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

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

Melissa Ulsøe Jessen (s194322)

November 5, 2024

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.

Notes

40 years ago Lax and Wendroff [116] proved mathematically that conservative numerical methods, if convergent, do converge to correct solutions of the equations. More recently, Hou and LeFloch [91] proved a complementary theorem which says that if a non-conservative method is used, then the wrong solution will be comptedm if this contains a shock wave.

Spurious oscillations are unavoidable if one uses linear methods of accuracy greater than one.

Gudonovs theorem: all (linear) schemes of accuracy greater than one will provide spurious oscillations. One must use non-linear methods, even when applied to linear problems.

A succesful new class of shock-capturing numerical methods are the so-called high-resolution methods. High-resolution methods = Oscillation-free near shock waves, and retain second-order accuracy in smooth parts of the flow.

Gudonov methods are shock-capturing upwind methods. He later proposed a very fast exact Riemann solver and also approximate Riemann solvers.

a: the generalisation of Gudonov's first-order method to second-order accuracy by van Leer. b: the development of new approximate Riemann solvers by Roe and others.

TVD = Total Variation Diminishing.

More advanced, not presented in this book are UNO, ENO and WENO methods. Also the discountinuous Galerkin finite element method, combining FEM and Gudonov theory.

Trento course: - Day 2: SWE is a non-linear scalar equation/ conservation law

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 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].

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* [5], 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 Spherical Fourier Neural Operators (SFNO) to generate weather forecasts around the globe [6].

1.3 Thesis overview

Chapter 2

Theory

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. We will also present the Finite Volume Method (FVM) including the Riemann problem. Lastly, we will dive into the theory behind the data-driven methods.

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 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:

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

We use the following notation for partial derivatives:

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

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.,

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}.$$

Establishing this relevant notation, we can now derive the shallow water equations in 1D and 2D.

2.2 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 [7].

2.2.1 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 (see 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

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

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

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

2.2.2 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:

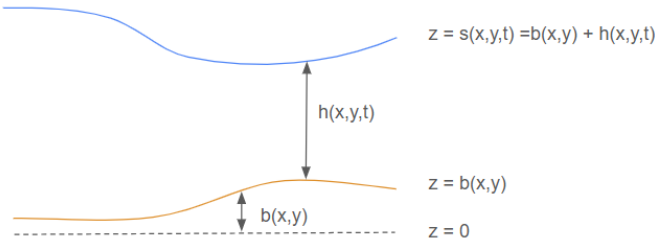


Figure 2.1: Illustration of a water column with a free surface.

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 (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.

2.2.3 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

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

These are the simplified momentum equations for the shallow water equations. 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.

2.2.4 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 $x-$ and $y-$momentum conservation equations in (2.2.7), 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 [8], 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

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

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.

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

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

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = \mathbf{S}(\mathbf{U}), \tag{2.2.27}$$

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:
In this project, we will begin by considering the homogeneous one-dimensional case of the shallow water equations. In the homogeneous case we assume that $\mathbf{S}(\mathbf{U}) = 0$. The vector form of the SWE in 1D is then given by

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

where

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

In 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

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}), \tag{2.2.28}$$

where

$$\mathbf{U} = \begin{bmatrix} h \\ hu \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} \quad \text{and} \quad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ -ghb_x \end{bmatrix}. \tag{2.2.29}$$

If $\mathbf{S}(\mathbf{U}) = 0$, the equation (2.2.28) is called a conservation law, and otherwise it is called a balance law.

2.2.6 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 (2.2.28) and (2.2.29) allows discountinuous solutions. The integral form is obtained by integrating the vector form (2.2.28) 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.

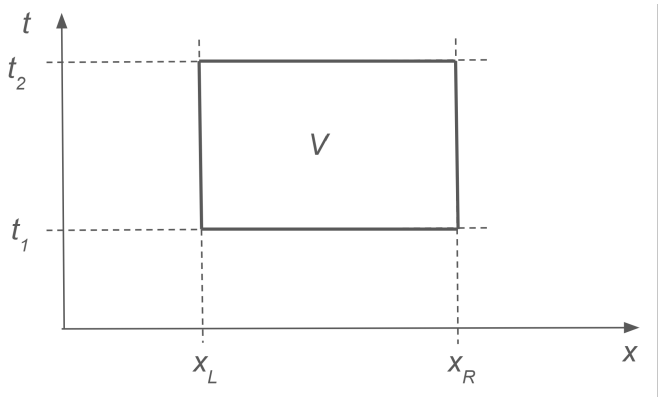


Figure 2.2: Illustration of a control volume V in the $x-t$ plane. Illustration modified from [3].

First we integrate the vector form of the SWE (2.2.28) over x from x_L to x_R to obtain

$$\int_{x_L}^{x_R} \mathbf{U}_t \, \mathrm{d}x + \int_{x_L}^{x_R} \mathbf{F}(\mathbf{U})_x \, \mathrm{d}x = \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.2.30}$$

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 (2.2.30):

$$\int_{x_L}^{x_R} \mathbf{U}_t \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)) + \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.2.31}$$

Then we integrate (2.2.31) 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

$$\int_{x_L}^{x_R} \mathbf{U}(x, t_2) \, \mathrm{d}x = \int_{x_L}^{x_R} \mathbf{U}(x, t_1) \, \mathrm{d}x + \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_L, t)) \, \mathrm{d}t - \int_{t_1}^{t_2} \mathbf{F}(\mathbf{U}(x_R, t)) \, \mathrm{d}t + \int_{t_1}^{t_2} \int_{x_1}^{x_2} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x\mathrm{d}t, \tag{2.2.32}$$

which is the integral form of the conservation laws for the SWE in 1D. An alternative integral form of (2.2.28) is stated in [3]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{x_L}^{x_R} \mathbf{U}(x, t) \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)) + \int_{x_L}^{x_R} \mathbf{S}(\mathbf{U}) \, \mathrm{d}x. \tag{2.2.33}$$

From (2.2.33) we get that the integral form of the homogeneous SWE in 1D is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{x_L}^{x_R} \mathbf{U}(x, t) \, \mathrm{d}x = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_R, t)), \tag{2.2.34}$$

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

2.2.7 SWE in Spherical Coordinates

TBD.

2.3 Finite Volume Method for the Shallow Water Equations

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 [9].

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.

2.3.1 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 is illustrated in Figure 2.3.

