Maximilian Zöch, Bsc

# Physics-Informed Neural Network Surrogate Models for River Stage Prediction

**Master's Thesis**
to achieve the university degree of
Master of Science
Master's degree programme: Computer Science

submitted to

**Graz University of Technology**

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Kappe

Co-Supervisors
Dr. Mahdi Abdelguerfi
Dr. Ted Holmberg

Canizaro Livingston Gulf States Center for Environmental Informatics

Graz, November 2024

Maximilian Zöch, Bsc

# ML-basierte Surrogatmodelle für die Vorhersage des Wasserstands von Flüssen

**Masterarbeit**
zur Erlangung des akademischen Grades eines
Diplom-Ingenieur
Masterstudium: Computer Science

eingereicht an der

**Technische Universität Graz**

Betreuer
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Institute of Interactive Systems and Data Science
Vorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Kappe

Mitbetreuer
Dr. Mahdi Abdelguerfi
Dr. Ted Holmberg

Canizaro Livingston Gulf States Center for Environmental Informatics

Graz, November 2024

# Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

| Date | Signature |
| --- | --- |

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

# Abstract

As extreme weather events become more frequent, flooding events become an increasing concern for a large number of comminutes around the world. River Analysis is an important tool to predict and prepare for such events. However, traditional river analysis tools such as HEC-RAS rely on solving partial differential equations (PDEs) which is a very computationally demanding task. Surrogate models are simplified models which trade model fidelity for improved inference performance. This thesis evaluates the feasibility to develop surrogate models for flood prediction using physics-informed neural networks.

The proposed physics-informed neural network based architecture learns an implicit function of the water depth in rivers on simulation data provided by the U.S. Army Corps of Engineers (USACE). The model has two key characteristics: A regularization term that is used to penalize results which do no conform to the underlying physical model, and random Fourier features which are used to encode the model input. Using those techniques, a model architecture to approximate the water surface level of a single river is presented. This single-river model is then conditioned by a geometry encoder, which leads to the multi-river model that is able to learn the geometry of the riverbed. This allows the multi-river model to approximate an entire system of rivers.

To conduct a quantitative analysis, the accuracy of model predictions are measured against the simulation data. Furthermore, in order to evaluate the model qualitatively, an analysis of the learned latent space as well as an ablation study are conducted. The results show that the models are capable of approximating the simulation data, with the multi-river model outperforming the single-river model. Furthermore, the ablation study demonstrates the impact of both the physics-based regularization and the random Fourier features on the representational power of the model. While the results demonstrate the general feasibility of the approach, with an average mean absolute error of 8.342 ft (2.54 m), the obtained results, however, are still significantly worse than the numerical simulation.

# Kurzfassung

Mit der Zunahme von Extremwetterereignissen werden Hochwasserereignisse für viele Menschen zu einem zunehmenden Problem. Die Analyse von Flüssen stellt daher ein wichtiges Instrument zur Vorhersage und Reaktion auf derartige Ereignisse dar. Traditionelle Modelle wie HEC-RAS allerdings basieren auf der numerischen Lösung partieller Differentialgleichungen, was einen sehr rechenintensiven Prozess darstellt. Surrogatmodelle sind eine vereinfachte Form eines Modells, deren Berechnung mit weniger Ressourcen möglich ist. Im Rahmen dieser Masterarbeit erfolgt eine Evaluierung der Machbarkeit eines Surrogatmodells auf Basis von physics-informed Neural Networks.

Die vorgestellte Modellarchitektur erlernt für den Wasserstand eines Flusses eine implizite Funktion mit Hilfe von Simulationsdaten, welche vom U.S. Army Corps of Engineers zur Verfügung gestellt wurden. Das Modell besitzt dabei zwei entscheidende Eigenschaften: Einen physikalisch motivierten Regularisierungsterm, welcher Vorhersagen, die nicht dem zu Grunde liegenden physikalen Modell entsprechen unwahrscheinlicher macht, und eine Transformation der Eingabewerte mit Hilfe von zufälligen Fourierfeatures. Zunächst wird ein Modell für nur einen einzelnen Fluss trainiert, welches dann durch einen Geometrie-Encoder konditioniert wird. Der Geometrie-Encoder erlernt die Form des Flussbetts und ermöglicht dem Modell so mehrere Flüsse zu repräsentieren.

Für eine quantitative Analyse der Ergebnisse wurden die Vorhersagen des Modells mit den Simulationsdaten verglichen. Darüber hinaus wird die Struktur des Latent Space und der Einfluss einzelner Komponenten in einer Ablation Studie untersucht. Die Ergebnisse der Evaluierung zeigen, dass die Modelle in der Lage sind, die Simulationsdaten vorherzusagen. Des Weiteren zeigen die Ergebnisse den Einfluss der Regularisierung und der Fourierfeatures auf die Fähigkeit des Modells, die Daten zu repräsentieren. Während die Ergebnisse zwar die generelle Machbarkeit des Ansatzes demonstrieren, ist die Genauigkeit des Modells mit einem durchschnittlichen absoluten Fehler von 2,54m signifikant schlechter als die des Ursprungsmodells.

# Acknowledgments

I would like to start by thanking Ted Holmberg for being such a great supporter throughout this project. I am also really grateful to Dr. Jay Ratcliff and Dr. Maik Flanagin from the U.S. Army Corps of Engineers for sharing their expertise in river modeling. I would also like to thank Dr. Mahdi Abdelguerfi, Professor Christian Guetl, and the Austrian Marshall Plan Foundation for giving me the opportunity to do this research at the University of New Orleans. Further, I would like to express my gratitude to Professor Hubert Chanson for providing me with figure 2.1 from his book.

Finally, I would like to thank my family for supporting me throughout this journey.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Flooding and other extreme weather events are a major challenge for many communities around the world. Extreme weather events can cause severe property damage, impact the physical and mental well-being of people (Augustin et al., 2024), and even cause loss of life. For example, the 2021 floods in Germany caused damage of around 33 billion euro and resulted in 189 people loosing their lives (Thieken et al., 2023). More recently, in the 2024 floods in eastern and central Europe caused severe property damage and loss of live (Henley, 2024). The Intergovernmental Panel on Climate Change (IPCC) projects that climate change will make flooding events like these occur more frequently (IPCC, 2022). Investing into flood prevention and other mitigation strategies is therefore more important then ever.

The physics of water flow in rivers are very well understood, so the flow can be simulated by solving the so-called governing equations of the process. However, the process of this numerical simulation is often time consuming and requires a lot of compute resources. Thus, in settings where model performance is critical (like for example when making real-time decisions), it is common practice to utilize simplified models. These surrogate models trade computational resources for model fidelity.

In recent years, Deep Learning has gained interest in the research community as a way of constructing surrogate models from data. The aim of this thesis is to study the potential of neural networks to build surrogate models based on data provided by the U.S. Army Corps of Engineers (USACE). This work was conducted in collaboration with the Canizaro Livingston Gulf States Center for Environmental Informatics (GulfSCEI) at the University of New Orleans.

## 1.1. Aim and Objective

The main aim of this project is to build a surrogate model based on the synthetic data provided by USACE. The project outlined in this thesis is focused on predicting the water surface height of rivers which is of most interest for flood prevention.

A central challenge when learning neural networks from physical data is that

the data tends to be sparse. For example, in case of real-world rivers, measurements are not taking at fixed intervals along the river. Instead, river stations measuring the height and flow velocity of the water are placed strategically at interesting points. Similarly, the simulation data is not provided in a dense grid. Cross-sections are placed strategically by modellers at points of interest where the flow of the river actually changes. However, the learned neural network has to make physically sound predictions at every location along the river.

A common method of encouraging neural networks to learn physically sound solutions is by using physics-informed regularization. Because the underlying laws of physics producing the sparse data are well known, the model can be penalized for predicting data that does not conform to it. The aim of this project is to study the network architecture and training scheme to train a physics-informed neural network that can approximate simulation data.

This should lead to a simple surrogate model that captures the main dynamics of the system. The surrogate model can also serve as a basis for future research in this area.

## 1.2. Methodology and Contribution

A model architecture is presented to learn a surrogate model for the simulation data. This architecture is based on principles used in state of the art Deep Learning models, and can either be trained on single river or be augmented to encode the shape of the riverbed in order to be trained on multiple rivers. Then, a training scheme used to train the models is presented which implements a physics-informed neural network.

The model was trained using simulation data computed using HEC-RAS, a river analysis tool developed by USACE. Because HEC-RAS provides a very powerful tool to perform river analysis, the surrogate model is only able to use a subset of the information provided by HEC-RAS, and focuses on predicting the water surface height from the geometry of the riverbed including its roughness. It does however neglect factors like structures built in the river or connected lakes.

Next, both a quantitative and qualitative analysis of the results is presented. In an ablation study, the impact of individual design decisions for the model is illustrated. It studies both architectural decisions made in the design of the model itself, as well as the impact of the physics-informed neural network. Lastly, implementation details on the process used to prepare the simulation data for Deep Learning is provided in order to help facilitate future research in

this direction.

## 1.3. Structure

The rest of this thesis is organized as follows. Chapter 2 covers background and related work. It discusses the foundations of river analysis, and derives the governing equations used. It further presents physics-informed neural networks, and discusses how the regularization term can be implemented using common Deep Learning frameworks. Lastly, it gives examples of Deep Learning based surrogate models used for weather and fluid simulation.

Chapter 3 covers requirements and the overall concept, before chapter 4 describes the methodology. Chapter 4 first describes the construction of a model trained only on single river, which is then extended to cover a whole set of rivers. Lastly, the chapter presents how the physics-informed regularization is used to implement the equations derived in chapter 2.

Chapter 5 analyzes the results of the models presented in the previous chapter. It provides both quantitative results for the prediction error, presents an ablation study for the different components of the river, and presents an analysis of the latent space.

Chapters 6 and 7 reflect on lessons learned and conclude the thesis respectively.

# 2. Background and Related Work

This chapter provides an overview of the background and related work for the methodology. Firstly, a high level introduction to the simulation of water flow in rivers is given. The basic concepts behind computational fluid dynamics are explained, after which the workflow of performing simulations in HEC-RAS is described. In the related work section, a brief review of physics-informed neural networks, model reduction, and deep learning surrogate models is provided. The literature reviewed focuses on use cases in fluid simulation and numerical weather prediction.

## 2.1. River Simulation

River simulation is an important part of modern day hydraulic engineering and flood protection. It is commonly performed by solving a series of partial differential equations using specialized simulation software. This section describes the basic concepts of unsteady flow in open channels. The *Saint-Venant* equations used for the physical model, the description of the river bed geometry including the friction force are discussed. Afterwards, the workflow in HEC-RAS, a software suite used to perform river simulation, is presented.

### 2.1.1. Open Channel Flow Analysis

The task of open channel flow analysis is to estimate the velocity and water depth of a river. Open channels are any water channel where the water flows with a free surface, like canals or rivers. There are two settings considered when simulating water in an open channel: Steady flow and unsteady flow. Steady flow analysis assumes that the velocity, pressure, and cross-sections of the river will stay the same over time. Unsteady flow analysis on the other hand assumes that these parameters do vary over time and is commonly used for the analysis of natural rivers. Cross-sections are 2D slices of the river bed along the entire river, and the volume between two of such cross-sections is called the *control volume*. This control volume is parameterized by the parameters timestamp $t$ and river position $x$. The timestamp is usually given as the day of the year. The river position is given as the river mile downstream from the river source. The surrogate models presented in this thesis focus on 1D unsteady flow analysis, which only predicts a single value for the entire cross section at a given position

$x$. (Chanson, 2004, p.318).

1D unsteady flow analysis can be described by the *Saint-Venant equations*. These equations model two fundamental properties of water in an open channel:

1. Continuity equation: The net mass influx is equal to the total mass increase of the volume. If no water is added to the channel (e.g. through rainfall), the total mass of the volume stays constant.
2. Momentum equation: The change of the momentum, plus the rate of change of the momentum flux is equal to all the forces that act on the fluid.

Momentum per unit volume is defined as

$$m = pv \tag{2.1}$$

where $p$ is the density of the fluid and $v$ its velocity. The momentum flux is defined as

$$\dot{m} = pv^2 \tag{2.2}$$

where $p$ and $v$ are again the fluid density and velocity respectively, and describes the rate of change of the momentum flowing through an area $A$ (Chanson, 2004, p.293). These two equations will now be discussed in further detail.

