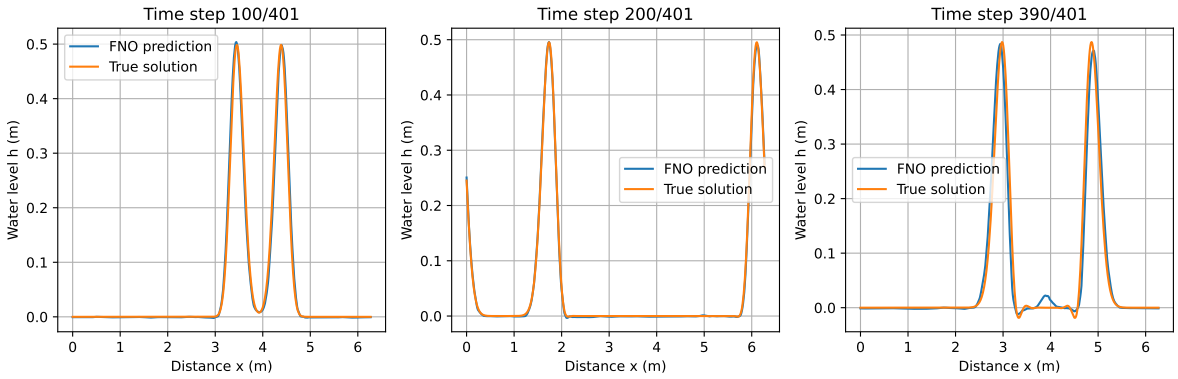


Figure 7.17: Predictions for the spherical shallow water equations in 1D.

From Figure 7.17 we see that the FNO predictions are smooth and rather accurate, but have some small errors in the top of the waves.

Comparison

To get an overview of the performance of the different models, we consider the MSE for the predictions for the 1D SWE spherical case.

Model	$\sigma = \pi/8$			$\sigma = \pi/16$			$\sigma = \pi/32$		
	MSE	MAE	Time (s)	MSE	MAE	Time (s)	MSE	MAE	Time (s)
CNN	1.31e-02	3.34e-02	91.33	2.32e-05	1.95e-03	112.70	5.92e-03	2.85e-02	8.48
FNO	2.05e-04	8.23e-03	108.79	5.78e-04	1.04e-02	564.82	5.19e-04	1.16e-02	125.80

Table 7.2: Test loss in terms of MSE and MAE, and time for training the models for the 1D spherical SWE.

From Table 7.2 we see that the CNN model is slightly faster and better than the FNO model for $\sigma = \pi/8$ and $\sigma = \pi/16$, but for $\sigma = \pi/32$, the performance of the CNN model is decreasing. Probably due to the fact that the smaller the σ, the more discontinuous the solution is, and the FNO model is better at capturing the discontinuities. We see that the MAE in general is higher than the MSE, and is also increasing for smaller σ. Additionally, we observe that the MAE is higher, as it places more weight on small errors compared to the MSE. And in the CNN case we see a lot of small errors/noise, which is why the MAE is higher than the MSE.

7.3 Data-driven results for the 2D SWE

In this section we will present the results for the 2D SWE using the data-driven models. We consider the same initial condition (Gaussian function) as in the 1D case, but now in two dimensions. We solve the 2d SWE using both a CNN and a FNO model, and compare the results in terms of run time and accuracy. We also compare the run time to the numerical method FVM, to see if the data-driven models can be used as a faster alternative to the FVM. We also test the ability of the FNO model to generalize to a finer grid, by training the model on a coarse grid and then making predictions on a fine grid. Finally, we will test the FNO model's ability to generalize further in time and make long-term predictions. It is also in this chapter, we will present our proof of concept for the 2D SWE.

7.4 2D SWE with initial Gaussian function

The initial condition for the 2D problem is a Gaussian function with a standard deviation of 0.1 and a mean of 0.5. The initial condition can be seen in Figure 7.18.

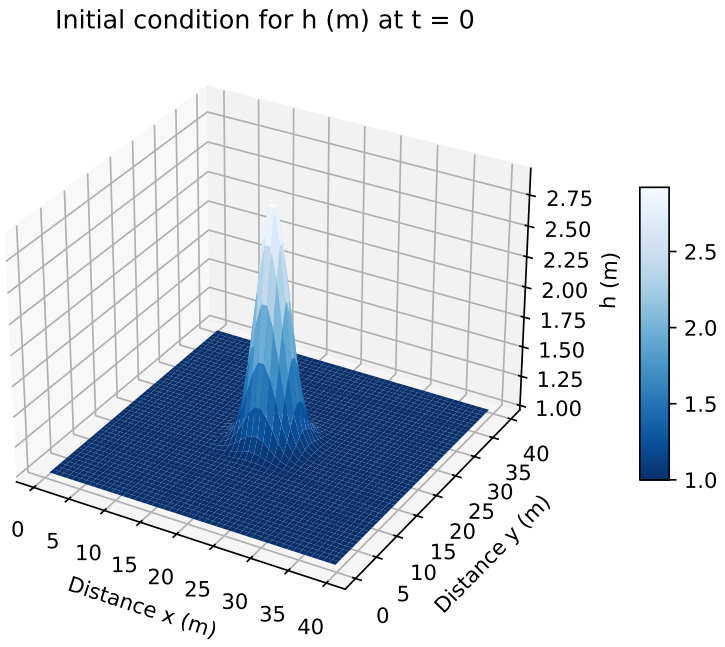


Figure 7.18: Initial condition for the 2D problem.

CNN Model

The training and validation loss for the 2D CNN model can be seen in Figure 7.19.