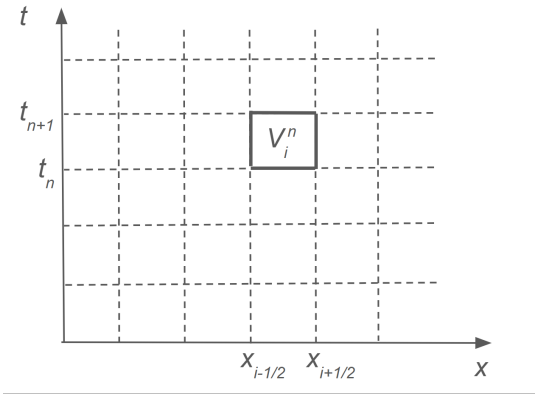


Figure 2.3: Illustration of the control volume V_i^n in the x, t plane.

For now, we will assume a uniform grid for simplicity. The Finite Volume Formula is derived from the integral

form (2.2.32). By dividing this 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

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n\right) + \Delta t \mathbf{S}_i. \tag{2.3.1}$$

The formula (2.3.1) 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:

$$\mathbf{U}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t_n)\ \mathrm{d}x, \tag{2.3.2}$$

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

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

and correspondingly for $\mathbf{F}_{i+1/2}^n$:

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

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 2.4.

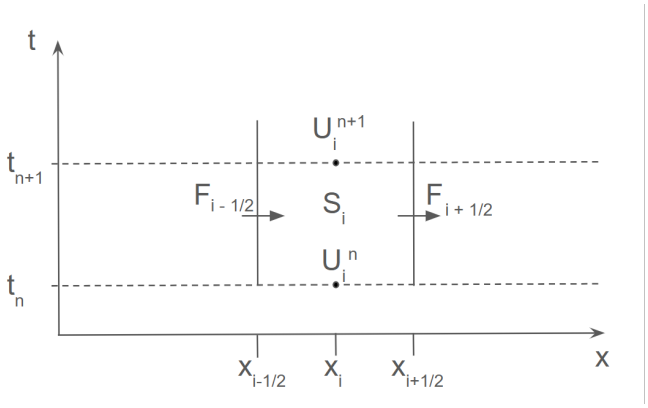


Figure 2.4: Illustration of the grid for the 1D SWE.

The central idea of the FVM is to define the numerical flux $\mathbf{F}_{i+1/2}^n$, at the cell interface, as a function of the cell averages $\mathbf{U}_i^n$ and $\mathbf{U}_{i+1}^n$, 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, $\mathbf{U}_i^n$, 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, 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 Finite Difference Method (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.

2.3.2 Finite Volume Method for the 2D SWE

We now extend the FVM to two space dimensions. Consider the 2D SWE in vector form (2.2.27) with $\mathbf{S}(\mathbf{U}) = 0$:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = 0. \tag{2.3.3}$$

Following the methods outlined in [2], an explicit finite volume scheme to solve (2.3.3) is given by

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^n + \frac{\Delta t}{\Delta x}(\mathbf{F}_{i-1/2,j} - \mathbf{F}_{i+1/2,j}) + \frac{\Delta t}{\Delta y}(\mathbf{G}_{i,j-1/2} - \mathbf{G}_{i,j+1/2}). \tag{2.3.4}$$

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

2.4 The Riemann problem

We will now define the Riemann problem, since it plays a crucial role in the finite volume method. In the Riemann problem we distinguish between what we call a wet bed and a dry bed. A wet bed is the case where the water depth is positive everywhere, whereas a dry bed is the case where the water depth is zero in some cells. The special Riemann problem where parts of the bed are dry is dealing with the so-called dry fronts or wet/dry fronts, which are challenging to handle numerically. We will leave these cases for now, and only consider wet bed problems. The Riemann problem for the shallow water equations in 1D with a zero source term is defined as the initial-value problem (IVP) [3]:

$$\begin{aligned} &\text{PDEs: } \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \\ &\text{ICs: } \mathbf{U}(x,0) = \begin{cases} \mathbf{U_L}, & \text{if } x < 0, \\ \mathbf{U_R}, & \text{if } x > 0. \end{cases} \end{aligned} \tag{2.4.1}$$

Here, the vectors $\mathbf{U}$ and $\mathbf{F}(\mathbf{U})$ in (2.4.1) are given by

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ hvu \end{bmatrix}, \tag{2.4.2}$$

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

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

which represents the conditions at time $t = 0$ 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 both exactly and approximately. There are several approximate Riemann solvers, such as the HLL and Roe solvers, which are based on the approximate solution of the Riemann problem. We will consider some of these solvers later in the thesis.

2.4.1 The Dam-Break problem

We now introduce the dam-break problem, a scenario of significant physical interest. This problem models the sudden release of water following the collapse of a dam, making it highly relevant for studying natural disasters such as floods and tsunamis. As a classic test case for numerical methods, the dam-break problem is commonly used to test the ability of a method to capture discontinuities in the solution. The dam-break problem is a special case of the Riemann problem (2.4.1). The difference is that in the dam-break problem, the initial velocity components, u_L, u_R, v_L and v_R, are zero, whereas in the Riemann problem they are allowed to be distinct from zero. The initial setup is visualized in Figure 2.5.

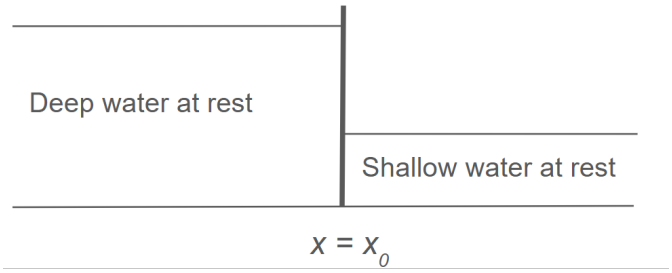


Figure 2.5: Initial conditions for the dam-break problem. An infinitely thin wall at $x = x_0$ divides two sections of water with different water levels.

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 $t = 0$.

2.4.2 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 (2.4.1), which consists of a combination of waves, including shock waves and rarefaction waves. In the solution of the Riemann problem (2.4.1) 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 (2.4.1) 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. In the example of the dam-break problem, with inital conditions as in Figure 2.5, the solution consists of a left rarefaction wave and a right shock wave.

2.4.3 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 (2.4.1). 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.

2.5 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 is equivalent to multiplication in the Fourier domain.

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.

2.5.1 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 [5].

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

$$v_{t+1}(x) := \sigma\left(W v_t(x) + \left(\mathcal{K}(a;\phi)v_t\right)(x)\right), \quad \forall x \in D \tag{2.5.1}$$

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

$$\left(\mathcal{K}(\phi)v_t\right)(x) = \mathcal{F}^{-1}\left(R_\phi \cdot (\mathcal{F}v_t)\right)(x), \quad \forall x \in D \tag{2.5.2}$$

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 2.6.