### Continuity Equation

The continuity equation states that the total mass increase is equal to the net influx and outflow into and out of the control volume (Chanson, 2004, eq. 16.1)

$$\int_{t_1}^{t_2}(Q_1 - Q_2)dt + \int_{x_1}^{x_2}(A_{t_1} - A_{t_2})dx = 0 \tag{2.3}$$

where $Q_j$ refers to the discharge which is defined as:

$$Q_j = vA_j \tag{2.4}$$

Here, $A_j$ refers to the area of the upstream cross-section at the positions $x_1$ and $x_2$ (Chanson, 2004, p.293). The subscripts $t_1$ and $t_2$ refer to two points in time. In this case we assume that the mass in the control volume stays constant and is set to zero. This assumption ignores effects like rainfall that would increase the total mass (Feng et al., 2023).

### Momentum Equation

The momentum equation states that net change of momentum in the control volume between the timestamps $t_1$ and $t_2$ and the change of the momentum flux over the entire control volume is equal to the sum of all the forces that act on the volume. Firstly, the net change of the momentum can be described by (Chanson, 2004, eq. 16.3):

$$\int_{x_1}^{x_2}((pvA)_{t_1} - ((pvA)_{t_2})dx \tag{2.5}$$

where $x_1$ and $x_2$ are the locations of the cross-sections upstream and downstream respectively, $p$ is the fluid density, $v$ the velocity, and $A$ the area of the cross-section. The difference of the momentum between the timestamps $t_1$ and $t_2$ is then integrated across the control volume from $x_1$ to $x_2$.

The rate of change of the momentum flux is described as (Chanson, 2004, eq. 16.4):

$$\int_{t_1}^{t_2}((pv^2A)_1 - ((pv^2A)_2)dt \tag{2.6}$$

It is similar to equation 2.5 for the momentum, where $x_1$ and $x_2$ again are the river locations, $p$ is the fluid density, and $A$ is the area of the cross-section. Now, however, the change in the momentum flux between the upstream and downstream cross-sections is integrated over time instead of space. The subscripts 1 and 2 refer to the upstream and downstream cross-section of the control volume (Chanson, 2004, p.293). The sum of the terms 2.5 and 2.6 need to be equal to forces that act on the control volume.

The following forces are considered (Chanson, 2004, p.293 USACE Hydrologic Engineering Center, n.d.):

1. Pressure forces
2. Gravity
3. Friction force

These forces will now be discussed in greater detail.

**Pressure.** The walls of the river bed apply pressure forces that keep the water inside the bed. There are two kinds of pressure that act on the control volume: Forces upstream and downstream of the cross-sections, and forces coming from walls on the side. Figure 2.1 gives an overview of the pressure forces which act on the volume.

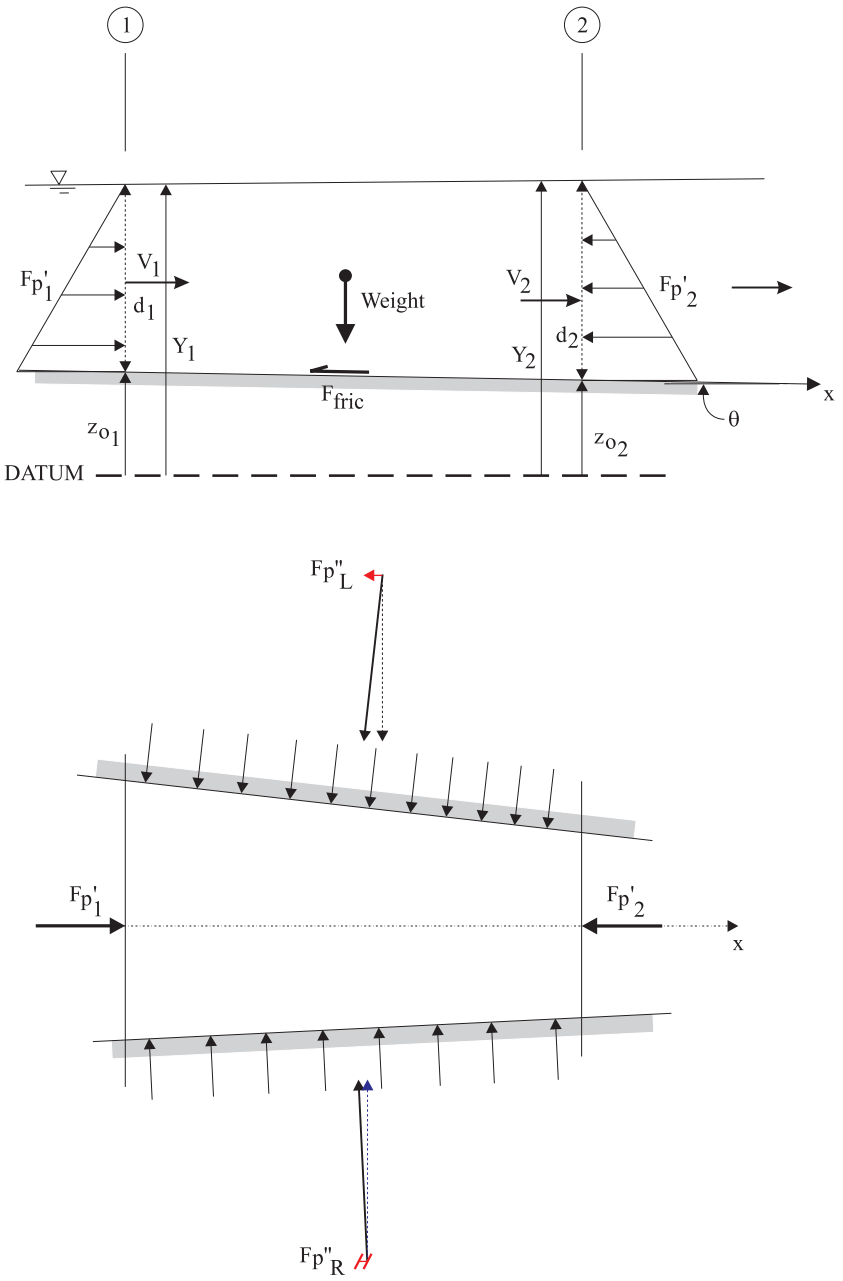



Figure 2.1.: The forces acting on the control volume. The upper view depicts the view from the side, the lower view the view from the top (Chanson, 2004, p.294)

The pressure forces at the cross sections can be described as following (Chanson, 2004, eq. 16.5):

$$\int_{t_1}^{t_2}(F'_{P_1} - F'_{P_2})dt = \int_{t_1}^{t_2}((pI_1)_1 - (pI_1)_2)dt \tag{2.7}$$

where $I_1$ is defined as:

$$I_1 = \int_0^d (d-y)Wdy \tag{2.8}$$

$F'_{P_1}$ and $F'_{P_2}$ are forces coming from the fluid upstream or downstream in the river, $d$ in the water depth (the distance from the water surface to lowest point of the river bed $z_0$), $y$ is the distance from bed level $z_0$, and $W$ is the width at the distance $y$.

The pressure forces from the side walls of the river bed are described by the following (Chanson, 2004, eq. 16.6):

$$\int_{t_1}^{t_2}(F''_{P_1} + F''_{P_2})dt = \int_{t_1}^{t_2}\int_{x_1}^{x_2} pgI_2 dxdt \tag{2.9}$$

where $I_2$ is defined as:

$$I_2 = \int_0^d (d-y)\frac{\partial W}{\partial x}dy \tag{2.10}$$

Here, the forces $F''_{P_L}$ and $F''_{P_R}$ describe the forces coming from the left and right wall of the river bed, $p$ is the fluid density, and $g$ the gravity. This equation assumes that the width of the channel only changes gradually and is only valid in that case (Chanson, 2004, p.294-295).

**Gravity.** The gravity forces acting on the water are given by the following formula (Chanson, 2004, eq. 16.7):

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2} pgAS_0 dxdt \tag{2.11}$$

where $p$ is the density, $g$ is the gravity, $A$ the surface of the cross-section, and $S_0$ the slope of the river bed. The slope can be approximated as $S_0 = \sin\theta \approx \frac{\partial z_0}{\partial x}$ where $z_0$ is the ground level of the riverbed and $\theta$ is the downhill slope of the riverbed (Chanson, 2004, p.295).

**Friction.** The friction forces act on the fluid counter to the gravity forces (Chanson, 2004, eq. 16.8):

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2} pgAS_f dxdt \tag{2.12}$$

where density $p$, gravity $g$, area $A$, are the same as in equation 2.11. $S_f$ is the friction slope which can be estimated using the Chézy-Mannings equation. The derivation of this equation is further detailed in section 2.1.2.
We can now combine the different forces to formulate the momentum equation for unsteady flow. The friction acts opposite to the gravity force, thus combining gravity and friction forces yields:

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2}(pgAS_0 - pgAS_f)dxdt \tag{2.13}$$

Combining the friction forces with the equations 2.7 and 2.9 for the cross section and side wall pressure forces, respectively, yields the following for the following sum of forces:

$$\int_{t_1}^{t_2}((pI_1)_1 - (pI_1)_2)dt + \int_{t_1}^{t_2}\int_{x_1}^{x_2} pgI_2 dxdt + \int_{t_1}^{t_2}\int_{x_1}^{x_2}(pgAS_0 - pgAS_f)dxdt \tag{2.14}$$

When combining this with the definition of the momentum (equation 2.5) and momentum flux (equation 2.6), the final momentum equation in integral form is (Chanson, 2004, eq. 16.10):

$$\begin{aligned}&\int_{x_1}^{x_2}((pVA)_1 - ((pVA)_2)dx + \int_{t_1}^{t_2}((pV^2A)_1 - ((pV^2A)_2)dt = \\ &\int_{t_1}^{t_2}((pI_1)_1 - (pI_1)_2)dt + \int_{t_1}^{t_2}\int_{x_1}^{x_2} pgI_2 dxdt + \int_{t_1}^{t_2}\int_{x_1}^{x_2}(pgAS_0 - pgAS_f)dxdt\end{aligned} \tag{2.15}$$

### Differentiable Form

The form of the partial differentiable equation can be derived from the integral form of the equations. If the parameters for the equation is differentiable with respect to $t$ and $x$, and if they are continuous, the partial differentiable equation (PDE) form of the Saint-Venant equations can be derived from the integral form.

The differentiable form of the continuity equation is then given as (Chanson, 2004, eq. 16.14):

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \tag{2.16}$$

where $A$ is again the area of the cross section, and $Q$ is the discharge (Chanson, 2004, p.297). The momentum equation can be similarly written in its PDE form (Chanson, 2004, eq. 16.15a):

$$\frac{\partial Q}{\partial t} + \frac{\partial v^2 A + gI_1}{\partial x} = g(S_0 - S_f)A + gI_2 \tag{2.17}$$

Now $I_1$ can be expressed in terms of $I_2$ (Chanson, 2004, eq. 16.16):

$$\frac{\partial g I_1}{\partial x} = gA\frac{\partial d}{\partial x} + gI_2 \tag{2.18}$$

where $Q$ refers to the discharge, the velocity $v$, area $A$, gravity $g$, water depth $d$, riverbed slope $S_0$, and friction force $S_f$ are the same as for the integral form given in equation 2.15. When substituting the term in equation 2.18 into equation 2.17, it yields (Chanson, 2004, eq. 16.15b):

$$\frac{\partial Q}{\partial t} + \frac{\partial v^2 A}{\partial x} + gA\frac{\partial d}{\partial x} = gA(S_0 - S_f) \tag{2.19}$$

The equations 2.16 and 2.19 are the PDE forms of the continuity and momentum equations, respectively, and form the basic *Saint-Venant equations* (Chanson, 2004, p.297).

## 2.1.2. Friction Estimation in Open Channels

Equation 2.12 introduced the friction slope $S_f$ in the momentum equation of the Saint-Venant equation. This factor models the friction of the riverbed, and therefore depends on its roughness. For example, riverbeds can vary from clean to being littered with different-sized stones, while floodplains often have different kinds of vegetation growing. It has historically been infeasible to accurately model all these properties in a physically sound manner. Modelers have therefore resorted to empirical correlations and a series of auxillery variables to empirically describe the relationship between the channel geometry and flow velocity. One of the most widely used formulas is the *Chézy-Manning* formula which will be introduced in the following sections.