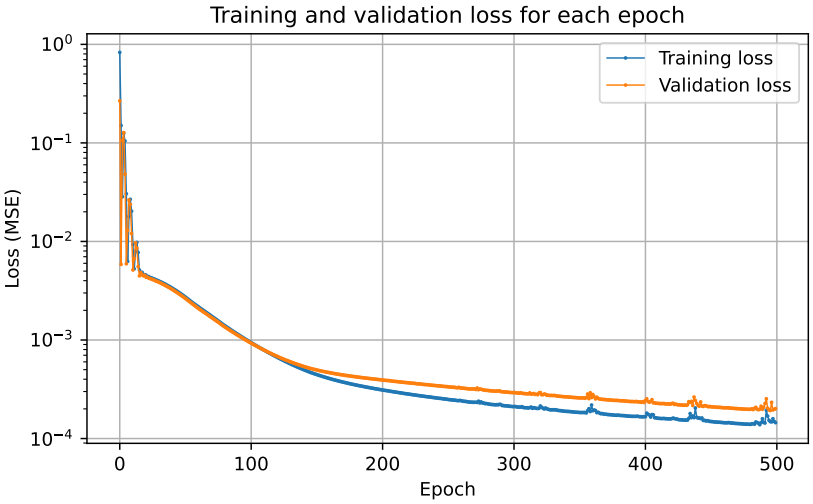


Figure 7.19: Training and validation loss for the 2D CNN model.

The error plot for the last prediction for the 2D CNN can be seen in Figure 7.20.

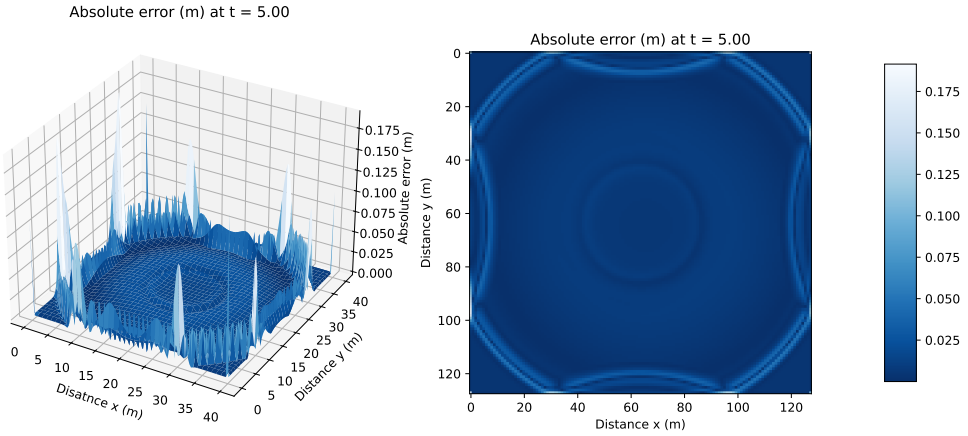


Figure 7.20: Error plot for the last prediction for the 2D CNN.

FNO Model

The training and validation loss for the 2D FNO model can be seen in Figure 7.21.

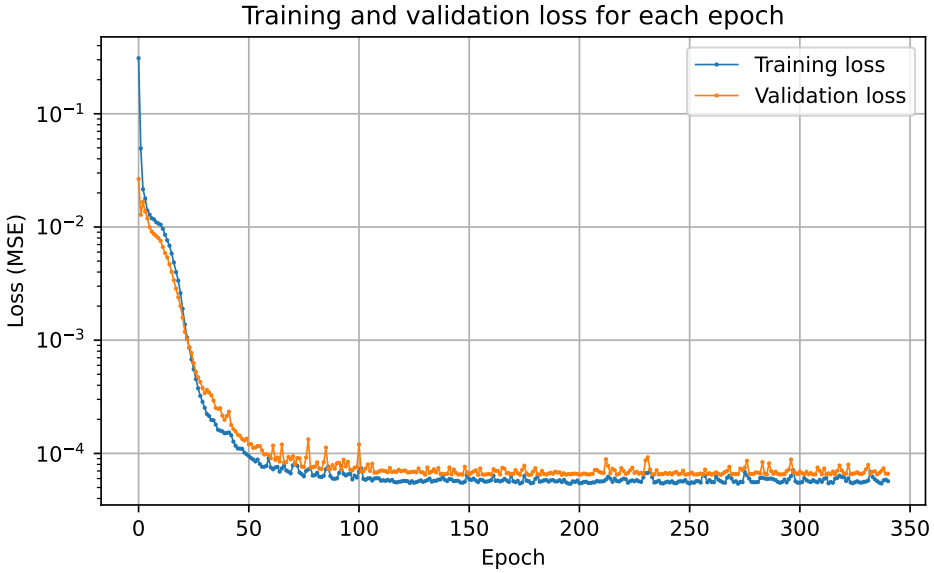


Figure 7.21: Training and validation loss for the 2D FNO model.

The error plot for the last prediction for the 2D FNO can be seen in Figure 7.22.