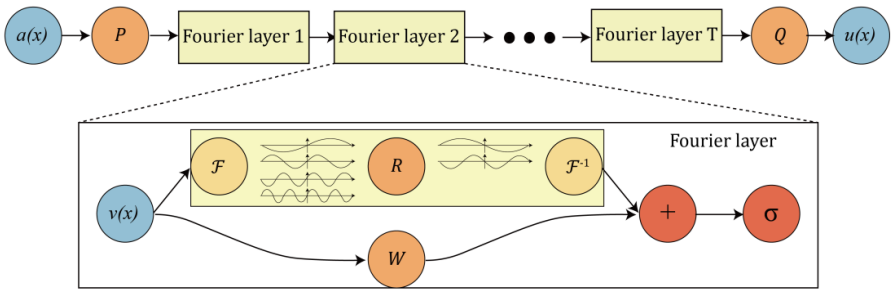


Figure 2.6: An overview of the network architecture with several Fourier layers. Illustration from [5].

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 $mathcalF$, 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.

Chapter 3

Methodology

In this chapter we introduce the numerical fluxes, used to solve the approximate Riemann problem. We will also present the numerical methods used in this thesis to solve the SWE in 1D and 2D. Finally, we will outline how the data needed for the data-driven methods is generated.

3.1 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 (2.4.1) to find the numerical flux. There are several numerical fluxes that can be used to solve the local Riemann problem, and we will consider some of them in this section.

3.1.1 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]. Godunov's method is a fundamental starting point. In the method we solve the non-linear Riemann problem at each cell interface.

Consider the initial-boundary value problem (IBVP) for a system of N nonlinear hyperbolic conservation (balance?) laws

$$\begin{cases} \text{PDEs:} & \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}), \quad x \in [a,b], \quad t > 0, \\ \text{ICs:} & \mathbf{U}(x,0) = \mathbf{U}^{(0)}(x), \quad x \in [a,b], \\ \text{BCs:} & \mathbf{U}(a,t) = \mathbf{B}_L(t), \quad \mathbf{U}(b,t) = \mathbf{B}_R(t), \quad t \geq 0. \end{cases} \tag{3.1.1}$$

The vectors $\mathbf{B}_L(t)$ and $\mathbf{B}_R(t)$ denote the boundary conditions at the left and right boundaries, respectively. The Godunov Upwind method in conservative form (2.3.1) solves the IBVP (3.1.1). We compute h_* and u_* by starting with a two-rarefaction wave structure, and solve the wetbed problem iteratively.

3.1.2 HLL solver

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

$$\mathbf{F}_{1+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } S_L \geq 0, \\ \mathbf{F}^{HLL} \equiv \frac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_L S_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & \text{if } S_L \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R \leq 0. \end{cases} \tag{3.1.2}$$

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

$$S_L = u_L - a_L q_L, \quad S_R = u_R + a_R q_R,$$

where $q_K(K = L, R)$ 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

$$h_* = \frac{1}{g}\left(\frac{1}{2}(a_L + a_R) + \frac{1}{4}(u_L - u_R)\right)^2, \tag{3.1.3}$$

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.

3.1.3 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

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

An estimate for the middle wave speed S^* can be calculated as

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

3.1.4 Rusanov

To obtain the next flux, we assume an estimate S^+ for the positive wave speed is available. Then we set

$$S_L = -S^+, \quad S_R = S^+. \tag{3.1.4}$$

By substituting (3.1.4) into the $\mathbf{F}^{HLL}$ in (3.1.2), we obtain the Rusanov flux as

$$\mathbf{F}^{Rus}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) - \frac{1}{2}S^+\left(\mathbf{U}_R - \mathbf{U}_L\right), \tag{3.1.5}$$

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

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

This sheme is upwind and based on a one-wave model. Therefore it is an incomplete Riemann solver.

3.1.5 Lax-Friedrichs

The Lax-Friedrichs method is a centred method, which is first-order accurate. We set the wave speed S^+ as the largest possible speed, while still ensuring stability, i.e.,

$$S^+ = \frac{\Delta x}{\Delta t}. \tag{3.1.6}$$

By inserting the wave speed (3.1.6) into the Rusanov flux, 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 $\mathbf{F}_R = \mathbf{F}(\mathbf{U}_R)$.

3.1.6 Lax-Wendroff

The Lax-Wendroff method is a centred method, which is second-order accurate in space and time. There are several versions of the Lax-Wendroff flux, but in this thesis we will use the following flux:

$$\begin{aligned} \mathbf{U}^{LW}_{i+\frac{1}{2}} &= \frac{1}{2}\left(\mathbf{U}_L + \mathbf{U}_R\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{F}_R - \mathbf{F}_L\right), \\ \mathbf{F}^{LW}_{i+\frac{1}{2}} &= \mathbf{F}(\mathbf{U})^{LW}_{i+\frac{1}{2}}. \end{aligned} \tag{3.1.7}$$

3.1.7 FORCE

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

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

It is possible to extend the FORCE scheme to multiple dimensions on structured meshes by using dimensional splitting. The FORCE scheme is first-order accurate.

3.1.8 Flux-splitting/Upwind

In the flux splitting we compute h_* and u_* by assuming a two-rarefaction wave structure. We define

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

to obtain

$$h_{S2} = \left(\frac{0.75}{\sqrt{g}} \cdot (q_L - q_R) + 0.5 \cdot \left(h_L^{1.5} + h_R^{1.5} \right) \right)^2,$$
$$h_S = h_{S2}^{\frac{1}{3}}$$

and

$$u_* = \frac{1}{2}(u_L + u_R) + \frac{1}{3}\sqrt{g}(h_L^{1.5} - h_R^{1.5}).$$

3.1.9 Roe solver

Consider the non-linear Riemann problem in (2.4.1):

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

where $\mathbf{A}$ is the Jacobian matrix of $\mathbf{F}$. The Roe solver is based on an approximation of the Jacobian matrix $\mathbf{A}$ by a Roe matrix $\tilde{\mathbf{A}}$, which is a constant coefficient matrix, to obtain the linear system

$$\mathbf{U}_t + \tilde{\mathbf{A}}\mathbf{U}_x = 0.$$

This means that the Roe solver solves the approximated Riemann problem

$$\mathbf{U}_t + \tilde{\mathbf{A}}\mathbf{U}_x = 0.$$
$$\mathbb{U}(x,0) = \begin{cases} \mathbf{U}_L, & x < 0, \\ \mathbf{U}_R, & x > 0. \end{cases}$$

exact. That is, the original non-linear conservation laws are replaced by a linearised system with constant coefficients.