### Chézys Formula

Chézys formula was first introduced as an empirical correlation between the velocity of the water $v$ and the hydraulic diameter $D_H$ (Chanson, 2004, eq. 4.23)[1]:

$$v = C_{\text{Chézy}}\sqrt{\frac{D_H}{4}S_f} \tag{2.20}$$

where $C_{\text{Chézy}}$ is the Chézy coefficient, which is an auxiliary variable for describing the roughness (Chanson, 2004, p.77). The hydraulic diameter $D_H$ is defined as the ratio between the area of the cross-section and the wetted perimeter (Chanson, 2004, p.70):

$$D_H = \frac{4A}{P_w} \tag{2.21}$$

[1] $S_f = S_0$ (Chanson, 2004, p.76)

| Channel Description | Minimum | Medium | Maximum |
|---|---|---|---|
| a. clean, straight, full stage, no rifts or deep pools | 0.025 | 0.030 | 0.033 |
| b. same as above, but more stones and weeds | 0.030 | 0.035 | 0.040 |
| c. clean, winding, some pools and shoals | 0.033 | 0.040 | 0.045 |
| d. same as above, but some weeds and stones | 0.035 | 0.045 | 0.55 |
| e. same as above, lower stages, more ineffective slopes and sections | 0.040 | 0.048 | 0.055 |
| f. same as d with more stones | 0.045 | 0.050 | 0.060 |
| g. sluggish reaches, weedy, deep pools | 0.050 | 0.070 | 0.080 |
| h. very weedy reaches, deep pools, or floodways with heavy stand of timber and underbrush | 0.075 | 0.1 | 0.15 |

Table 2.1.: A sample of Manning's $n$ values for natural rivers.

In case of simple rectangular channel the area $A$ is defined as $A = Wd$, so the hydraulic diameter becomes

$$D_H = \frac{4Wd}{W + 2d} \tag{2.22}$$

where the $W$ is the channel width and $d$ is the water depth. The geometry of the channel will be discussed in more detail in section 2.1.3 (Malcherek, 2019, p.67).

### Manning's Formula

The Chézy coefficient depends on the hydraulic diameter of the channels. Manning's formula defines a correlation between the hydraulic diameter $D_H$ and a newly introduced coefficient $n$ called *Manning's n* (Chanson, 2004, eq. 4.24):

$$C_{\text{Chézy}} = \frac{1}{n}\left(\frac{D_H}{4}\right)^{\frac{1}{6}} \tag{2.23}$$

The value for Manning's $n$ is empirically determined for different river conditions. As an empirical correlation, this formula is only valid if the water is not too shallow and the channel is not too narrow (Chanson, 2004, p.79). Table 2.1 shows a sample of different values for the Manning's $n$ (Te Chow, 1959).

We can now substitute equation 2.23 for $C_{\text{Chézy}}$ (see Chanson, 2004, eq. 4.25 analogously):

$$v = \frac{1}{n}\left(\frac{D_H}{4}\right)^{\frac{1}{6}}\sqrt{\frac{D_H}{4}S_f} \tag{2.24}$$

$$v = \frac{1}{n}\left(\frac{D_H}{4}\right)^{\frac{2}{3}}\sqrt{S_f} \tag{2.25}$$

Solving for $S_f$ gives the Chézy-Manning formula for the friction force:

$$S_f = \frac{4 \times 2^{\frac{2}{3}} n^2 v^2}{D_H^{4/3}} \tag{2.26}$$

### 2.1.3. River Channel Geometry

Thus far the cross-sections of the river channel have been assumed to be a simple rectangle. While this makes computing the hydraulic diameter very easy, it is also a very coarse approximation of the river geometry. In the following, different models for the cross-section geometry are briefly presented.

#### Trapezoidal Channel

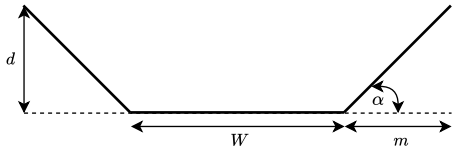



Figure 2.2.: Simple trapezoidal channel (Malcherek, 2019, p.113)

A trapezoid is a more accurate approximation shape of the river channel. It can more accurately approximate the side walls of the river channel, while the area and wetted perimeter required for the hydraulic diameter are still relatively easy to compute. For a trapezoid the area $A$ is defined as:

$$A = Wd + md^2 \tag{2.27}$$

where $W$ is the width of the channel and $d$ the water depth. The wetted perimeter $P_w$ is defined as:

$$P_w = W + 2d + \sqrt{1 + m^2} \tag{2.28}$$

A sketch of the trapezoid river channel can be seen in figure 2.2 (Malcherek, 2019, p.116).

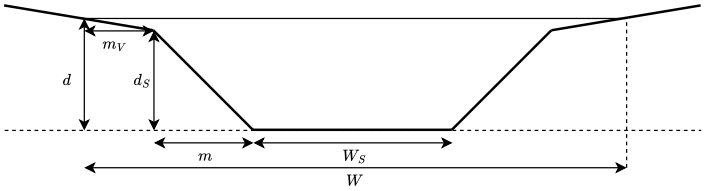



Figure 2.3.: Trapezoidal channel with floodplain (Malcherek, 2019, p.116)

### Trapezoidal Channel with Floodplain

Many natural rivers flow over their trapezoidal river channel in cases where flooding occurs. The trapezoidal model can be extended to account for these floodplains (see figure 2.3). In this case the area $A$ can be computed as

$$A = W_S d + m d_S^2 + 2 d_V m d_S + m_V d_V^2 \tag{2.29}$$

where $d_V$ is defined as

$$d_V = \max(d - d_S, 0) \tag{2.30}$$

The wetted perimeter $P_w$ is then defined as (Malcherek, 2019, p.116-117):

$$P_w = W_S + 2 d_s \sqrt{1 + m^2} + 2 d_v \sqrt{1 + m_V^2} \tag{2.31}$$

### Double Trapezoidal Channel

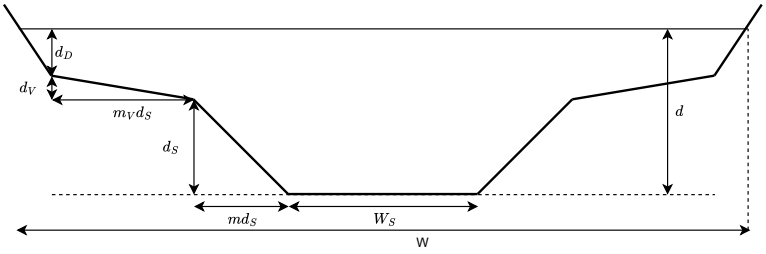



Figure 2.4.: Trapezoidal channel with floodplain and dam (Malcherek, 2019, p.119)

Extending the channel model even further, dams built to prevent flooding can now also be taken into account. If the water overflows the trapezoidal river channel, it first reaches the floodplains discussed earlier. Here, a second trapezoid that models the walls of a dam which can be added to the model. The profile of the cross-section is depicted in figure 2.4. The area $A$ is then defined as

$$A = W_S d + m d_{S'}^2 + 2 m d_{V'} d_S + m_V d_{V'}^2 + 2 m d_D d_S + m d_D^2 \tag{2.32}$$

where $d_{S'}$, $d_{V'}$, and $d_D$ are defined as:

$$d_{S'} = \min(d, d_S) \tag{2.33}$$
$$d_{V'} = \max(\min(d - d_S, d_V), 0) \tag{2.34}$$
$$d_D = \max(d - d_S - d_V, 0) \tag{2.35}$$

The wetted perimeter is then defined as (Malcherek, 2019, p.119-121):

$$P_w = W_S + 2d_{S'}\sqrt{1+m^2} + 2d_{V'}\sqrt{1+m_V^2} + 2d_D\sqrt{1+m^2} \tag{2.36}$$

### 2.1.4. HEC-RAS

HEC-RAS (Hydraulic Engineering Center River Analysis System) is a suite of tools for performing river analysis developed by the U.S. Army Crops of Engineers[2].

HEC-RAS consists of the following components:

1. 1D Steady Flow Analysis
2. 1D/2D Unsteady Flow Analysis
3. Quasi-unsteady or fully unsteady flow movable boundary sediment transport computations
4. 1D water quality analysis

The following section will give an overview of the relevant parts of the software. Firstly, the geometric modelling of rivers will be discussed, after which an overview of the workflow to perform steady and unsteady flow simulations is provided. HEC-RAS, however, is a large, and complex software which means that providing an in-depth discussion of all the capabilities HEC-RAS offers is beyond the scope of this thesis. For more information please refer to USACE Hydrologic Engineering Center, 2024 on which this summary is based.

#### Modelling Geometry in HEC-RAS

First of all, modelers draw a network of rivers, junctions, and reaches. Reaches are a continuous stretch of a river that splits larger rivers into smaller, more manageable pieces. Junctions are created where two rivers meet or split. In addition, modelers can add storage areas, 2D flow areas, pumps, and boundary condition lines. Storage areas are still bodies of water, like lakes; 2D flow areas are areas of water that require more complex 2D flow simulation algorithms. Storage and 2D flow areas can be connected to each other as well as to river

[2]HEC-RAS is available at https://www.hec.usace.army.mil/software/hec-ras/

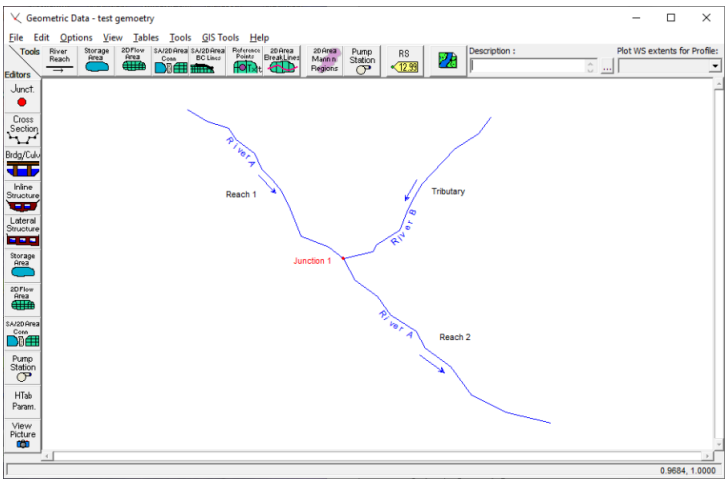



Figure 2.5.: Editor for entering rivers, junctions, and reaches in HEC-RAS (USACE Hydrologic Engineering Center, 2024, Entering and Editing Geometric Data)

reaches. Figure 2.5 depicts the editor to create and modify river networks in HEC-RAS.

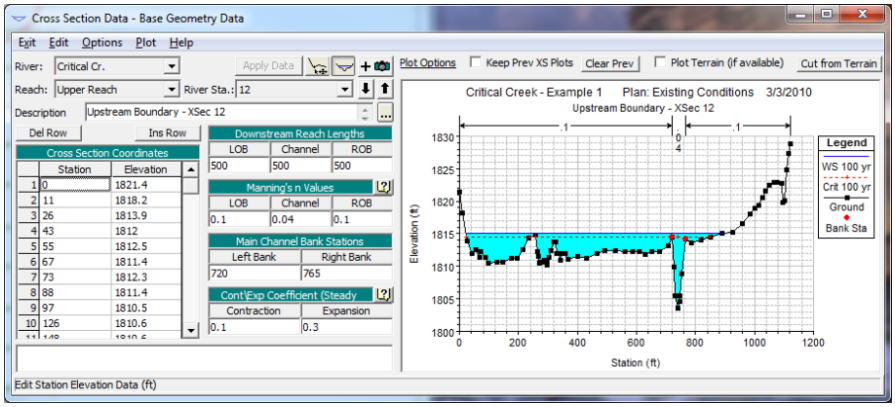



Figure 2.6.: Editor for cross-sections in HEC-RAS (USACE Hydrologic Engineering Center, 2024, Entering and Editing Geometric Data)

After creating the river schematics of rivers, junctions and reaches, cross-sections are placed in relatively small intervals along the river to compute the simulation. The placement of cross-sections is very important for the quality of the resulting river model. Cross-sections are therefore placed strategically at interesting points where, for example, the slope, shape, or roughness of the river changes, or at levees and hydraulic structures such as bridges and culverts. The required data for each cross-section is:

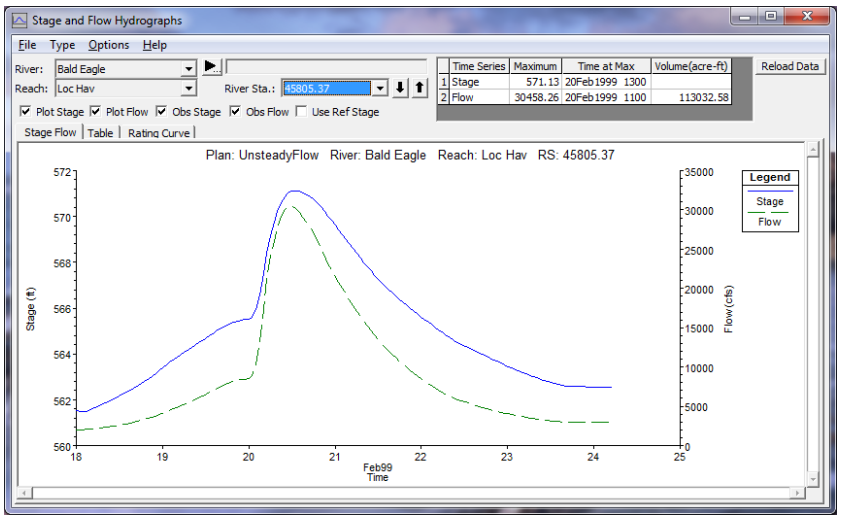



Figure 2.7.: Simulation results plotted in HEC-RAS

1. River and reach where the cross-section is specified
2. Station
3. Coordinates (position and elevation)
4. Coefficients for roughness and contraction and expansion

The station does not need to be a physical river station. HEC-RAS only requires a numerical value of the river mile or kilometer where the river station is located in the river. The coefficients provided for the cross-section are contraction and expansion coefficients for steady flow simulation, and the Manning's $n$ values. For each cross-section at least 3 three Manning's $n$ values need to be provided to describe the roughness of the left and right over bank as well as the main channel. Figure 2.6 depicts the graphical user interface used to edit one of the cross-section in HEC-RAS.