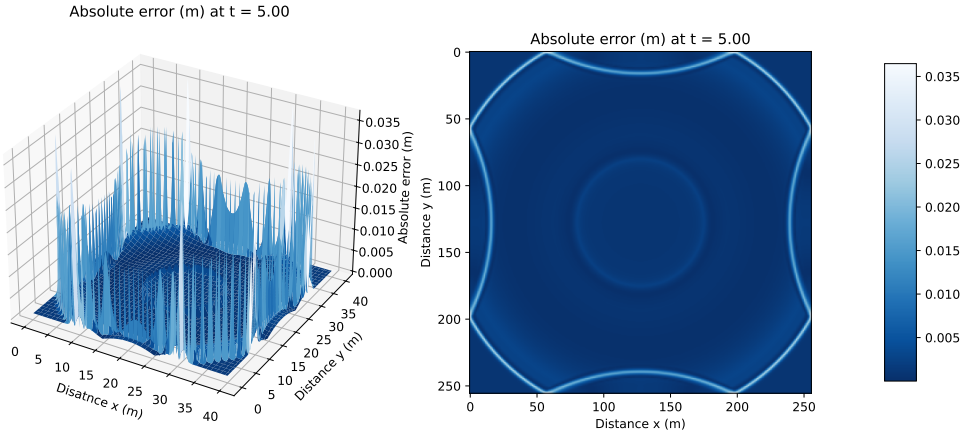


Figure 7.22: Error plot for the last prediction for the 2D FNO.

Comparison

To get an overview of the performance of the different models, we consider the MSE and MAE for the predictions for the 2D SWE case.

Model	$N = 64$				$N = 128$			
	Epochs	MSE	MAE	Time (s)	Epochs	MSE	MAE	Time (s)
CNN	500	4.84e-04	1.17e-02	67.43	500	2.42e-04	8.72e-03	704.70
FNO	500	1.44e-04	7.68e-03	509.06	500	6.30e-05	4.44e-03	704.70

Table 7.3: Test loss in terms of MSE and MAE, and time for training the models for the 2D SWE.

We see that the FNO model in general needs fewer epochs to converge compared to the CNN model, but it also takes longer time to train. From the theory we know that FNO are supposed to work when we are training on a coarse grid and then make predictions on a fine grid. To test this, we will train the models on a coarse grid and then make predictions on a fine grid. The table below shows the results when the FNO-model is trained on a grid with $N = 64$ and then makes predictions on a grid with $N = 128$.

Model	$N = 128$				
	Epochs	MSE	MAE	Training time (s)	Prediction time (s)
FNO	100	1.35e-04	6.26e-03	92.85	5.57
CNN	100	1.58e-02	1.58e-02	13.83	1.01

Table 7.4: Test loss in terms of MSE and MAE, and time for training the FNO model on a grid with $N = 64$ and then making predictions on a grid with $N = 128$.

From the results in Table 7.4 we see that the FNO model is able to generalize to a finer grid. This way, it is possible to train the model on a coarse grid and then make predictions on a fine grid, which is a great advantage when solving the SWE numerically, as it is very computationally expensive to solve the SWE on a fine grid using the FVM.

Model	$N = 256$				
	Epochs	MSE	MAE	Training time (s)	Prediction time (s)
FNO	1	3.24e-01	5.58e-01	1.30	112.01
CNN	1	2.51e-02	2.51e-02	12.03	14.98

Table 7.5: Test loss in terms of MSE and MAE, and time for training the FNO model on a grid with $N = 64$ and then making predictions on a grid with $N = 256$.

We will also time the predictions to compare the speed of the models. We can also compare to the run time of the numerical method FVM, see Table 6.3. Handle the scalability isssues.

Long-term predictions

We are also very interested in testing the ability of the FNO to generalize further in time. We will therefore train the model on a time interval $[0, 5]$ and make predictions for $t = X$. Our first approach was to make a prediction, and based on that prediction make a new prediction for the next time step. This way we can make predictions for a longer time interval. For the first time steps the results were fine, but as we made predictions further in time, the error increased, probably because the error from the previous time step was carried over to the next time step, i.e., the error was accumulated. So, the next approach was to make sequences of the data and then train the model to predict the next time step based on the sequence. This way, the newest prediction becomes a part of a sequence and thus, do not have the same influence on the next prediction as in the previous approach. The hope is that this approach makes long-term predictions more stable. We make predictions for up to $n = 20$ time steps in the future.

For this case we generate data with a constant time step size $\Delta t = 0.025$, where the other data was generated with a variable time step size, to ensure satisfaction of the CFL condition. To ensure stability, we have to choose the time step size small enough. What we did was, we investigated the time step size for the generated data. To create Δt we halfed the smallest time step size, and this is how we got $\Delta t = 0.025$.

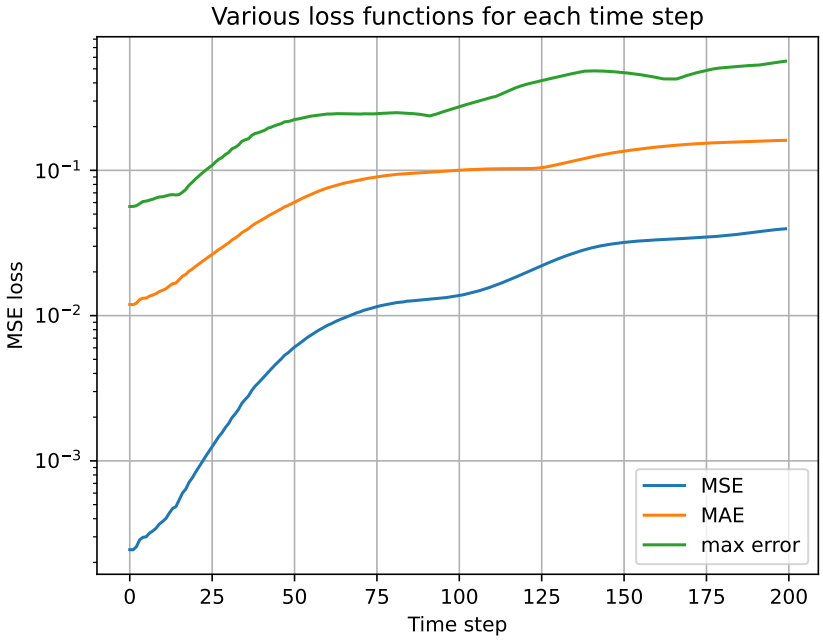


Figure 7.23: MSE and MAE for the long-term predictions for the 2D FNO model.