The main idea in the Roe solver is to find average values $\tilde{h}, \tilde{a}, \tilde{u}$ and $\tilde{\psi}$ for the depth h, the celerity a (??), the velocity component u and the scalar ψ. The method thus use the following Roe averages:

$$\begin{cases} \tilde{h} = \sqrt{h_L h_R}, \\ \tilde{u} = \dfrac{u_L\sqrt{h_L} + u_R\sqrt{h_R}}{\sqrt{h_L} + \sqrt{h_R}}, \\ \tilde{a} = \sqrt{\dfrac{1}{2}(a_L^2 + a_R^2)}, \\ \tilde{\psi} = \dfrac{\psi_L\sqrt{h_L} + \psi_R\sqrt{h_R}}{\sqrt{h_L} + \sqrt{h_R}}. \end{cases} \tag{3.1.8}$$

The average eigenvalues (of what?) are

$$\tilde{\lambda}_1 = \tilde{u} - \tilde{a}, \quad \tilde{\lambda}_2 = \tilde{u}, \quad \tilde{\lambda}_3 = \tilde{u} + \tilde{a},$$

with the correspinding right eigenvectors

$$\tilde{\mathbf{R}}^{(1)} = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{\psi} \end{bmatrix}, \quad \tilde{\mathbf{R}}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{R}}^{(3)} = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{\psi} \end{bmatrix}.$$

The wave strenghts $\tilde{\alpha}_j$ described by Roe averages are given by

$$\begin{cases} \tilde{\alpha}_1 = \dfrac{1}{2}\left[\Delta h - \dfrac{\tilde{h}}{\tilde{a}}\Delta u\right], \\ \tilde{\alpha}_2 = \dfrac{1}{2}\left[\tilde{h}\Delta\psi\right], \\ \tilde{\alpha}_3 = \dfrac{1}{2}\left[\Delta h + \dfrac{\tilde{h}}{\tilde{a}}\Delta u\right]. \end{cases} \tag{3.1.9}$$

Applying theory of linear systems with constant coefficients. The numerical flux is

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) - \frac{1}{2}\sum_{j=1}^{3} \tilde{\alpha}_j \left|\tilde{\lambda}_j\right| \tilde{\mathbf{R}}^{(j)}. \tag{3.1.10}$$

The Roe flux (3.1.10) is used in the explicit conservative scheme to solve the SWE in 1D.

3.2 Implementation of the FVM in 1D

The code to solve the SWE in 1D using FVM is based on the Godunov scheme with the exact Riemann solver. The exact solution of the Riemann problem is found by using the Riemann invariants and the Rankine-Hugoniot conditions [4].

3.3 True solution

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.

3.4 FVM in 2D

3.4.1 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:

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+1,j} - \mathbf{F}_{i,j}) - \frac{\Delta t}{\Delta y}(\mathbf{G}_{i,j-1} - \mathbf{G}_{i,j}). \tag{3.4.1}$$

3.5 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 [10] 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 3.1.

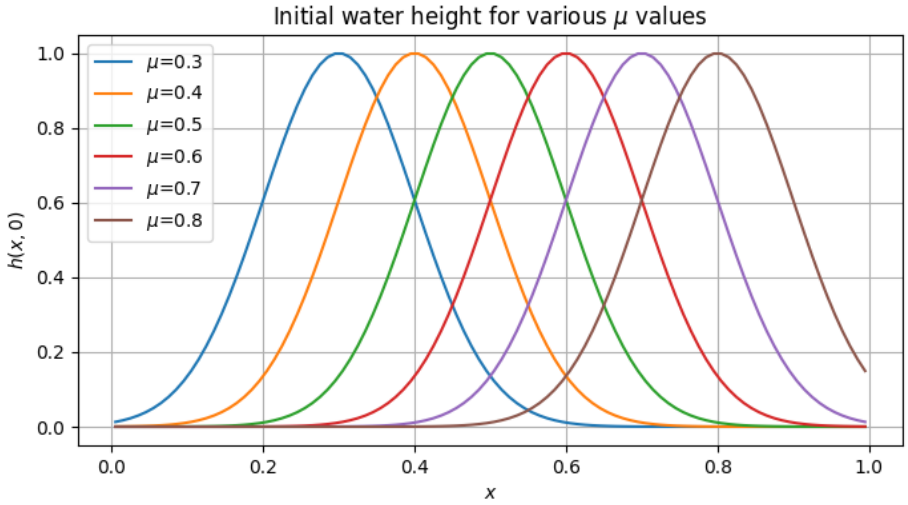


Figure 3.1: Initial conditions for the data generation.

Chapter 4

Results

4.1 Numerical results

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 verify the implementation, as it will be used to generate data for the data-driven methods, including neural networks and Fourier neural operators.

4.1.1 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 [11], can be seen in Figure 4.1.

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

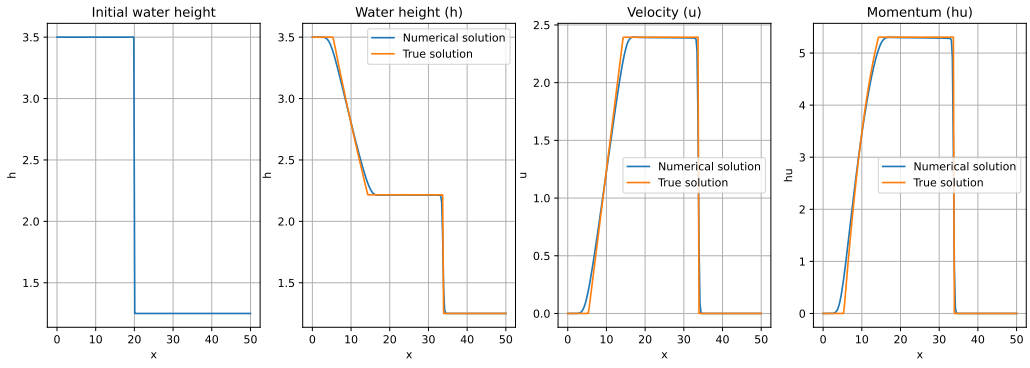


Figure 4.1: The initial water height h at $t = 0$, together with the water height, the fluid velocity u and the momentum hu after $t = 2.5$ seconds.

From Figure 4.1 we see that the numerical solution aligns well with the true solution, and successfully captures the discontinuity.

4.1.2 Toro test cases

We have tested the method on the five test cases from Toros book [1]. The initial conditions for the five test cases are given in Table 4.1.

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 4.1: Initial conditions for the five test cases.

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

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 4.2 and the final solution after $t = 7.0$ seconds is given in Figure 4.3. In test case 1, we observe a right shock wave and a left rarefaction wave.