Lastly, after setting up the river schematic and entering data for the cross sections, data for junctions and water structures is added. For junctions, modeler's need to select which simulation method to use. HEC-RAS provides a simplified and an energy balance method to model the flow at the junction. Water structures include bridges, cutlets, dams, weirs, sluice gates and others. When adding these to the river model, HEC-RAS can take into account the energy loss that occurs in the flow around them. In addition, cross-sections need to be placed around structures like bridges in order to accurately simulate the fluid dynamics in those areas of the river (USACE Hydrologic Engineering Center, 2024, Entering and Editing Geometric Data).

#### Performing Flow Analysis

After the river geometry is set up in HEC-RAS, flow analysis can be performed. For steady flow analysis, the system requires the number of flow profiles, the peak flow data, and boundary conditions to be computed. Likewise, for unsteady flow analysis, boundary conditions for the external boundaries, as well as some internal locations, and initial conditions for the river flow and 2D flow area are required. Boundary conditions are the flow rates and stage at the boundary of the river system, and are essential for defining the start and end point of the hydraulic model. The initial conditions define the start conditions for the stage and flow at each cross-section.

After an unsteady flow model has been computed, it needs to be calibrated using real-world data. This is generally done with data of the river stage, as it is seen as the most accurate data reaching an accuracy within $\pm 1$ feet. During the calibration, the roughness values are adjusted to fit the real-world data. For free-flowing rivers, the roughness coefficient will be smaller the faster the water flows and the higher the stage is. In some exceptional cases, the Manning's $n$ can increase with the stage when the overbanks are rougher than the main channel (USACE Hydrologic Engineering Center, 2024, Performing a Steady Flow Analysis, Performing a 1D Unsteady Flow Analysis). After the simulation has been completed, the results can be viewed with visualization tools integrated into HEC-RAS. Figure 2.7 depicts the flow and stage plotted over time. Additionally, the data can also be exported to a HEC-DSS file (USACE Hydrologic Engineering Center, 2024, View Results).

## 2.2. Related Work

In the following section different approaches for surrogate models will be discussed. First, the technique of physics informed neural networks is described which can be used to provide physical prior knowledge to neural networks. Then, projective reduced order models are discussed as a method to create surrogate models that does not rely on machine learning techniques. Finally, examples are presented on how Deep Learning is used in different sciences to develop surrogate models that either are only trained on data or include physical priors.

### 2.2.1. Physics-informed Neural Networks

Physics-informed neural networks, as first introduced by Raissi et al., 2019 are a technique to include a physics based prior knowledge to the neural networks. The main idea is that a lot of data collected in the real world is actually governed by the laws of physics. In addition, in many cases, datasets are either small, or

at least very sparse like the river flow data considered in this thesis. However, when training a neural network on the data, we need to make physically correct predictions at every position along the river. These so called *governing equations* are typically PDEs that take the following form:

$$f(t,x) = u(t,x) + \mathcal{N}[u] \tag{2.37}$$

where $u_t$ is a latent solution $u(t,x)$ to a problem and $\mathcal{N}[\cdot,\lambda]$ is a nonlinear differentiable operator that is parameterized by parameters $\lambda$. In case of a physics-informed neural network, the functions $u(t,x)$ and $f(t,x)$ are modeled using neural networks. Both these networks share their learnable parameters. The authors of Raissi et al., 2019 give the one-dimensional *Burgers equation* to illustrate this. It is defined as:

$$\mathcal{N}[u,\lambda] = \lambda_1 u \frac{\partial u}{\partial x} - \lambda_2 \frac{\partial u}{\partial x} \tag{2.38}$$

We can now model the function $u(t,x)$ as a neural network. In order to compute the differentiable operator, automatic differentiation is used. Automatic differentiation (AD) is implemented in common Deep Learning frameworks like PyTorch (Paszke et al., 2019) or JAX (Bradbury et al., 2018). Using automatic differentiation, the partial derivative of $u(t,x)$ can be computed and $\mathcal{N}$ can be constructed. This is possible due to the fact that neural networks are differentiable, universal function approximators. The parameters $\lambda$ are fixed when we are only interested in a solution for the function $u(t,x)$. It is, however, also possible to let the neural network predict these parameters which the authors term *data-driven discovery* of partial differential equations.

The loss term for training physics-informed neural networks usually follows the form:

$$\text{MSE} = \text{MSE}_u + \text{MSE}_f \tag{2.39}$$

where $\text{MSE}_u$ is the mean squared error between predictions and the initial and boundary training data. This loss minimizes the distance between the predictions of the network and the known measurements. It is defined as:

$$\text{MSE}_u = \frac{1}{N_u} \sum_{i=1}^{N_n} \left[ u(t_u^i, x_u^i) - u^i \right]^2 \tag{2.40}$$

where $u^i$ is the value of $u(t,x)$ at $t_u^i$ and $x_u^i$. The second term of equation 2.39 is the $\text{MSE}_f$ and it enforces the physical structure by minimizing the error of the residual of the partial differential equation. It is thus defined as:

$$\text{MSE}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left[ f(t_f^i, x_f^i) \right]^2 \tag{2.41}$$

where $t_f^i$ and $x_f^i$ are the conditions at collocation points. These collocation points are randomly sampled in the domain of the function. However, we do not require training samples $u^i$ at these points. This allows us to enforce the underlying physics of the process that has produced the data at many more points than we actually have data for. This is what enables physics-informed neural networks to not overfit even if they are trained on very small or sparse datasets.

### 2.2.2. Reduced Order Models

Reduced order models are a method to accelerate the computation of large dynamical systems. There are many ways to construct reduced order models, including purely data driven neural networks and approaches that use physics-informed neural networks (Padula et al., 2024). This section focuses on the Galerkin projection technique to construct reduced order models, while Deep Learning based techniques are covered in section 2.2.3.

A core assumption of reduced order modelling is that the result of the high fidelity model lies on a low dimensional solution manifold (Hesthaven et al., 2016). The Galerkin projection specifies a vector field by projecting the dynamical system to a subspace $S$ (Holmes et al., 2012, eq. 4.3):

$$F_s = PF(u) \tag{2.42}$$

where $F(u)$ is nonlinear operator and $P$ is the projection onto subspace $S$. The following system of $n$ number of ordinary differential equations (ODE) then describes the reduced order model (Holmes et al., 2012, eg. 4.7):

$$\frac{\partial a(t)_j}{\partial t} = \langle F(u(t)) \, , \phi_j \rangle \qquad\qquad j = 1, ..., n \tag{2.43}$$

where $u(t)$ is a linear combination of time-dependent coefficients $a(t)_j$ and the orthonormal basis $\phi_j$ that spans the space $S$ (Holmes et al., 2012). Solving the system of ODEs in equation 2.43 gives a reduced order model that can approximate the behaviour of the original model.

A common way of finding a suitable basis is to use the *proper orthonormal decomposition* on the solutions of the high fidelity model. For this, the parameter space is sampled and the corresponding solutions are computed. The orthonormal basis functions are then given by the eigenfunctions of the sampled solutions. For an $N$-order reduced model, only the eigenfunctions corresponding to the $N$ largest eigenvalues are used. In order to ensure a reduced basis with sufficient accuracy, a large number of samples may be required which is sometimes infeasible in practise. Alternatively, the basis can be computed using

the greedy basis generation algorithm (Hesthaven et al., 2016).

Reduced order models are used to for example to evaluate the aeorelastic performance of aircrafts, as part of optimization algorithms to discover parameters, and when performing uncertainty quantification using Monte Carlo sampling. All these use cases require a large number of model evaluations and thus benefit from the computationally more efficient reduced order models (Benner et al., 2015).

### 2.2.3. Deep Learning for Physical Sciences

With the amazing progress in computer vision, natural language processing, and other areas, Deep Learning has been increasingly used to replace traditional numerical models. Large neural networks are able to learn non-linear patterns in data and can be used to build surrogate models for much larger numerical models. These surrogate models can either be purely *data-driven*, meaning they are trained on synthetic simulation data or real-world data, or be physics-based and include the underlying governing equations as a regularization term. Both of these setups will be discussed in the following chapter. For a more in-depth review of Deep Learning and fluid simulation please refer to the surveys H. Wang et al., 2024 and Brunton et al., 2020.

#### Data-Driven Surrogate Models

Accelerating fluid simulation using Deep Learning has been of interest to the scientific community for a long time. For example, Takbiri-Borujeni et al. (2020) trained a Convolutional Neural Network (CNN) on image data of simulations. CNNs are a standard model architecture for solving tasks in computer vision. The model learns a feature hierarchy on images by repeatedly convolving an image with a learned filter (Krizhevsky et al., 2012). While this results in local features and equivariant activation maps, the architecture can sometimes struggle to learn global dependencies in images. For generative models the convolutional network can be augmented with an attention mechanism (H. Zhang et al., 2019). A different architecture, the Vision Transformer (ViT) was proposed to address this weakness (Dosovitskiy et al., 2021a) for discriminative models.

Similarly, Lee and You, 2019 used generative adversarial neural networks (GAN) to generate fluid flows. They additionally augment their training loss using a physically motivated regularization term. GANs are a popular deep learning architecture to generate images, audio, and other modalities (Brock et al., 2019; Karras et al., 2018; Kong et al., 2020). The original GAN formulation was proposed as a generator trying to fool a discriminator trying to discrimi-

nate between a real and fake samples, which is why the architecture is termed *adversarial* (Goodfellow et al., 2014). The training process of GANs, however, is very unstable because neither the generator nor the discriminator may get too strong. Follow up work to the original GAN later proposed the discriminator (renamed to critic) to instead estimate the Wasserstein distance between the data and generated distributions (Arjovsky et al., 2017; Gulrajani et al., 2017). In this formulation, a strong critic only provides a better approximation of the distance, and thus the training becomes more stable.

Another type of surrogate model was introduced by Sanchez-Gonzalez et al. (2020) that is trained on particle-based fluid simulation data where message passing neural networks are utilized to learn the simulation. The message passing framework (Gilmer et al., 2017) formulates learning on graphs in two phases. First, messages are passed on the graph for a number of $T$ timestamps during which hidden states at each node are updated. At each timestamp, the features are updated using a vertex update function. In the readout phase, the last hidden state at timestamp $T$ is transformed by the readout function.

One of the many use cases for Deep Learning for physical sciences that has seen an increased interest recently is accelerating weather prediction. Currently, state-of-the-art weather prediction is done using numerical weather prediction (NWP). These systems aim to predict the weather globally by solving the underlying partial differential equations of the weather system. Deep Learning models trained on historical weather data, however, can provide competitive results while being orders of magnitude faster. Pathak et al. (2022) use a special Transformer architecture to learn a global weather forecast or climate model. Very similarly, Bi et al. (2023) propose a 3D-earth specific Transformer that is derived from the Swin Transformer architecture (Liu et al., 2021) used in computer vision. Another surrogate model for weather prediction is GraphCast (Lam et al., 2023) which uses graph neural networks to learn a global weather prediction. All of these models were trained on the ERA5 dataset with 39 years worth of historical weather data.

#### Deep Learning Models with Physical Prior Knowledge

In addition to a purely data-driven approach, some models aim to include the underlying physical model in the model or training process. For example, Bai et al. (2020) uses a physics-informed neural network (PINN) (see section 2.2.1) to estimate fluid flow. The authors used the Boltzmann equation and Navier-Stokes equations as constraints. NSFnets (Jin et al., 2021) similarly uses a PINN that is constrained using the Navier-Stokes equations. The model uses spatial and temporal coordinates as input and predicts velocity and pressure fields. Recent work has also experimented with using different model architectures like using

CNNs to learn surrogates for the shallow water equations (Donnelly et al., 2024).