From Figure 7.23 we see that the MSE and MAE are increasing as we make predictions further in time. The error plot for the prediction for $n = 20$ time steps in term of maximum absolute error can be seen in Figure 7.24. The corresponding plot for $n = 40$ time steps can be seen in Figure 10.14 in Appendix.

The error plot for the long-term predictions can be seen in Figure 7.24.

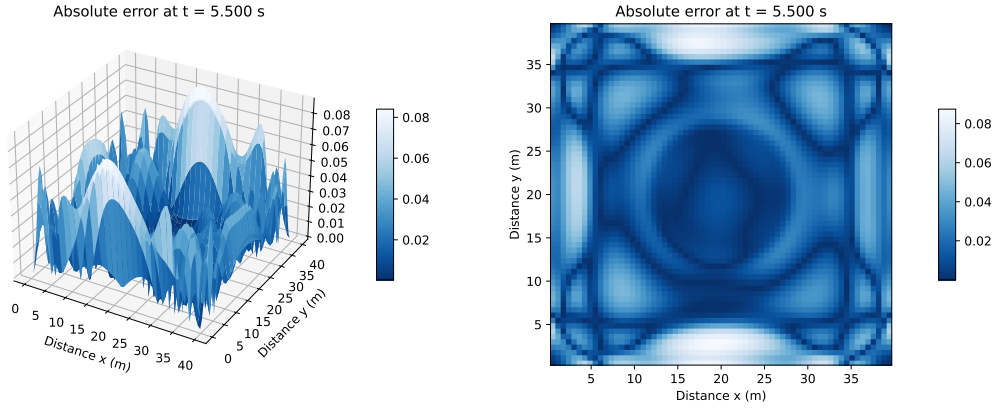


Figure 7.24: Error plot for the long-term prediction for the 2D FNO model.

From the error plot in Figure 7.24, we see that the error is only slightly bigger than for the short-term predictions in Figure 7.22. This indicates that the FNO model is able to generalize further in time and make long-term predictions. The higher maximum absolute error for $n = 40$ is probably due to accumulating errors. The lower MSE and MAE do not neccessarily mean that the predictions are better, but can also be due to the fact that the

water is in a stable state for $n = 40$ time steps in the future, and thus, the predictions are more stable. Therefore, when making predictions it is important to consider the physical properties of the system, and not only the loss functions. The different ways to calculate error, can also be considered depending on the goal of the predictions, i.e., if it is important to minimize the MSE or the maximum absolute error, depending on the physical properties of the system. Is it better to have one large errror or many small ones.

7.5 2D spherical SWE

Chapter 8

Discussion

In this project, we have implemented a Finite Volume Method to solve the shallow water equations in 1D, succesfully simulating the dam break problem.

Generating data from the numerical model, how much accuracy do we lose? Truncation errors.

Chapter 9

Conclusion

- By training the FNO on a coarse grid and evalue on a fine grid, the run time is XX times faster, while maintaining the same accuracy.
- Long-term prediction: the FNO can predict the solution for a longer time than the training time. However, I do not have good results for training on coarse grid, predicting for a fine grid for long-term prediction.

FNO's ability to generalize to unseen data.

9.1 Further work

In this project, we have implemented a Finite Volume Method to solve the shallow water equations in 1D, succesfully simulating the dam break problem. Although we attempted to extend this method to 2D, time constraints prevented us from completing it. We also began adapting the 1D method to accommodate a curved bottom surface instead of a flat one but couldn't finish this implementation either. Both extensions are promising, and it would be interesting to observe how the model performs with a curved bottom. Another interesting extension is to implement the FVM in 3D and use it to solve the SWE in spherical coordinates.

Bibliography

[1] E.F. Toro. *Shock-Capturing Methods for Free-Surface Shallow Flows*. Oxford University Press, 2001.

[2] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 2009.

[3] E.F. Toro. *Computational Algorithms for Shallow Warer Equations*. Springer, 2024.

[4] Ilya Peshkov. Advanced numerical methods for environmental modeling. https://www.unitn.it/dricam/1116/advanced-numerical-methods-environmental-modeling, 2023. Lecture notes.

[5] Manuel J. Castro, Sergio Ortega, and Carlos Parés. Well-balanced methods for the shallow water equations in spherical coordinates. https://www.sciencedirect.com/science/article/pii/S0045793017303237, 2017.

[6] Alex Bihlo and Roman O. Popovych. Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, 456:111024, 2022.

[7] David J. Raymond. Shallow water on a sphere. http://kestrel.nmt.edu/~raymond/classes/ph332/notes/sphere/sphere.pdf, New Mexico Tech. Accessed 2024.

[8] L. Gavete, B. Alonso, M.L. Gavete, F. Ureña, and J.J. Benito. An adaptive solver for the spherical shallow water equations. *Mathematics and Computers in Simulation*, 79(12):3466–3477, 2009. The International Conference on Approximation Methods and numerical Modeling in Environment and Natural Resources.

[9] Joseph Galewsky, Richard K. Scott, and Lorenzo M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, Jan 2004.

[10] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. https://arxiv.org/pdf/2010.08895, 2021.