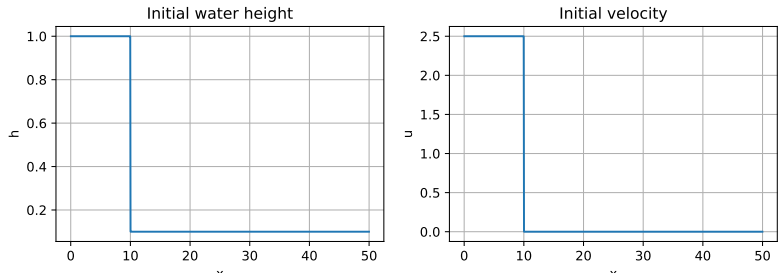


Figure 4.2: Initial conditions for the test case.

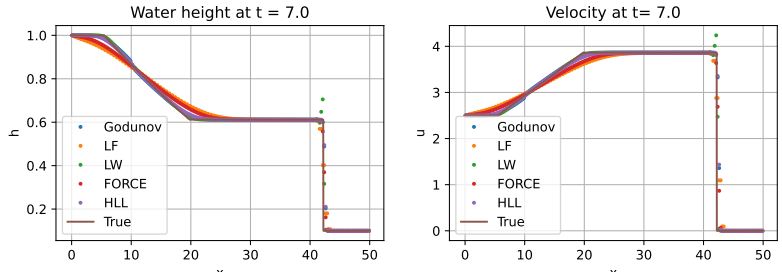


Figure 4.3: Final solution for the test case.

For this test case all the fluxes work well, but there are minor differences in the solution, which can be seen in Figure 4.3.

Test case 2

The initial conditions for test case 2 are illustrated in Figure 4.4.

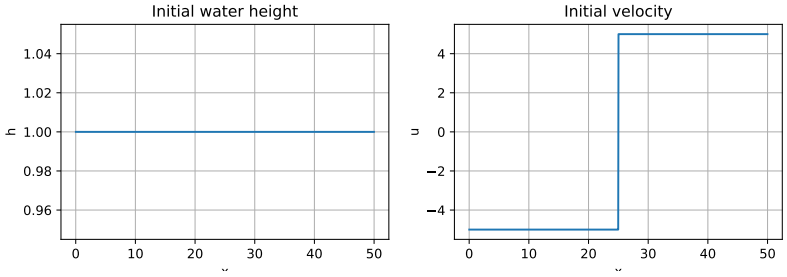


Figure 4.4: Initial conditions for the test case.

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 HLLC flux. The final solution after $t = 2.5$ seconds is given in Figure 4.5.

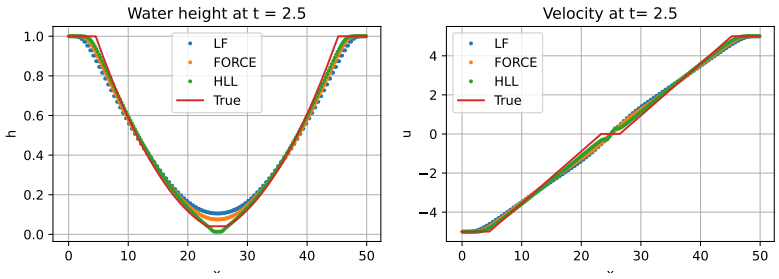


Figure 4.5: Final solution for the test case.

For some of the other fluxes, Godunov method with exact Riemann solver, Lax-Wendroff or flux-splitting/upwind, it was not possible to get an acceptable solution.

Test case 3

The initial conditions for test case 3 are given in Figure 4.6, and the final solution after $t = 4.0$ seconds is given in Figure 4.7.

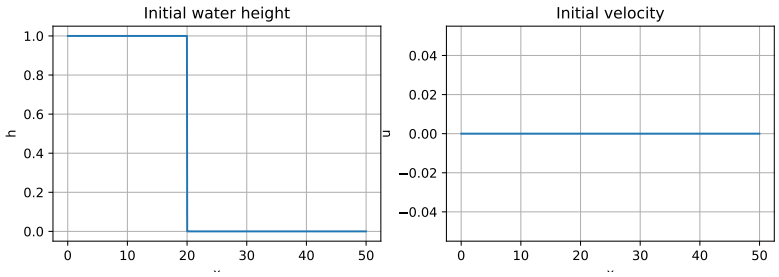


Figure 4.6: Initial conditions for the test case.

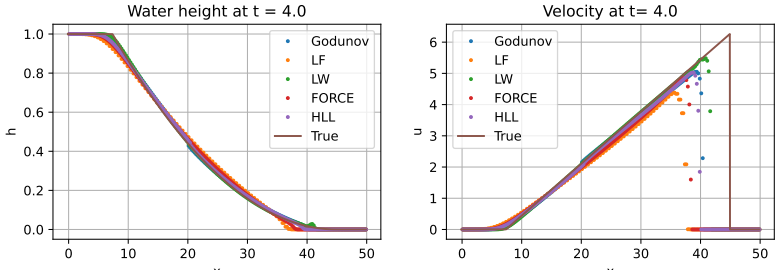


Figure 4.7: Final solution for the test case.

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.

Test case 4

The initial conditions for test case 4 are given in Figure 4.8, and the final solution after $t = 4.0$ seconds is given in Figure 4.9.

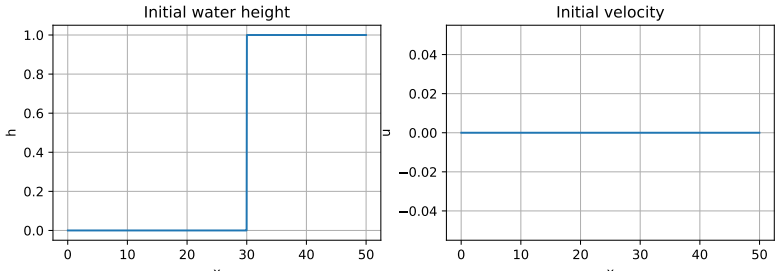


Figure 4.8: Initial conditions for the test case.

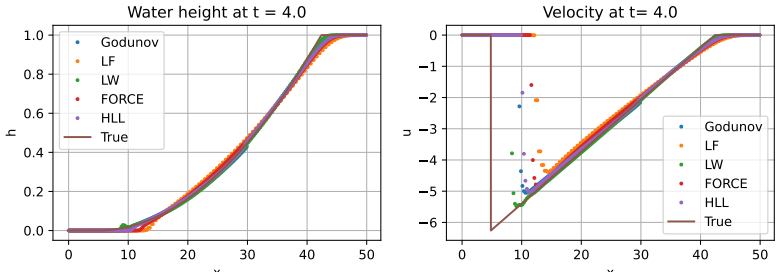


Figure 4.9: Final solution for the test case.