All those models use a training-time regularization term to include physical prior knowledge into the model. Another option however is to include the prior knowledge in the structure of the model itself. R. Wang et al. (2021) propose a network architecture that encodes symmetry constraints for space, time, uniform motion, rotation, reflection, and scaling. Follow up work (R. Wang et al., 2022) suggests that hard constraints for symmetry might not always be desirable and proposes a relaxed constraint formulation as real-world data might not fulfill the constraint exactly. They propose a model that is biased towards a symmetric solution but not constrained to it.

#### Surrogate Models for Flood Prediction

The work presented in this thesis is most similar to the model architecture presented in Feng et al. (2023). The architecture used in their work also does not include special structural priors to enforce symmetry or similar constraints on the prediction, but includes physical prior knowledge through a regularization term. Unlike other works related to flood prediction (Shi et al., 2023), the model presented is not based on a recurrent neural network architecture. Instead, the network will model the problem as an implicit function.

## 2.3. Summary

This chapter provided an overview of the background and related work. Firstly, an overview of unsteady flow analysis was provided and the Saint-Venant equations were introduced, which enforce the continuity and momentum principles. The continuity equation enforces the principle of conservation of mass while the momentum principle states that the change of momentum and momentum flux is controlled by the forces acting on the control volume. These equations act as the governing equations that are solved by software like HEC-RAS to simulate water flow. The workflow of modelers using HEC-RAS to perform river analysis was also presented.

Furthermore, the chapter discussed how the geometry of the riverbed is oftentimes modeled. This includes the shape of the riverbed as well as estimating the roughness, which depends on the conditions of the riverbed and can be described by that Manning's $n$ coefficient.

Because river analysis is very computationally intensive, reduced order or surrogate models are commonly deployed in settings where many model evaluations are required. These models trade model fidelity for compute efficiency.

The chapter gave an overview of reduced order models as well as techniques using neural networks. There, methods only learning from data and methods encoding prior physical knowledge using physics-informed neural networks were explored. Physics-informed neural networks use the residual of the governing equations as a regularization term during the training process. A range of examples of surrogate models in the fields of fluid dynamics and weather simulation was given.

# 3. Requirements and Concept

This chapter outlines the requirements of the project and gives an overview of the solution. It first summarizes the motivation and problem statement, after which requirements are outlined. Then, an overview of the architecture is presented.

## 3.1. Motivation

Recent flooding events (Helmore, 2024; Henley, 2024) have demonstrated the importance of flood prevention and preparedness. The aim of this thesis is to study how surrogate models can be constructed for simulation data obtained using HEC-RAS.

As discussed in the section 2.2.3, Deep Learning is able to provide weather forecasts with comparable accuracy as traditional numerical weather prediction at a fraction of the time. This is accomplished by training large neural networks on massive amounts of historical data. Towards the goal of similar models for river analysis, building a surrogate model on simulation data is an important initial step.

Unlike for numerical weather prediction, however, less data is available for river flow. Thus, regularization techniques like physics-informed neural networks become more important. These techniques are studied on synthetic data to assess how faithfully the neural network model can reproduce the flow data.

## 3.2. Problem Statement

As outlined in section 2.1.1, river simulation using numerical models is the predominant method for estimating fluid flow. The underlying governing equations are used to describe the physics used to model the behaviour of the river. Software like HEC-RAS is used to solve those partial differential equations.

Furthermore, as outlined in section 2.2.2, surrogate models are lightweight models constructed from data produced by their high-fidelity counterparts. Deep Learning excels at learning complex, non-linear patterns from large

amounts of data. Recently, physics-informed neural networks (see section 2.2.1) have been proposed as a mechanism to provide physics-based prior knowledge to neural networks.

The aim of this project is to evaluate the feasibility of training physics-informed neural networks on synthetic data provided by HEC-RAS to build surrogate models.

## 3.3. Requirements

The following section outlines the requirements of the project. This covers both scientific and engineering requirements.

**Model and Evaluation.** The primary aim of this thesis is to develop a set of physics-informed neural networks that can faithfully represent simulation data produced by HEC-RAS. The design of this model should be motivated by contemporary methods used to solve similar problems. The design decisions should then be evaluated both qualitatively and quantitatively. The trade-offs between different model hyperparameters and model configurations should be explored, to provide an overview of the impact of different elements of the design. The results obtained in this work should further future research in this area.

**Data Representation.** The data used is stored in non-standard file formats used by HEC-RAS. A pipeline should therefore be developed to merge the data from multiple sources into standard representations amendable to machine learning workflows.

**Modularity and Reusability.** To aid the requirements outlined above, the system should be designed to be as modular as possible. Thus, the model architecture should not be designed as a monolith, but be able to be adopted to different scenarios evaluated. Moreover, the dataset should be stored in a data format that makes it reusable for future research.

## 3.4. Conceptual Components

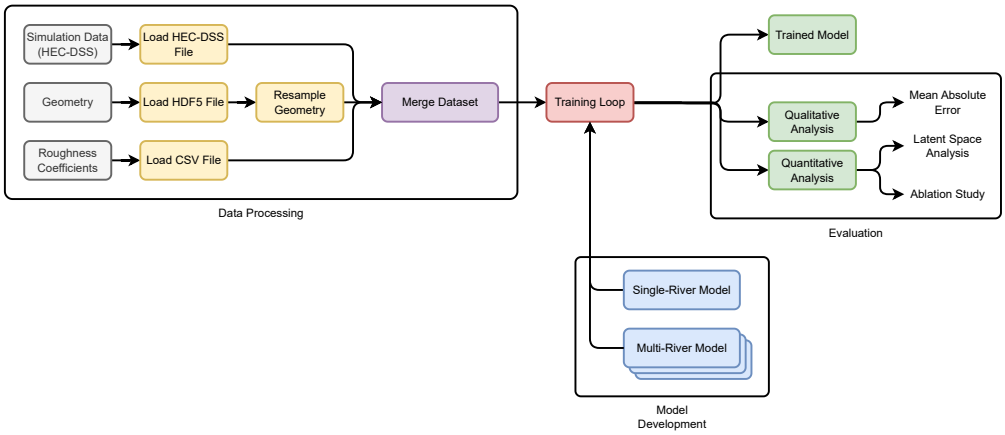



Figure 3.1.: Conceptual Components

Figure 3.1 depicts the conceptual components of the system. The inputs to the system are the simulation data acquired using HEC-RAS, the geometry including the Manning's $n$ coefficients, and the outputs of the system are the trained models including evaluation metrics. The different conceptual components are discussed in more detail below.

**Data Preprocessing.** The data used to train the network is stored in a variety of different file formats, and comes from different data sources. In general, the data consists of the simulation data produced by HEC-RAS and the corresponding project data. The simulation data is stored as time series using HEC-DSS, a proprietary file format used by HEC-RAS. The project data contains information about the river geometry, which includes the shape of the riverbed and the Manning's $n$ coefficients used to describe the roughness. As part of the data preprocessing, the geometry gets resampled so that every cross-section uses the same number of points. Then, all the data sources are merged into one a single dataset. This component implements the data representation requirement, and can be reused for different experiments.

**Model Development.** A model is developed to map the day of the year and river mile to the water surface height. This kind of design lends itself very well to the physics-informed regularization that is used as part of the training loop. The model can be configured to only represent a single or multiple rivers. In case of the multi-river model, the model is then augmented with a geometry encoder, for which two different architectures are developed. In total, four different models were constructed and trained. This design is proposed to satisfy the Modularity and Reusability requirement. The model development

conceptual component is part of the requirement to develop and evaluate a physics-informed neural network.

**Training Loop.** The training loop component uses the preprocessed data from the Data Processing component and the configured model from the Model Development component to train the model. During the training process, the model is regularized using an underlying physical model (In this case, the Saint-Venant equations described 2.1.1 are used). This component outputs the trained model.

**Evaluation.** Finally, the trained model is evaluated in the evaluation conceptual component. This component performs a quantitative and a qualitative analysis of the trained model, implementing the evaluation requirement. For the quantitative analysis, the results of the model are measured against the simulation data produced by HEC-RAS. For the qualitative analysis, an ablation study is performed and the latent space learned by the geometry encoder is visualized. The visualizations produced are discussed in chapter 5.

## 3.5. Summary

This chapter provided an overview of the requirements and concept of the system. As flood prevention becomes ever more important, Deep Learning based surrogate models can help accelerate river analysis. In this thesis, a surrogate model based on simulation data is presented.

For the system, the following requirements are specified: A model should be trained and evaluated, data should be represented in a unified format, and the entire system should be developed to be modular and reusable way to aid future research. This is achieved by dividing the system into distinct conceptual components; these cover merging and preprocessing the data acquired from different sources, constructing different model variants, training, and finally evaluating the model. For the evaluation, the error of the model prediction is measured against the simulation data and an ablation study is conducted. The output of the system are a trained model, and the results of the evaluation.

The next chapter will discuss the data preprocessing, design of the model architecture, and training loop in greater detail. Chapter 5 will cover the evaluation.

# 4. Methodology and Development

This chapter will introduce the architecture and training scheme of the surrogate model. The aim of the surrogate model is to approximate the results of HEC-RAS for the water depth. It is trained on a dataset of simulations of different rivers within the continental United States.

This chapter introduces two Deep Learning models: A single-river model that is trained for each individual river and a multi-river model that is trained on all rivers at once. The single-river model demonstrates how an implicit neural representation can be learned for the water depth of the river given time $t$ and river mile $x$. The multi-river model then extends the single-river model with a learned geometry encoder, and can thus be used on multiple rivers without retraining.

Lastly, the training scheme used for both models, including the physics-informed regularization, as well as details about the model development are explained.

## 4.1. System Overview

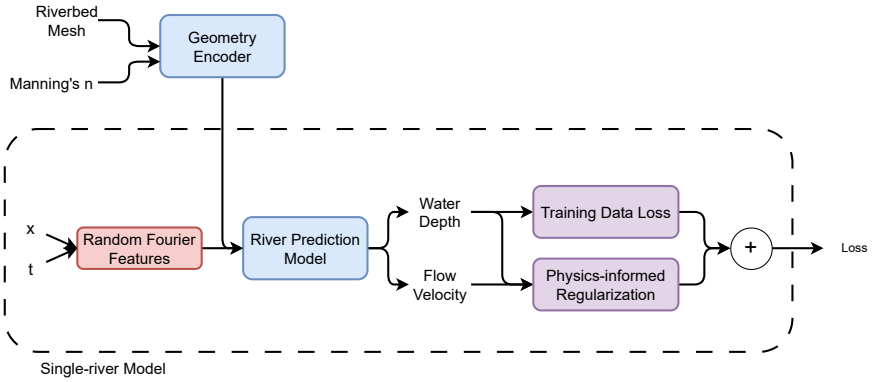



Figure 4.1.: System Overview

Figure 4.1 depicts the overall architecture of the system. The aim of the system is to predict the water depth at a given river mile $x$ and point in time $t$. For this,

the two input values are first encoded using a random Fourier feature encoding, and are then passed to the river prediction model. The river prediction model is used to predict the water depth and flow velocity. The training loss consists of two components: Firstly, the training data loss which is minimized for the predicted water depth against the ground-truth depth computed by HEC-RAS; and secondly, a physics-informed regularization, which uses both the water depth and flow velocity. The design of this architecture is similar to the architecture presented in Feng et al., 2023.

The multi-river model inherits the structure of the single-river model and conditions it on features learned by a geometry encoder. This encoder model learns a global shape descriptor from the 3D mesh of the riverbed and Manning's coefficient.

## 4.2. Single-River Model

This section describes the single-river model. The aim of this model is to estimate water depth $d$ for a single river as an implicit function over time. The model additionally predicts the flow velocity $v$, as it is required for calculating residuals for the physics-informed regularization. The goal thus is to learn the continuous function $f(x,t) \rightarrow [v,d]$. As the dataset contains a single value per day, $t$ represents the day of the year.

Implicit neural representations like this are a popular method for representing 3D surfaces for computer vision tasks. For example, DeepSDF (Park et al., 2019) and DISN (Xu et al., 2019) learn the signed distance between the surface and a given coordinate in space. Perhaps most prominently, Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020) use a neural implicit representation to estimate the radiance for a position in space and viewing angle. By learning to render different projections of a volume, the model implicitly learns the underlying geometry of a scene. Similarly, the single-river model implicitly learns the geometry of the river (the shape of the riverbed and roughness), and is then able to predict the water depth at a novel position $x$ and time $t$.

The following sections will now go into more detail about the neural network architecture used for the model.