[11] Boris Bonev, Christian Hundt, Thorsten Kurth Jaideep Pathak, et al. Modeling earth's atmosphere with spherical fourier neural operators. https://developer.nvidia.com/blog/modeling-earths-atmosphere-with-spherical-fourier-neural-operators/, 2023.

[12] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. https://arxiv.org/abs/2306.03838, 2023.

[13] C.B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. Kluwer Academic Publishers, 1994.

[14] Wikipedia. Leibniz integral rule. https://en.wikipedia.org/wiki/Leibniz_integral_rule, 7/10-2024.

[15] Adrian E. Gill. *Atmosphere-Ocean Dynamics*. Academic Press, 1982.

[16] Longitude and latitude map with degrees. https://animalia-life.club/qa/pictures/longitude-and-latitude-map-with-degrees, Accessed: 2024.

[17] Wikipedia. Coriolis force. https://en.wikipedia.org/wiki/Coriolis_force, 14/11-2024.

[18] Boris Bonev, Jan S. Hesthaven, Francis X. Giraldo, and Michal A. Kopera. Discontinuous galerkin scheme for the spherical shallow water equations with applications to tsunami modeling and prediction. *Journal of Computational Physics*, 362:425–448, 2018.

[19] Claes Eskilsson and Spencer J. Sherwin. Discontinuous galerkin spectral/hp element modelling of dispersive shallow water systems. *Journal of Scientific Computing*, 22:269–288, 2005.

[20] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[21] Wikipedia. Gaussian function. https://en.wikipedia.org/wiki/Gaussian_function, 14/10-2024.

[22] A. P Engsig-Karup and J. S. Hesthaven. An introduction to discontinuous galerkin methods for solving partial differential equations. https://www2.compute.dtu.dk/~apek/DGFEMCourse2009/, 2009. Lecture notes.

Chapter 10

Appendix

The code can be found at: `https://github.com/MelissaJessen/Shallow-Water-Equations`.

10.1 Additional figures from 1D SWE spherical CNN and FNO

Results for the other sigma values.

CNN, $\sigma = \frac{\pi}{8}$.

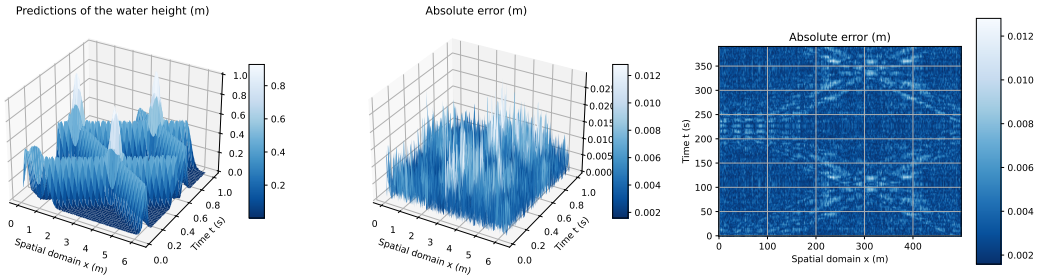


Figure 10.1: Error for the 1D SWE spherical CNN with sigma=1.

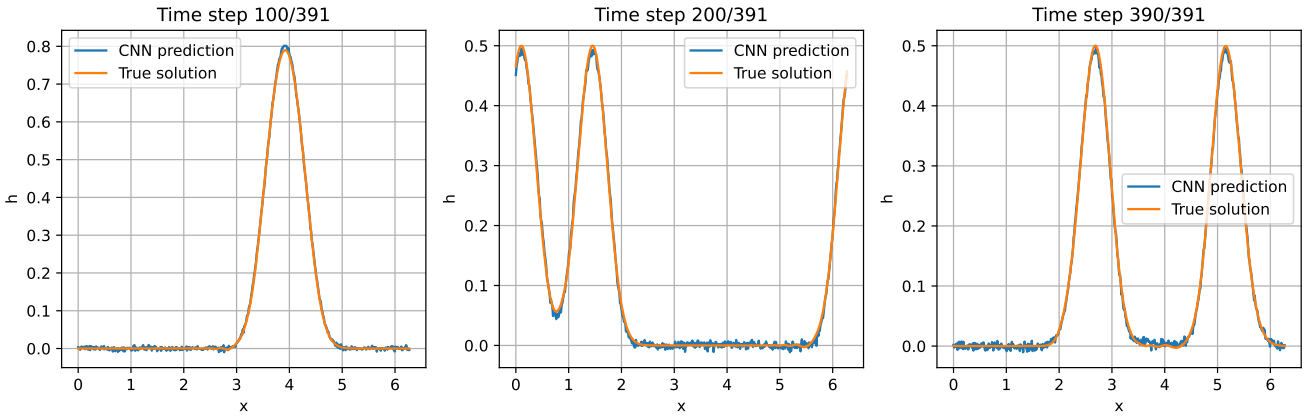


Figure 10.2: Results for the 1D SWE spherical CNN with sigma=1.

CNN $\sigma = \frac{\pi}{32}$.

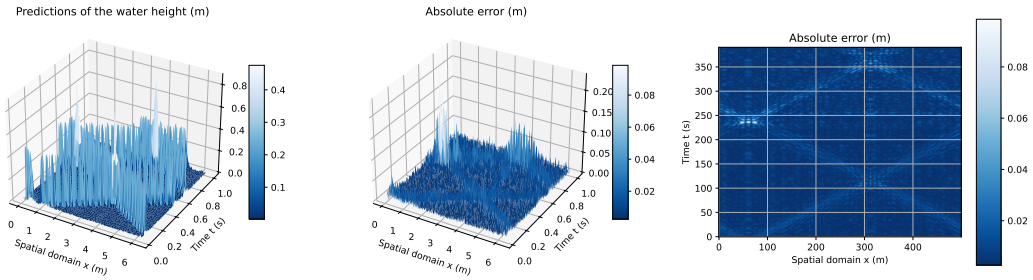


Figure 10.3: Error for the 1D SWE spherical CNN with sigma=1.