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.

Test case 5

The initial conditions for test case 5 are given in Figure 4.10, and the final solution after $t = 5.0$ seconds is given in Figure 4.11.

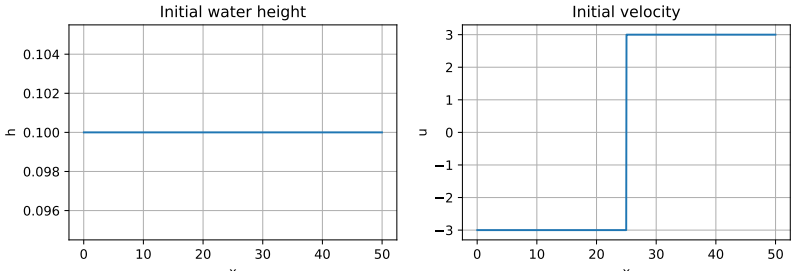


Figure 4.10: Initial conditions for the test case.

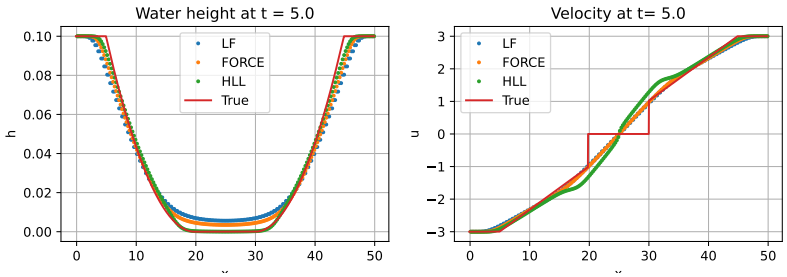


Figure 4.11: Final solution for the test case.

From Figure 4.11 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 dont work are: Godunov method with exact RP, Lax-Wendroff and flux-splitting/upwind. 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 solve the problems, consider the table below.

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

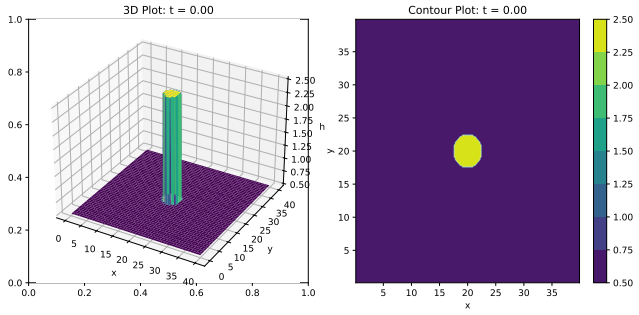
Table 4.2: Overview of which fluxes that were able to solve the test cases.

4.1.3 2D idealised Circular Dam Break Problem

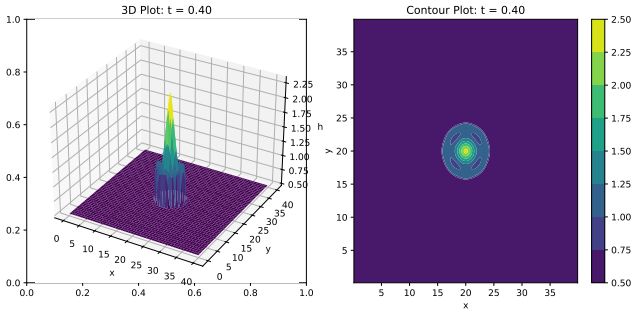
Consider the idealised circular dam break problem with a horizontal bottom. We assume there is an infinitely thin circular wall at radius $R = 2.5$ m is a square domain of size 40×40 with centre at $(x_c, y_c) = (20, 20)$. The initial conditions are (ch. 15 Toro)

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

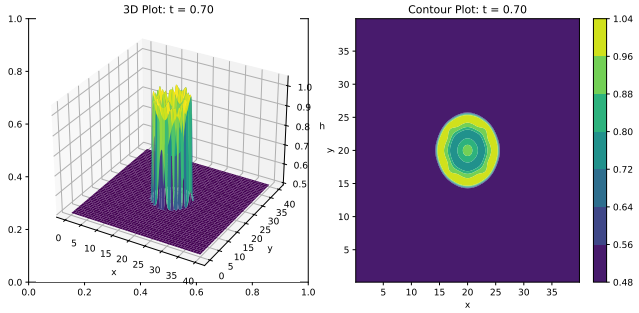
We use a mesh of size 200×200. The results after $t = 0.0, 0.4, 0.7$ and 1.4 seconds are given in Figure 4.12.



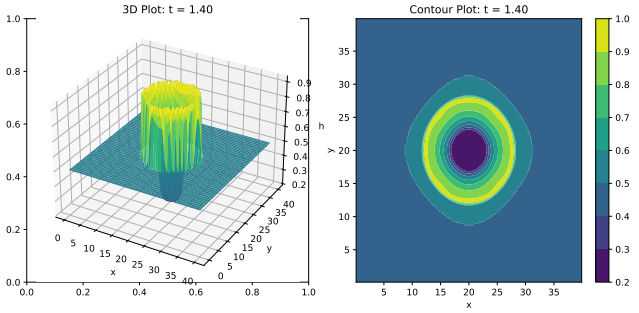
(a) 2D dam break problem after $t = 0$.



(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 4.12: Snapshots of the 2D dam break problem at different times.

By comparing Figure 4.12 with the results from the book by Toro [3], and see that the numerical solution aligns well with the true solution from the book.

4.2 Data-driven results

In this section we present the results of the data-driven models. We start by showing the numerical solution of the shallow water equations, then we present the predictions of the FNN 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:

$$\begin{aligned} h(x,0) &= h_0 \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \\ u(x,0) &= 0, \end{aligned} \tag{4.2.1}$$

where $h_0 = 1, \mu = 0.5, \sigma = 0.1$. 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 4.13, in both a contour plot and a 3D plot.

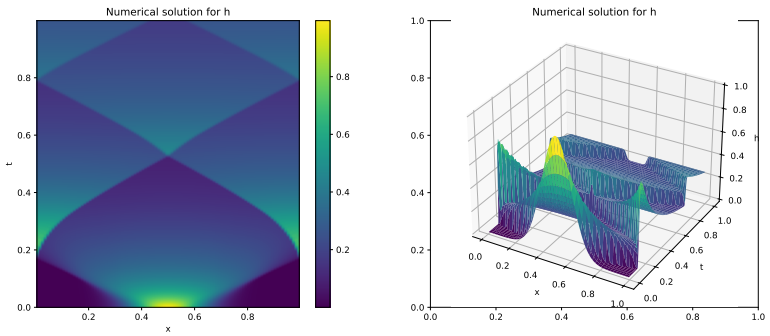


Figure 4.13: Numerical solution of the shallow water equations.