### 4.2.1. Model Architecture

The 2D input coordinates are first transformed into a high dimensional embedding space using a coordinate encoding (see section 4.2.2 for details). This

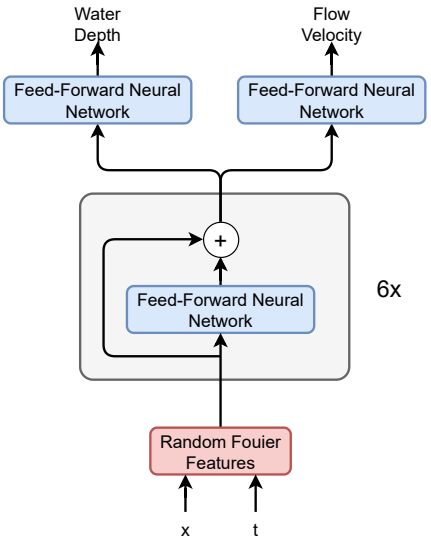



Figure 4.2.: Structure of the single river model

embedding space is then subsequently transformed by a series of residual blocks. This latent space is later transformed back using two separate feed-forward networks for the water depth and flow velocity. A ReLU activation function is used as part of these networks in the final layer to ensure the values predicted are always positive. Figure 4.2 depicts the structure of the single-river model.

An important feature of the architecture are residual connections. It can be observed that vanilla deep neural networks become saturated when increasing their network depth. If this is the case, the training and test errors of a deep neural network are higher than of a shallower counterpart. It should, however, always be possible to construct a network with an error that is at least as low as the one of a shallow network by learning an identity function in the last layers (He et al., 2015). Residual neural networks therefore group the model into a series of so-called residual blocks, each of which learns a residual function $f(x) + x$. With this structure, it is easier to learn the identity function, because $f(x)$ only has to map its inputs to 0. Residual connections are used in most state of the art architectures (Dosovitskiy et al., 2021b; Liu et al., 2022; Vaswani et al., 2017).

The residual blocks use fully connected layers to learn a transformation from embedding space to a higher dimensional latent space and back. Inside the blocks, leaky ReLU is used as the activation function. The model consists of

6 blocks, where each block has 512 hidden dimensions, and an 256 dimensional embedding space. The model has a total of around 1,662,978 trainable parameters.

### 4.2.2. Coordinate Encoding

Neural networks exhibit a bias towards low frequencies (Basri et al., 2020; Rahaman et al., 2019). In experiments on synthetic data it can be observed that during the training process, lower frequencies are fitted first regardless of their amplitude. It is a major reason why over-parameterized neural networks are still able to generalize beyond the training data.

For neural networks to learn high frequency functions, models like NeRFs (Mildenhall et al., 2020) use random Fourier features to overcome this low frequency bias:

$$\gamma(v) = [\cos(2\pi Bv), \sin(2\pi Bv)]^T \tag{4.1}$$

where $B$ is a matrix of random Fourier base frequencies (Basri et al., 2020).

These features can also be viewed as a generalization of positional embedding used in Transformer models (Vaswani et al., 2017). As a result, the network gains a hyperparameter that controls the convergence behaviour of the network. By tuning the standard deviation $\sigma$ of $B$ the trade-off between over-fitting and under-fitting can be controlled. Section 5.2.2 provides an ablation study that demonstrates the effect of this.

## 4.3. Multi-River Model

While the single river model implicitly encodes the geometry of the river by learning to predict the water depth, it is necessary to train a new model for every river. The multi-river model extends the single-river one with a geometry encoder to explicitly encode the geometry of the river bed.

The multi-river model learns a function $f(x, t, M) \rightarrow [v, d]$ where $x$, and $t$ are the river mile and time as in the single-river counterpart, and $M$ is a mesh of the riverbed constructed from the cross sections using in HEC-RAS. The mesh is encoded using a learned geometry encoder that maps the mesh $M$ to a single global feature vector. This feature vector is added to the coordinates $[x, t]$ after they have been transformed to latent space. The generation procedure for this mesh is discussed in more detail in section 4.4.1.

### 4.3.1. Geometry Encoder

HEC-RAS uses a simplified representation of the riverbed geometry, which was introduced in section 2.1.3. In the surrogate model, this coarse approximation is replaced with a learned geometry encoder that encodes the shape of the cross-sections and their corresponding Manning's $n$ values. The single-river model of the previous section is then conditioned using this global representation.

For the geometry encoder, two popular architectures for learning representations on geometry were evaluated. The first one, which is heavily inspired by the PointNet (Qi, Su, et al., 2017) architecture encodes a point cloud of the riverbed. The second architecture, a graph convolutional network, treats the mesh as a graph. The following sections will describe each of these architectures in greater detail.

#### Geometry Encoder Baseline

As a simple baseline, an embedding layer is used to learn a set of latent codes for the rivers. The baseline encoder allows the model to represent multiple rivers and is able to generalize to cross-sections that were not part of the training dataset. It, however, does not learn a representation that is aware of the geometry of the riverbed.

### Point Cloud Encoder

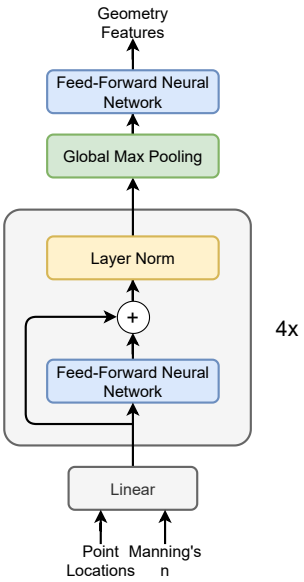



Figure 4.3.: Point cloud geometry encoder

Point clouds are a popular shape representation, where a surface is represented by a dense, unordered collection of points. The architecture of the point cloud geometry encoder follows the principles of the PointNet architecture (Qi, Su, et al., 2017) as well as follow-up work (Qi, Yi, et al., 2017).

The core idea of PointNet is to learn features using a point-wise multilayer perceptron (MLP), which means its weights are shared between the points. These features are then aggregated using a symmetric function. In order to learn rotation invariant features on the points, the input points are transformed to a canonical orientation by a learned rotation matrix (The rotation matrix is learned by the so-called *T-Net*, which is a smaller version of the architecture). This design ensures that the entire model is permutation invariant, and scales well to a large number of points. The symmetric function used in PointNet is global max pooling. PointNet++ (Qi, Yi, et al., 2017) extends that architecture by applying PointNet to multiple local regions of the point cloud. It can be viewed as similar to the structure of CNNs that also aim to learn a hierarchy of local filters to describe images.

The point cloud encoder model is organized in 4 residual blocks to learn point-wise features on the point cloud. Layer Normalization (Ba et al., 2016)

is used in the encoder to stabilize the training. The network does not include the *T-Net* because it is only trained on the meshes of riverbeds and thus does not need to be rotation invariant. The multi-scale features proposed in PointNet++ are also not used in this model, as they increase the resources required for training while not significantly improving model performance. Figure 4.3 depicts the structure of the model.

The point cloud encoder is trained on point clouds with $3,500$ points that are uniformly sampled from the meshes. The features at each point are its $(x, y, z)$ coordinates and a vector of the Manning's $n$ values.

### Graph Convolutional Network

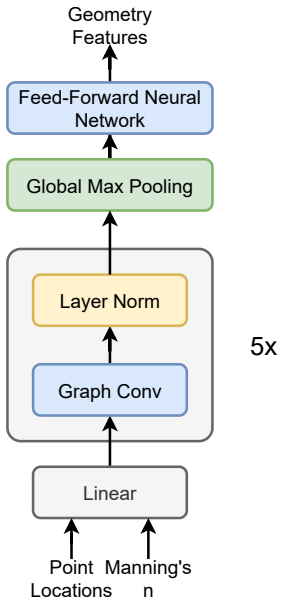



Figure 4.4.: Graph convolutional geometry encoder

Graph Convolutions aim to generalize the concept of convolutions to graphs. For each node, the features of neighboring nodes get accumulated, and a learned function is applied. In fact, standard convolution on images can be seen as a special case of graph convolution where the pixels are nodes in a graph connected to a regular grid (Wu et al., 2021; S. Zhang et al., 2019).

Graph Convolutional Networks are popular for learning features on meshes, as meshes can be readily interpreted as graphs. For example, Pixel2Mesh and follow up work (N. Wang et al., 2018; Wen et al., 2019) reconstruct 3D meshes from images by deforming a template mesh using graph convolutions. Another very similar approach is MeshCNN (Hanocka et al., 2019), which uses local surface features and so-called edge pooling to learn representations on triangular meshes.

The graph convolutional geometry encoder uses a simplified convolutional architecture. The convolutional layers are used to learn local features on the mesh surface. The model consists of 5 graph convolutional layers, each followed by layer normalization and leaky ReLU activation functions. It, however, does not include any pooling layers; similar to the point cloud encoder, the features are then aggregated using a global maximum pooling operation. This global feature vector is then transformed by a feature extractor consisting of multiple dense layers. The full model is depicted in figure 4.4.

The graph convolutional geometry encoder is trained directly on the meshes of the riverbed. Similarly to the points sampled for the point cloud encoder, the vertices' features are their $(x, y, z)$ position and a vector of Manning's $n$ values.

## 4.4. Training

The following section outlines the training procedure for both the single- and multi-river model. First, the dataset and the different training and testing splits are outlined. Then, the loss term and physics-informed regularization are discussed.

### 4.4.1. Data Preprocessing

| **Feature** | **Description** |
|---|---|
| t | Normalized timestamp of the sample |
| x | Normalized position of the cross-section |
| Mannings' n | The Mannings'n coefficient for the cross-section (see section 2.1.2) |
| Geometry | A list of points that describe the geometry of the cross-section |
| stage | Water surface height predicted by HEC-RAS |

Table 4.1.: Features of a sample in the dataset

Both the single-river and the multi-river models were trained on simulated output data from HEC-RAS. The data was produced according to the workflow

outlined in section 2.1.4. A total of 3,240 stations from 63 rivers in the United States was used. The stations along the river are not placed in a fixed distance, but are placed by river modelers to achieve the highest fidelity simulation with the least amount of cross-sections. On average, the cross-sections are 0.74 miles apart. For each station, one data point per day was provided, resulting in a total of 1,182,600 samples. Table 4.1 depicts the features of a sample.

For each river, the river mile $x$ and time $t$ are standardized to a range of $[-1, 1]$. Furthermore, the slope and level of the riverbed are calculated from each of the stations using the cross-section geometry.

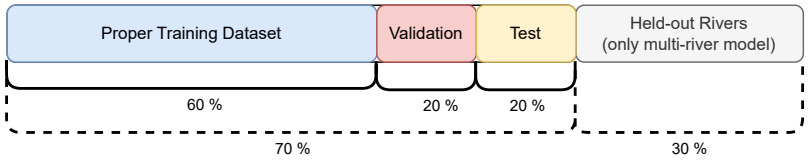



Figure 4.5.: Splits of training dataset

The dataset is split in two ways: From the dataset used to train the multi-river model 30% of river sections (reaches) are withheld in order to test generalization to novel sections. The remaining dataset is then split into the usual 60% proper training, 20% validation and 20% test datasets. Thus, for every river that is not withheld in the first step, 60% of the stations become part of the *proper training dataset*. With this setup the model can be evaluated for how well it performs on novel river stations of known rivers and novel rivers all together. Figure 4.5 illustrates the different dataset splits. When training a single-river model, the stations of one river are split into 60% training, 20% validation and 20% test sets.

### Riverbed Geometry

HEC-RAS only provides the geometry of the 2D cross-sections as a list of points and a list of three or more Manning's $n$ coefficients. In order to encode the river using the geometry encoders, these individual sections are combined to a single mesh that models the entire river.

The number of points and Manning's $n$ values varies from cross-section to cross-section; as a first step, the lists of points are resampled with a constant number of 50 vertices and the Manning's $n$ coefficients are padded to a 16-dimensional vector. The 3D vertex positions of the mesh are then constructed by combining the $x$ and $y$ coordinates of the resampled points of the cross-sections with the river mile as the $z$ coordinate. A natural limitation of this approach is

that the resulting river geometry does not model the real-world curvature of the river.

### 4.4.2. Supervised Training on Simulation Data

The loss function for the training is defined as

$$l(\hat{y}, y_{\text{stage}}) = (\hat{y}_{\text{stage}} - y_{\text{stage}})^2 + \lambda \times \text{residuals}(\hat{y}_{\text{depth}}, \hat{y}_{\text{flow}}) \tag{4.2}$$

where $\hat{y}_{\text{depth}}$ and $\hat{y}_{\text{flow}}$ are the predicted water depth and flow respectively. The predicted stage $\hat{y}_{\text{stage}}$ is defined as $\hat{y}_{\text{stage}} = \hat{y}_{\text{depth}} + z_0$, where $z_0$ is the lowest level of the riverbed. The weight $\lambda$ is a tunable hyperparameter and is set to $\lambda = 0.1$.