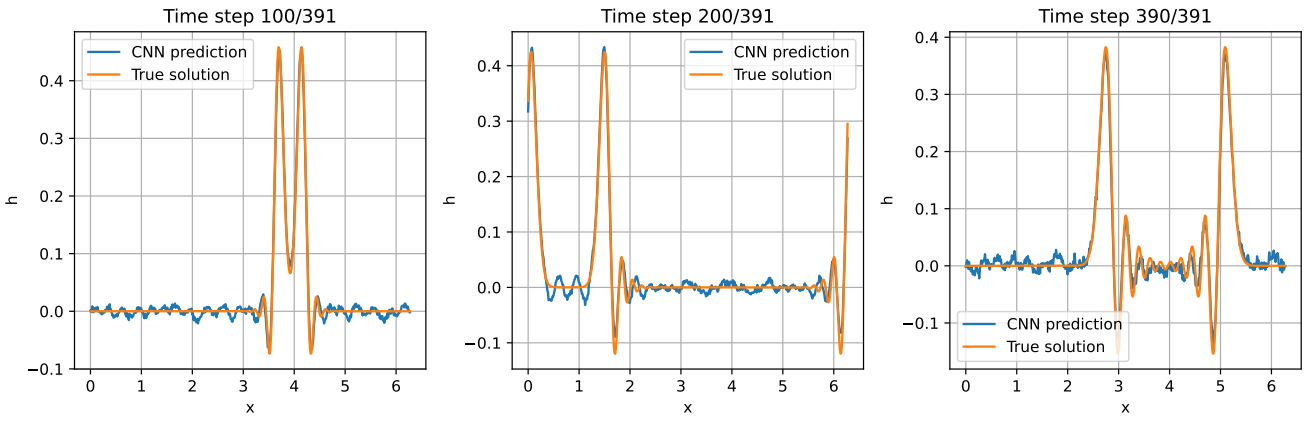


Figure 10.4: Results for the 1D SWE spherical CNN with sigma=1.

FNO, $\sigma = \frac{\pi}{8}$.

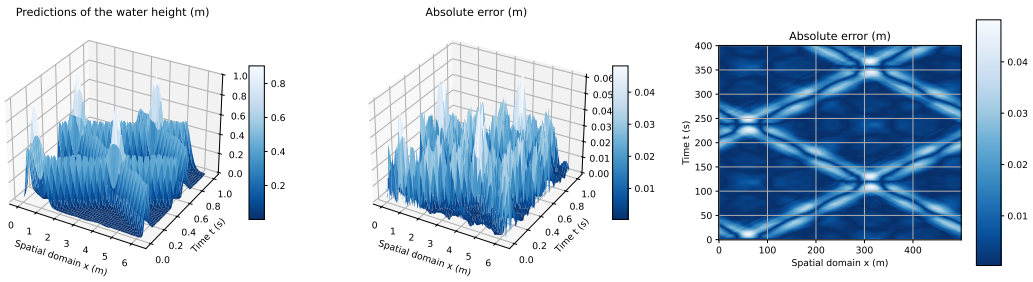


Figure 10.5: Error for the 1D SWE spherical CNN with sigma=1.

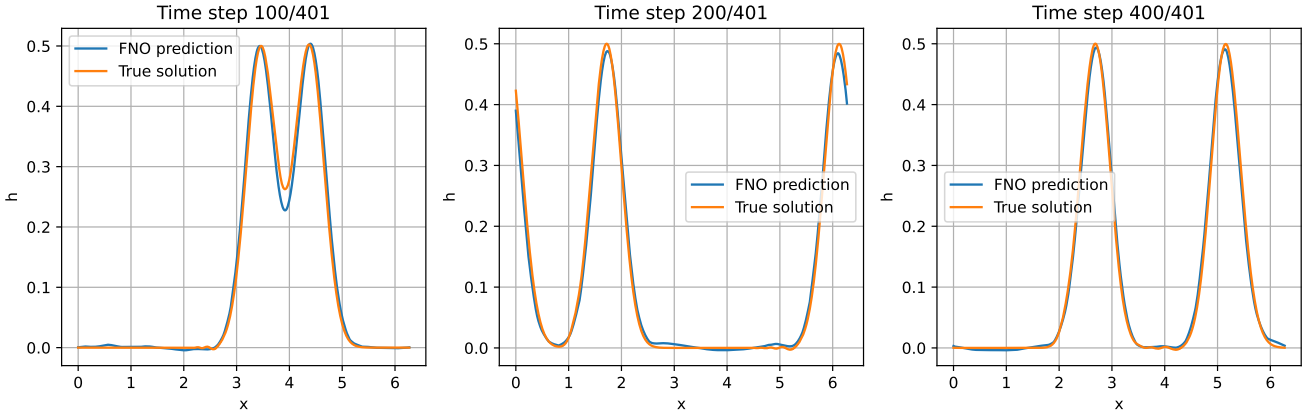


Figure 10.6: Results for the 1D SWE spherical CNN with sigma=1.

FNO $\sigma = \frac{\pi}{32}$.