4.2.1 FNN

In the feedforward neural network, we train the model using the data generated by the numerical solution of the shallow water equations. The neural network consists of the following layers: a input dense layer, three hidden dense layers with ReLU activation functions, a batch normalization layer for stability, a dropout layer to prevent overfitting, and an output dense layer. We have also included L2 regularization in the hidden layers to prevent overfitting. Themodel has been trained using the Adam optimizer with a learning rate of 0.01, a batch size of 16 and a total of 3000 epochs. We train the model on the data from $t = 0$ to $t = 0.8$, and test it on the data from $t = 0.8$ to $t = 1.0$. The predictions of the model are shown in Figure 4.14.

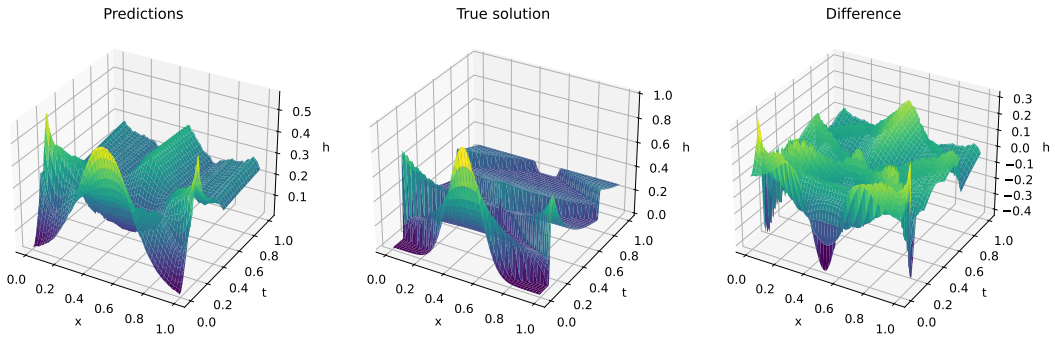


Figure 4.14: Predictions.

From Figure 4.14 we see that the neural network, to some extent, is able to learnt the dynamics of the solution, but it is not accurate enough to be used as a solver. We also see that the further we get in time, the worse the predictions become. This is also somehow expected. To understand the performance of the model, we consider the predictions for some given time steps, shown in Figure 4.15.

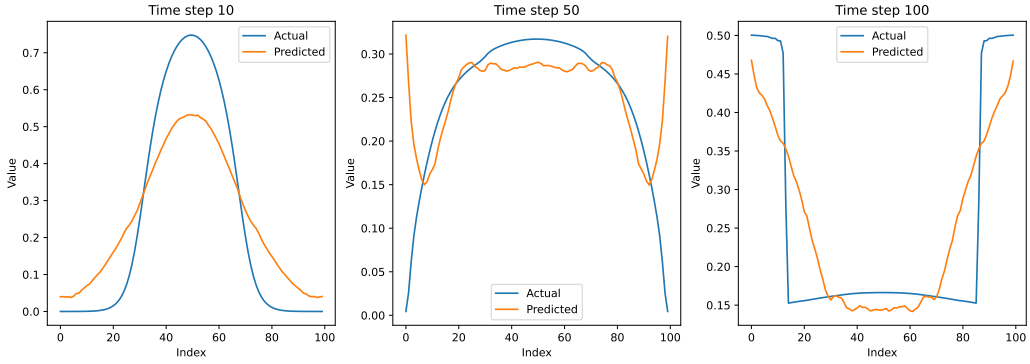


Figure 4.15: Predictions for some given time steps.

Again, we see that the model finds some of the dynamics, but in practise, it can not be used as a solver.

The training and validation loss are shown in Figure 4.16.

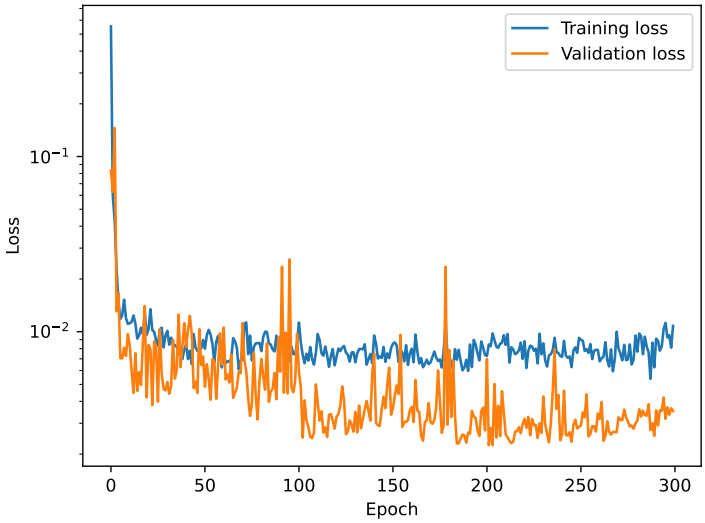


Figure 4.16: Training and validation loss.

From Figure 4.16, we see that a higher number of epochs, would probably not be beneficial, as the validation loss is increasing after a certain number of epochs.

We have also trained a LSTM model, but the results are not shown here, as the model was not able to learn the dynamics of the solution.

4.2.2 FNO

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 2 input channels, 1 output channel, 100 Fourier modes in the first dimension and 1 Fourier mode in the second dimension. The model is trained using the Adam optimizer with a learning rate of 0.001, a total of 3000 epochs and the critera is to minimize the mean squared error (MSE). The results of the model are shown in Figure 4.17.

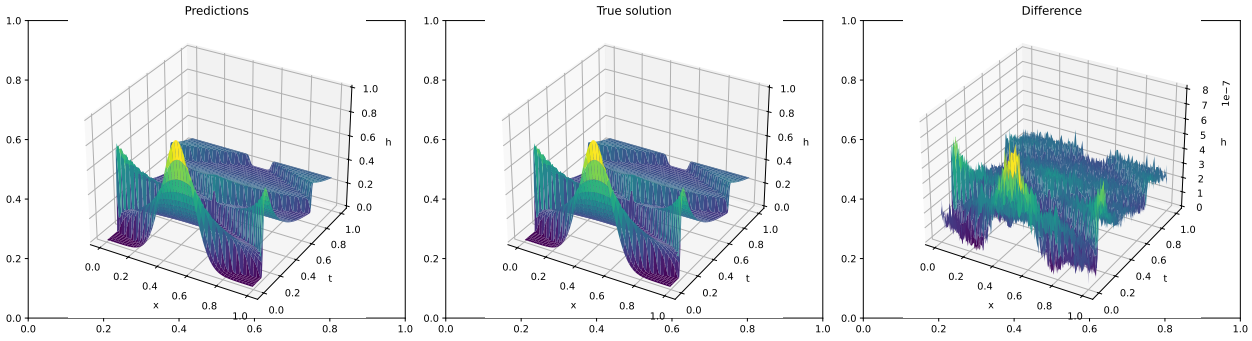


Figure 4.17: FNO predictions.