The first term in the loss function directly penalizes the mean squared error of the predictions of the neural network against the stage predicted by HEC-RAS. As we only aim to predict levee overtopping, only the stage is penalized as part of the loss function. The second term is the physics-based regularization that penalizes predictions that do not follow the governing equations.

### 4.4.3. Physics-Informed Regularization

In addition to supervising on the simulated data, the model is also regularized to adhere to the underlying physical model. The residuals of the Saint-Venant equations that were derived in section 2.1.1 are used. The residual of the continuity equation is defined as:

$$\frac{\partial d}{\partial t} + v\frac{\partial d}{\partial x} = 0 \tag{4.3}$$

where $t$ is the day of the year, $x$ is the river mile, and $d$ and $v$ refer to the water depth and flow velocity, respectively. The total amount of water is assumed to stay constant. The residual of the momentum equation is defined as:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + g\frac{\partial d}{\partial x} + g(S_f - S_0) = 0 \tag{4.4}$$

where $g$ is the gravity, $S_0$ the slope, and $S_f$ calculated according to the Chézy-Manning equation (see equation 2.26):

$$S_f = \frac{4 \times 2^{\frac{2}{3}} n^2 v^2}{D_H^{4/3}} \tag{4.5}$$

Note that these equations are slightly different from the ones introduced in chapter 2. When assuming the channel to be a simple rectangle, the Saint-Venant

equations can be simplified for ease of implementation. The multi-river model still has access to the full riverbed geometry through the geometry encoder and does not need to rely on this coarse approximation used in the regularization term. This formulation has also been used for other physics-informed neural networks (Feng et al., 2023). The derivation of the equations 4.3 and 4.4 can be found in appendix A.

The full regularization term is the sum of the residuals weighted by a hyperparameter $\lambda = 0.1$. The partial derivatives like $\frac{\partial d}{\partial t}$ that appear in the residuals are computed using automatic differentiation in PyTorch. This leverages the fact that neural networks are differentiable, universal function approximators. For a more detailed introduction to physics-informed neural networks please refer to section 2.2.1.

### 4.4.4. Hyperparameters

| **Setting** | **Geometry Encoder** | **# Hidden Dims** | **# Params** | **# Epochs** |
|---|---|---|---|---|
| Single-River | - | 256 | 1,662,978 | 50 |
| Multi-River | Baseline | 128 | 852,802 | 25 |
| | Point Cloud | 128 | 1,305,538 | 25 |
| | Graph Convolutional | 128 | 1,357,570 | 25 |

Table 4.2.: Hyperparameter Configurations

Table 4.2 depicts the hyperparameters used for the different training configurations. The single-river model has slightly more parameters than the multi-river counterparts, as the single river model has more hidden dimensions. All models were trained using the Adam optimizer (Kingma & Ba, 2017) with a learning rate of 0.00005. The values for $\beta_1$ and $\beta_2$ were set to 0.9 and 0.99 respectively. These parameters were found using a hyperparameter search on the single-river model.

## 4.5. Development

| Tool | Usage |
|---|---|
| PyTorch | PyTorch is a Deep Learningn library that was used to implement the neural networks |
| PyTorch3D | PyTorch3D provides tools for geometric Deep Learning using PyTorch. It was used to implement the geometry encoders |
| `pydsstools` | A utility Python library to load the data stored by HEC-RAS |
| Pandas | Pandas provides tools for data analysis in Python |

Table 4.3.: Tools Overview

Table 4.3 provides an overview of the tools used for the development of the model. The source data is first loaded using `pydsstools`, then merged and transformed using Pandas, and finally a model developed using PyTorch and PyTorch3D is trained. The development of these components is described in greater detail in the following sections. The models were trained on a system with four A100 NVIDIA GPUs.

### 4.5.1. Data Format and Preprocessing

The model was trained on simulated data produced by HEC-RAS which stores flow and stage data using a custom file format called *HEC-DSS*. In addition, the geometry data is stored as part of the HEC-RAS project. During the data pre-processing step, the results and geometry data are written to a single Pandas[1] dataframe.

HEC-RAS offers options to view the files inside the software (USACE Hydrologic Engineering Center, 2024, Viewing Data Contained in an HEC-DSS File ), but for this project, the data was loaded using the `pydsstools` Python library[2]. The file format is conceptually organized like a container where each time series is stored at paths. The paths are built using the river, river reach, and station identifier. For each station, the flow and stage are fetched and stored with the path in a Pandas dataframe.

The geometry is stored as part of the project in seperate HDF5 files. The HDF5 files for the cross-section geometry for each river station is loaded as a `numpy` array and added as a column to the dataframe. The geometry data of the cross-sections is duplicated for each timestamp. This data duplication does trade ease of implementation for the size of the dataset on disk.

[1] https://pandas.pydata.org/
[2] https://github.com/gyanz/pydsstools

### 4.5.2. Model Development

Both the single-river and multi-river models were developed in PyTorch (Paszke et al., 2019). All models were implemented using the standard high-level API as PyTorch modules. The models were trained using the implementation of Adam (Kingma & Ba, 2017) provided by PyTorch. The gradients for the physics-informed regularization were computed using the `torch.autograd.grad` method.

A major benefit of using PyTorch is the dynamic nature of the library. Instead of storing the entire mesh of a river section for each training example, only an index is stored in the dataframe. During model training, the model then looks up the correct mesh and passes it to the geometry encoder. This is implemented using a `MeshStorage` class that holds a dictionary of river identifiers and corresponding meshes. Another benefit of this design is that for the baseline geometry encoder the river identifier can just be used directly.

### 4.5.3. Geometry Representation

For the geometry representation the *PyTorch3D* library was used. PyTorch3D provides a wide range of tools to implement common model architectures that work in 3D. A core component of the library is the `Meshes` data structure which implements variable-sized packing. Most datatypes used in Deep Learning have a constant size and can be easily batched as multi-dimensional arrays. For example, images always have the same pixel size regardless of content. Meshes, however, vary in the number of vertices and faces and therefore cannot be batched in the same way. PyTorch3D thus stores the vertices and faces in a single, large array with indices that store where each mesh in the batch begins (Ravi et al., 2020). A similar technique called sequence packing is used in large language models to handle sentences with different lengths efficiently (Krell et al., 2022).

## 4.6. Summary

This chapter introduced the surrogate models and training scheme used to approximate the simulation data provided by HEC-RAS. Firstly, the single-river model was presented. It learns an implicit neural representation of a single river, and is able to predict the water depth at novel locations and points in time for the river it has been trained on. A crucial aspect is transforming the input values with random Fourier features which helps to mitigate the spectral bias of neural networks.

Next, the multi-river model was presented. It used the same structure as the single-river counterpart, except that it conditions the latent space using

a learned geometry encoder. Two architectures, a point cloud encoder and a graph convolutional encoder, were discussed for the encoder model. The point cloud encoder uniformly samples a point cloud of the mesh and learns point-wise features, that are subsequently aggregated using a global pooling operation. The graph convolutional encoder works similarly, but the features are learned in a local neighbourhood of the mesh.

Afterwards, the training scheme was detailed. The model is trained using a data term and a physics-informed regularization term. The regularization term is used to penalize the model for predictions that do not conform to the underlying physical model. The physical model used were the Saint-Venant equations which were presented in chapter 2.

Lastly, the chapter described the development of the river models. Both models were developed using the PyTorch Deep Learning framework, and PyTorch3D was additionally used for the geometry encoder models. The data to train the models are stored as HEC-DSS files in case of the simulation results, while the project files are stored as HDF5 files. The different data sources are merged and preprocessed using Pandas.

# 5. Evaluation

This chapter evaluates the performance of both the single-river and multi-river models. First, a quantitative analysis of the single-river model is provided. Then, an ablation study is conducted to investigate the impact of the random Fourier features and physics-informed regularization on the model. Next, the multi-river model the performance of the multi-river model is analyzed and the latent space of the geometry encoder is visualized. Lastly, the results are interpreted.

## 5.1. Results Overview

| **Setting** | **Geometry Encoder** | **Test Error** | **Held-out Error** |
|---|---|---|---|
| Single-River | - | 9.716 ft (2.96m) | - |
| Multi-River | Baseline Encoder | 8.661 ft (2.64m) | 18.860 ft (5.74m) |
| | Point Cloud Encoder | 8.678 ft (2.64m) | 24.497 ft (7.46m) |
| | Graph Convolutional | 8.342 ft (2.54m) | 18.663 ft (5.68m) |

Table 5.1.: Overview of the results

Table 5.1 gives an overview of the results of the model configurations. For the multi-river setting, model architectures using a point-cloud based encoder (see 4.3.1), and graph convolutional encoder (see 4.3.1) were evaluated. Additionally, results for the baseline encoder introduced in section 4.3.1 are presented. Please refer to section 4.4.4 for more details on the hyperparameters used for training.

The error values presented in the table are calculated using the means absolute error. As outlined in section 4.4.1, the dataset was split in a proper training, test, and validation dataset. In addition, some river sections (also called reaches) were held back to evaluate generalization performance. The used hyperparameters are detailed in 4.4.4

The following section will present the results for the single-river and multi-river models in detail.

## 5.2. Single-River Model

This sections discusses the evaluation of the single-river model. First, the stage prediction results are discussed. Then, an ablation study for the physics-informed regularization and random Fourier features is performed.

### 5.2.1. Stage Prediction Results

Figure 5.2 depicts the mean absolute error of the stage for the single-river model. Each of the box plots shows the distribution of the error over all the cross-sections for one river. The model architecture and training hyperparameters (e.g. number of blocks, hidden dimensions, learning rate, regularization etc.) have been tuned on part of the Arkansas River. The results on different rivers therefore also shows how well the *architecture* generalizes to different rivers.

The mean absolute error achieved on average by all models is 9.716 feet (2.96 meters). It can be observed that there is a variance in the error within the rivers (i.e. the slices of the simulation) as well as between the rivers. For example, the error of the Arkansas River is far above the average error achieved. Conversely, it is possible to model some of the rivers with far better accuracy (e.g. Tensas River).

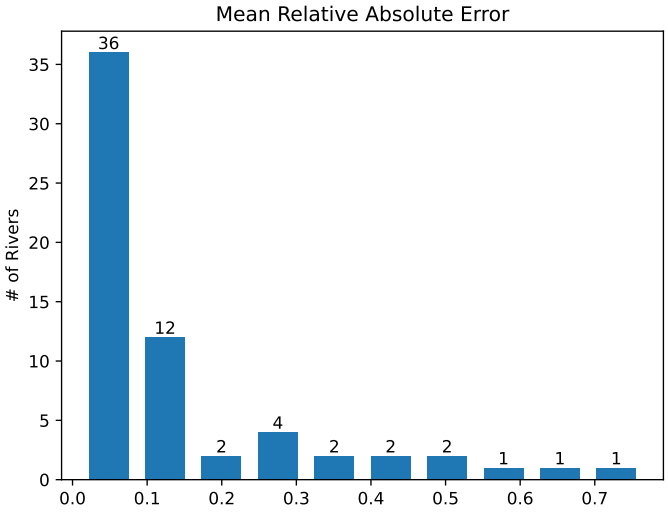



Figure 5.1.: Histogram of the relative error scores of the river

To better asses the quality of the stage predictions on the rivers the mean relative absolute error (MRAE) is computed. The MRAE is calculated by deviding

the mean absolute error of the stage by the average ground truth stage:

$$\frac{\frac{1}{N}\sum_{i=1}^{N}(|\hat{y}_i - y_i|)}{\frac{1}{N}\sum_{i=1}^{N} y_i} \tag{5.1}$$

where $\hat{y}$ is the predicted stage, $y$ is the ground truth. Figure 5.1 depicts a histogram of the mean absolute relative error for each of the single river models. For the vast majority of rivers, the relative error is close to 0. This indicates that the model architecture is able to make accurate predictions on a many different rivers.

### 5.2.2. Ablation Study

We aim to explore the effect of individual components of the single-river model and the training mechanism. For this, an ablation study is presented (see figure 5.3), where the model is trained without the physics-informed regularization and Fourier features.

The base model depicted in figure 5.3a shows that it is possible to predict a coarse curve without random Fourier features or physics-informed regularization. The low frequency bias of the model acts like a regularizer, which prevents overfitting. As shown in figure 5.3b, random Fourier features let the network also learn the high frequency features of the data. However now the curve does not follow the ground truth data as well as before. To remedy this, the physics-informed regularization term is introduced as part of the training process of the model. The model is now able to model both the high frequency features as well as the global shape of the curve.