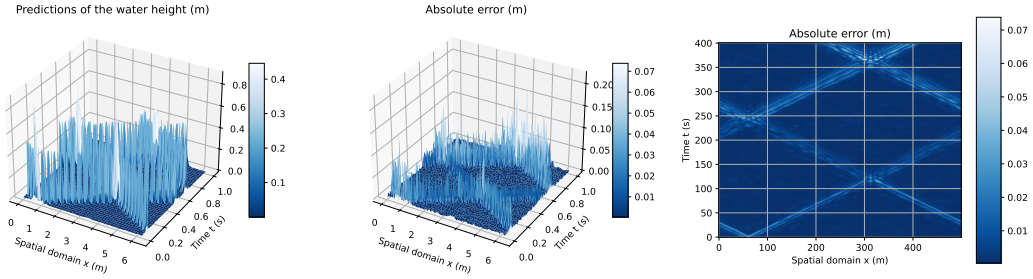


Figure 10.7: Error for the 1D SWE spherical CNN with sigma=1.

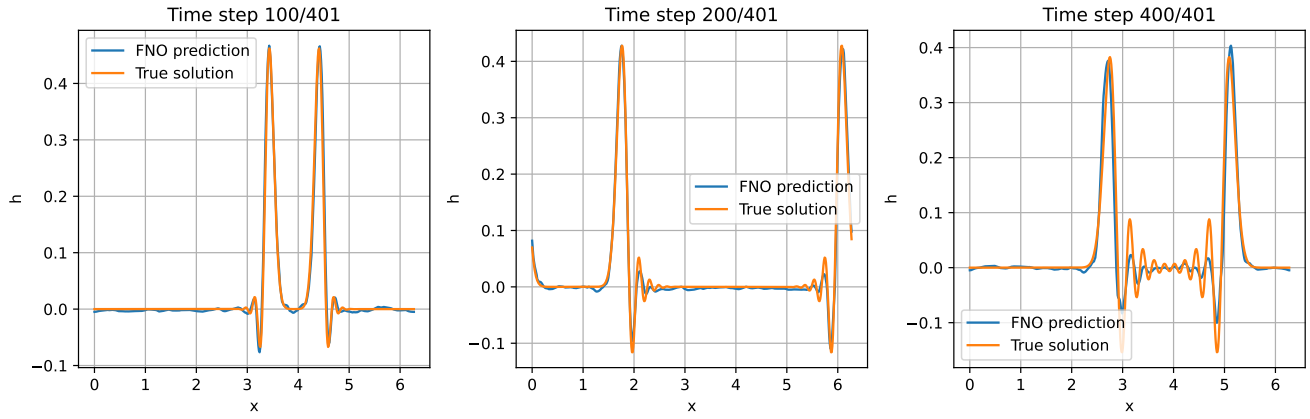


Figure 10.8: Results for the 1D SWE spherical CNN with sigma=1.

10.2 FNO Toro test 1

To test the FNO model on a more challenging problem, we consider the Toro test case 1. We use the same model as before, but we train it on the data from the Toro test case 1. Again, we use the first 80% of the data for training and the last 20% for testing. The results of the model are shown in Figure 10.10 and **??**.

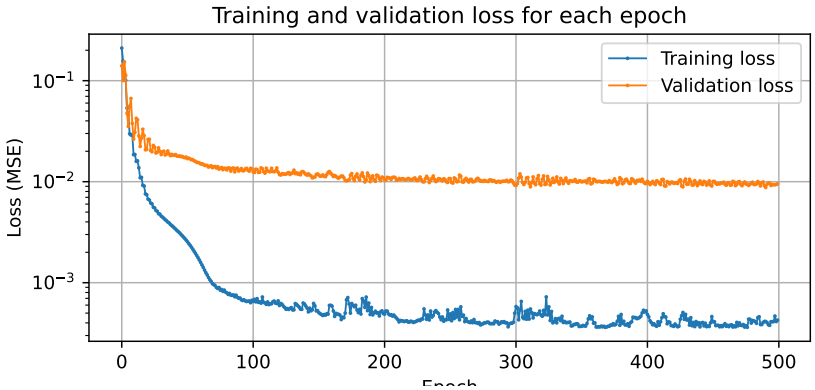


Figure 10.9: FNO Toro test 1 loss.

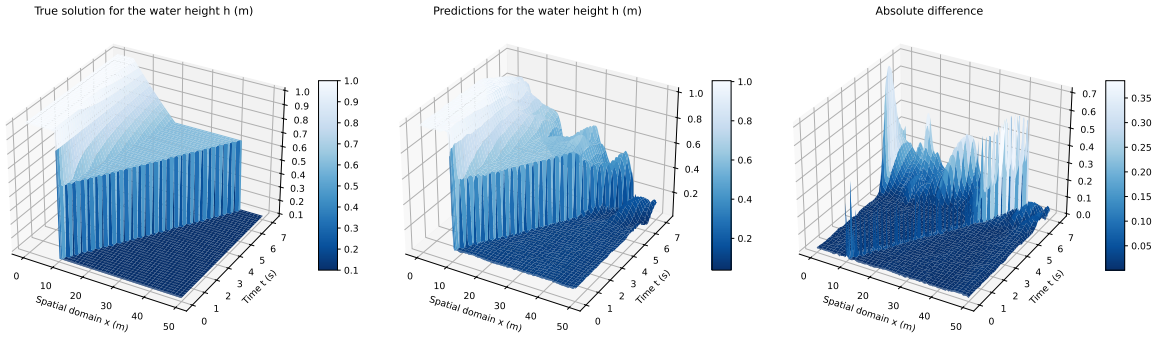


Figure 10.10: FNO Toro test 1 predictions in 3d.