From Figure 4.17 we see that the FNO model is able to learn the dynamics of the solution, and it is able to predict the solution with high accuracy. From the figure we also see that distribution of the error is more or less uniform in the domain. If the solution has a higher value, we see correspondingly, the error is a bit higher, i.e, the error follows the solution. The loss for each epoch is shown in Figure 4.18.

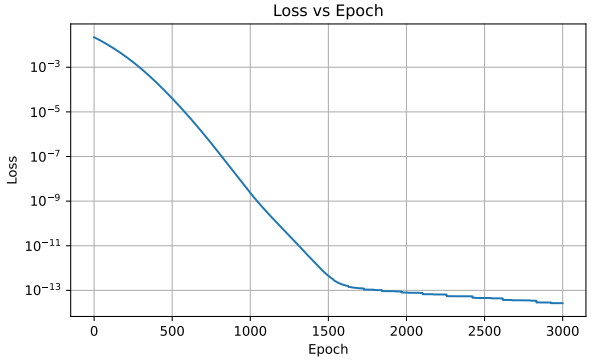


Figure 4.18: FNO loss.

From Figure 4.18 we see that the loss is decreasing for each epoch, and the model is able to learn the dynamics of the solution.

4.2.3 Neural Operator

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

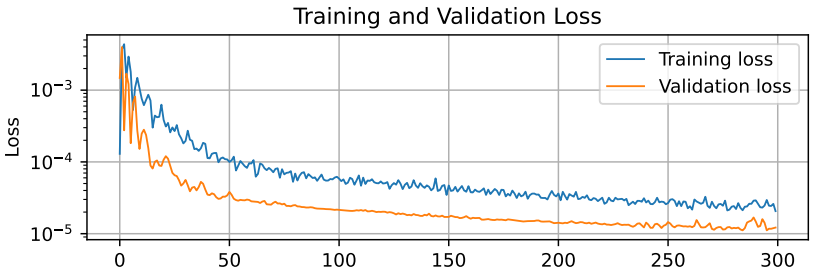


Figure 4.19: FNO loss.

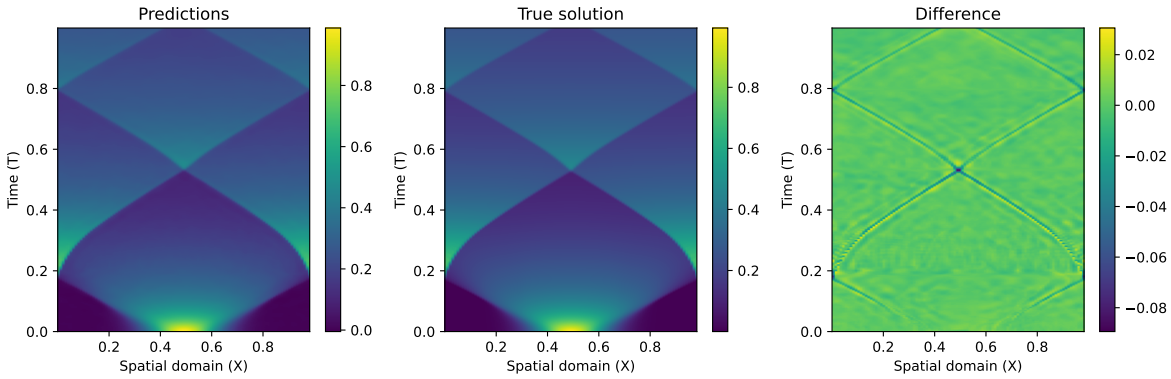


Figure 4.20: FNO predictions.

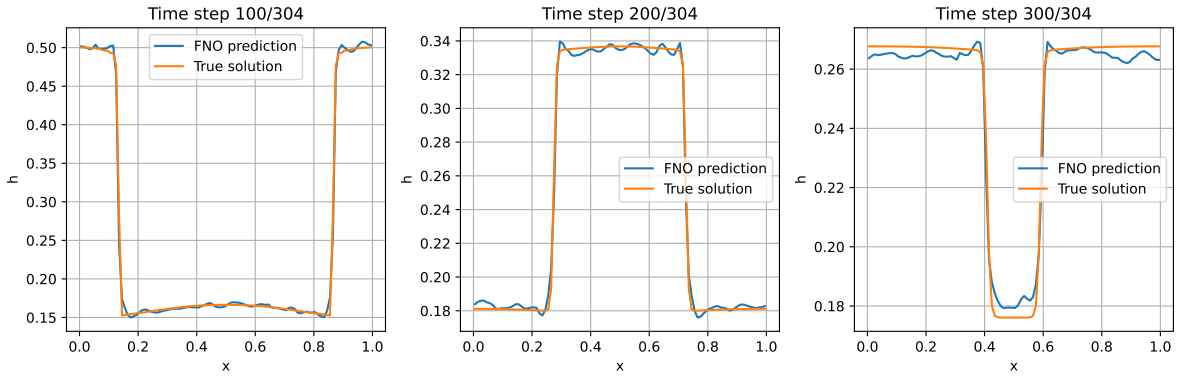


Figure 4.21: FNO predictions timesteps.

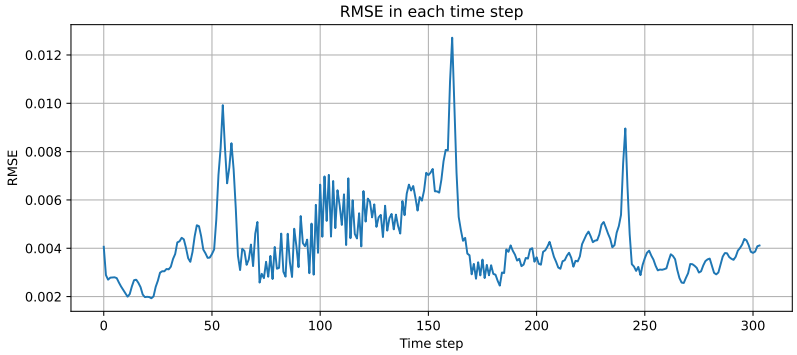


Figure 4.22: FNO predictions RMSE.

Chapter 5

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.

Chapter 6

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] 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.

[6] 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.

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

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

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

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

[11] 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.