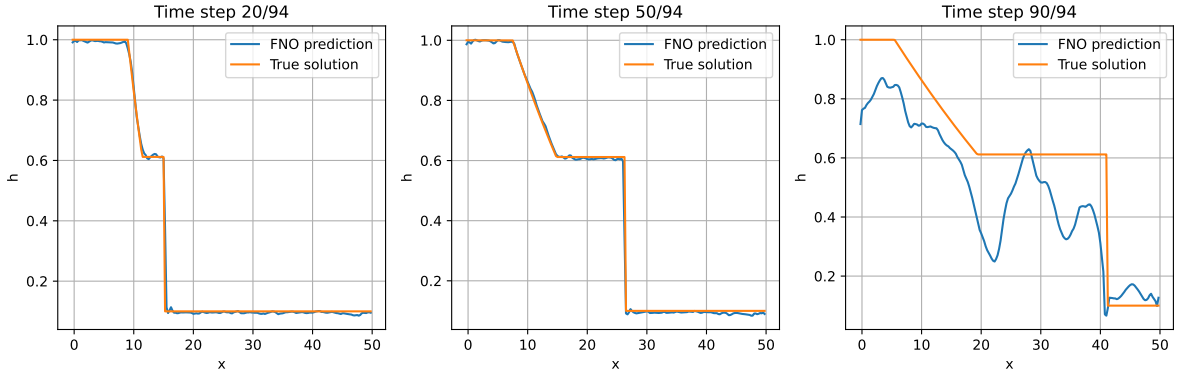


Figure 10.11: FNO Toro test 1 predictions timesteps.

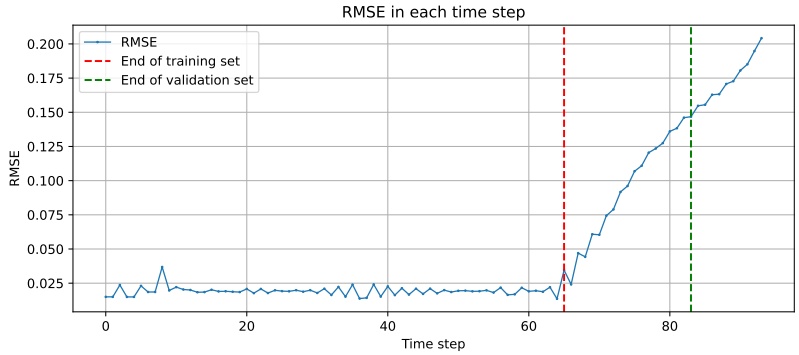


Figure 10.12: FNO Toro test 1 predictions RMSE.

10.3 2D SWE long term prediction

The results for the long-term predictions can be seen in Figure 10.13.

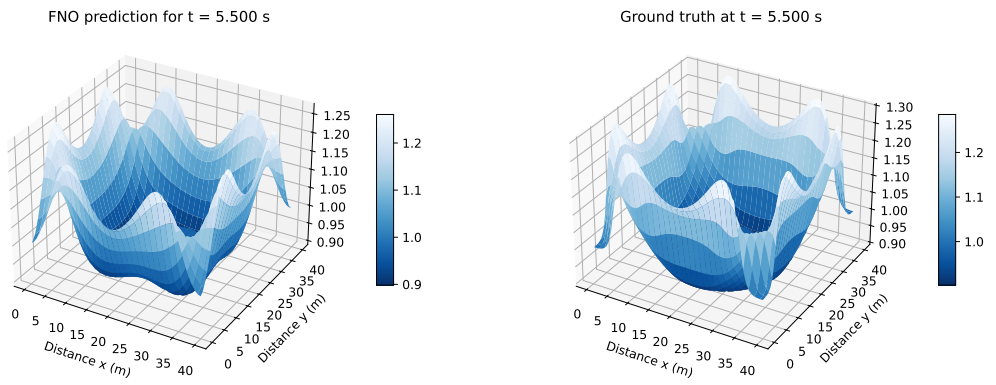


Figure 10.13: Long-term prediction for the 2D FNO model.

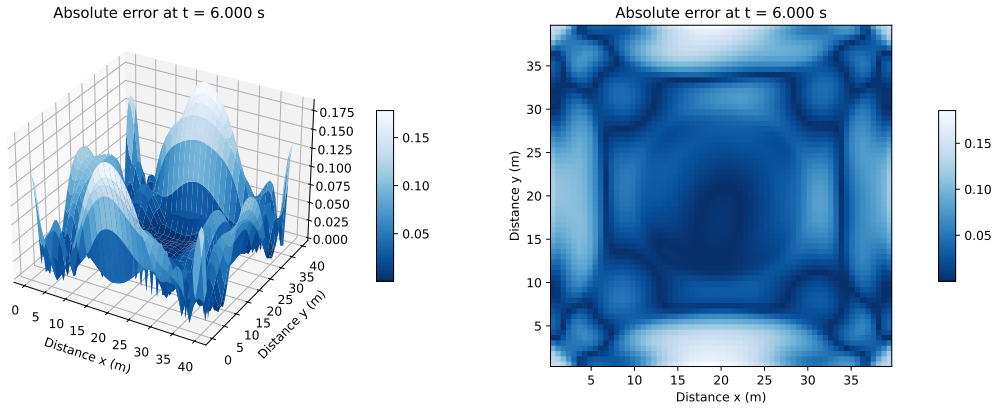


Figure 10.14: Error plot for the long-term prediction for the 2D FNO model.