The following sections will discuss the effects of the parameters for the regularization term and random Fourier features in greater detail.

#### Physics-Informed Regularization

The physics-informed regularization can aid fitting the model by smoothing out intermediate values (see section 4.4.3). Experiments were conducted to empirically demonstrate the effect of the regularization. The single-river model was fit on the simulation data of the the Arkansas River with weights 0.01, 0.1, 1.0, as well as no regularization at all. The value for $\sigma$, which controls the random Fourier features, was fixed at $\sigma = 4$. The results of these experiments are depicted in Figure 5.4.

It can be seen that without the regularization (top right), the model does capture many fine details of the curve, but fails to capture the overall shape well.

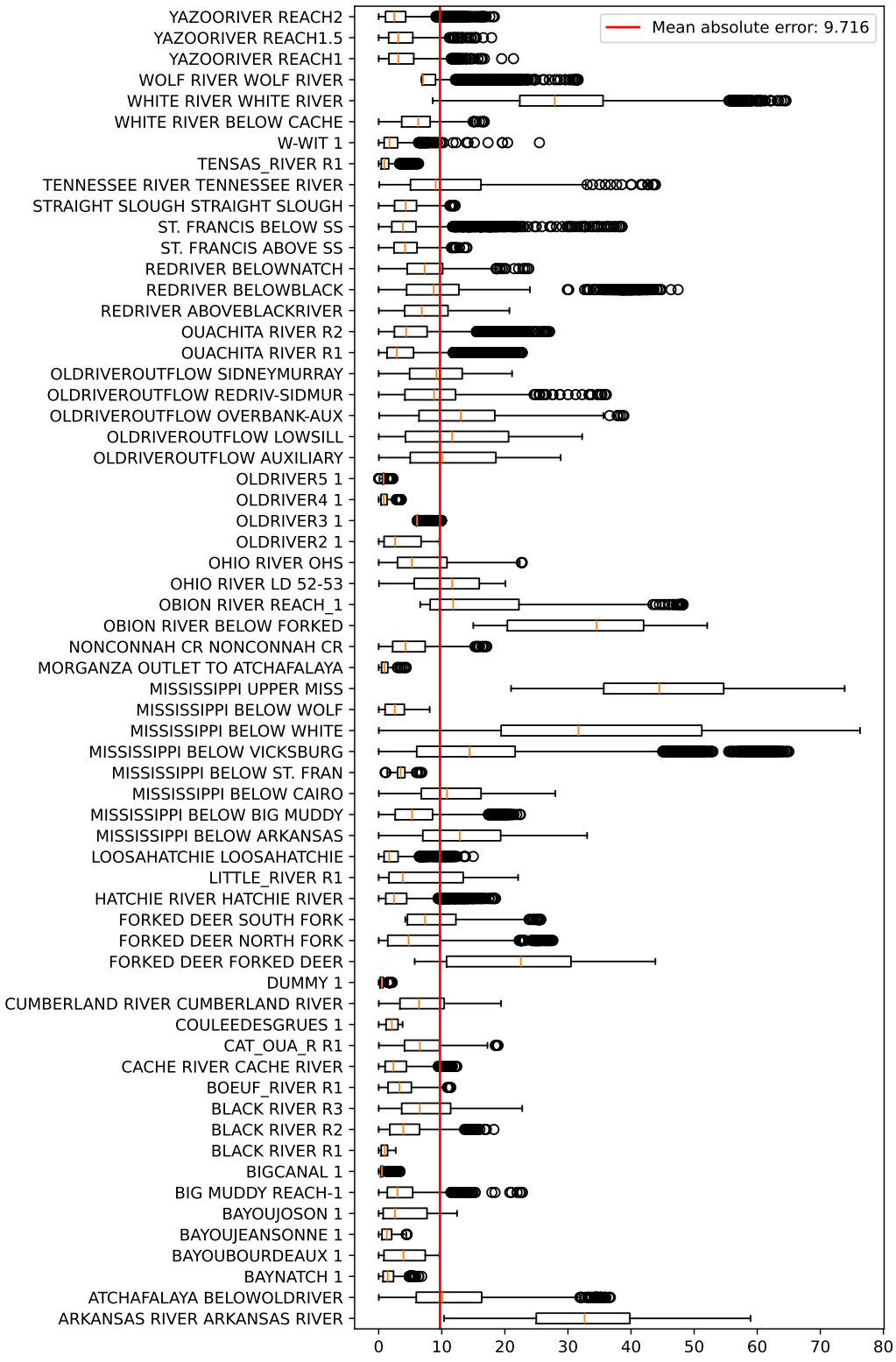
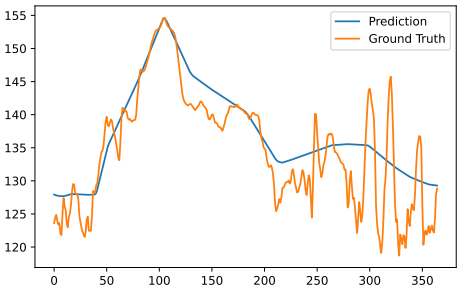
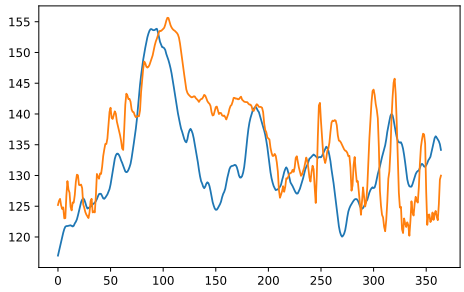


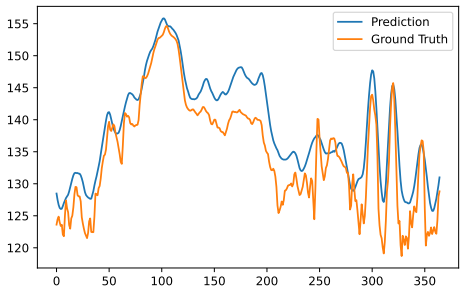

Figure 5.2.: Mean absolute error of each river predicted by the single-river model.




a) Base Model




b) With random Fourier features




c) With physics informed reg. and rand Fourier features

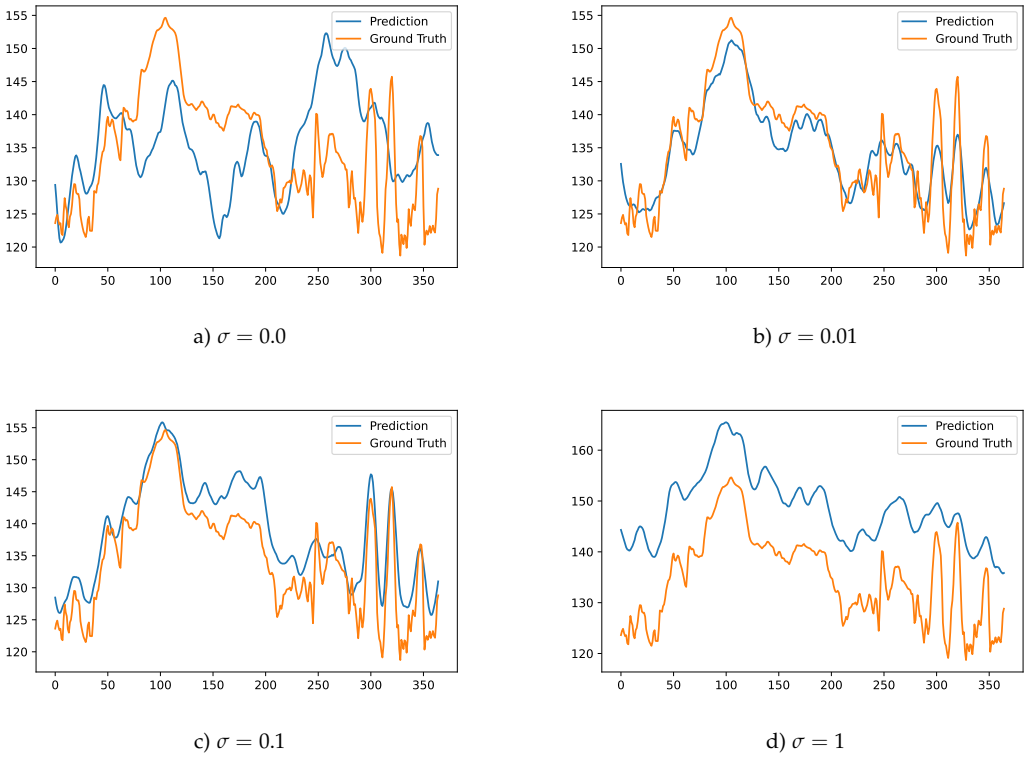
Figure 5.3.: Overview of the ablation study

When using the weights 0.001 and 0.1, the simulated water surface height is captured well by the surrogate model. The last plot depicts an over-regularized model. It can be observed how the curve becomes too smooth and loses some of the higher frequency features. Using a hyperparameter search, 0.15 was found as a good value for the weight of the regularization.

### Random Fourier Features

Next, we demonstrate the impact of the Fourier feature encoding, which was introduced in section 4.2.2, on the model predictions. For this, the standard deviation $\sigma$ of the normal distribution used to sample the random Fourier features is altered while the weight of the physics-informed regularization is fixed at $\lambda = 0.1$.

Figure 5.5 depicts the results of the experiments. When replacing the random Fourier features with a learned linear transformation (see figure 5.5a), low frequency bias becomes visible. The model is not able to fit the higher frequencies of the data. As shown in figures 5.5b and 5.5c how a higher value for $\sigma$ helps the model to learn those high-frequency features. However, if the value of $\sigma$




a) $\sigma = 0.0$ b) $\sigma = 0.01$

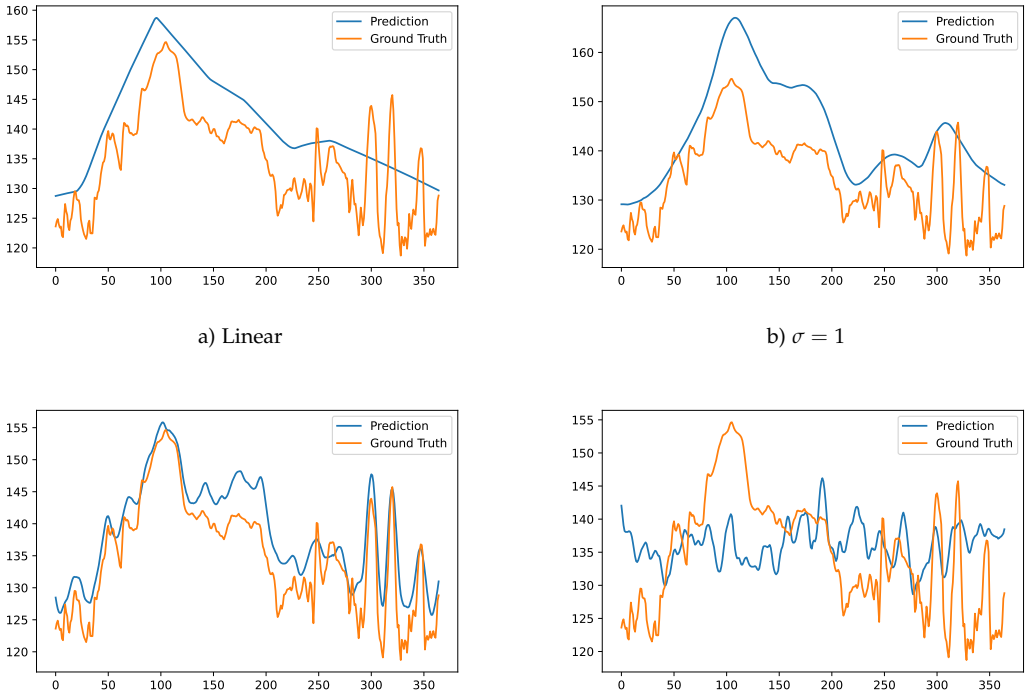
c) $\sigma = 0.1$ d) $\sigma = 1$

Figure 5.4.: Predictions on models with different amount of physics-informed regularization. Without it, the model struggles to capture the global shape of the simulation data.

grows too large (see figure 5.5d) the model is not able to learn the global shape of the data.

## 5.3. Multi-River Model

This section presents the results for the multi-river models. Like for the single-river model the stage prediction results are discussed, and then compared to the errors achieved using the single-river models. Then, the embedding space learned by the geometry encoder is discussed.




a) Linear

b) $\sigma = 1$

c) $\sigma = 4$

d) $\sigma = 10$

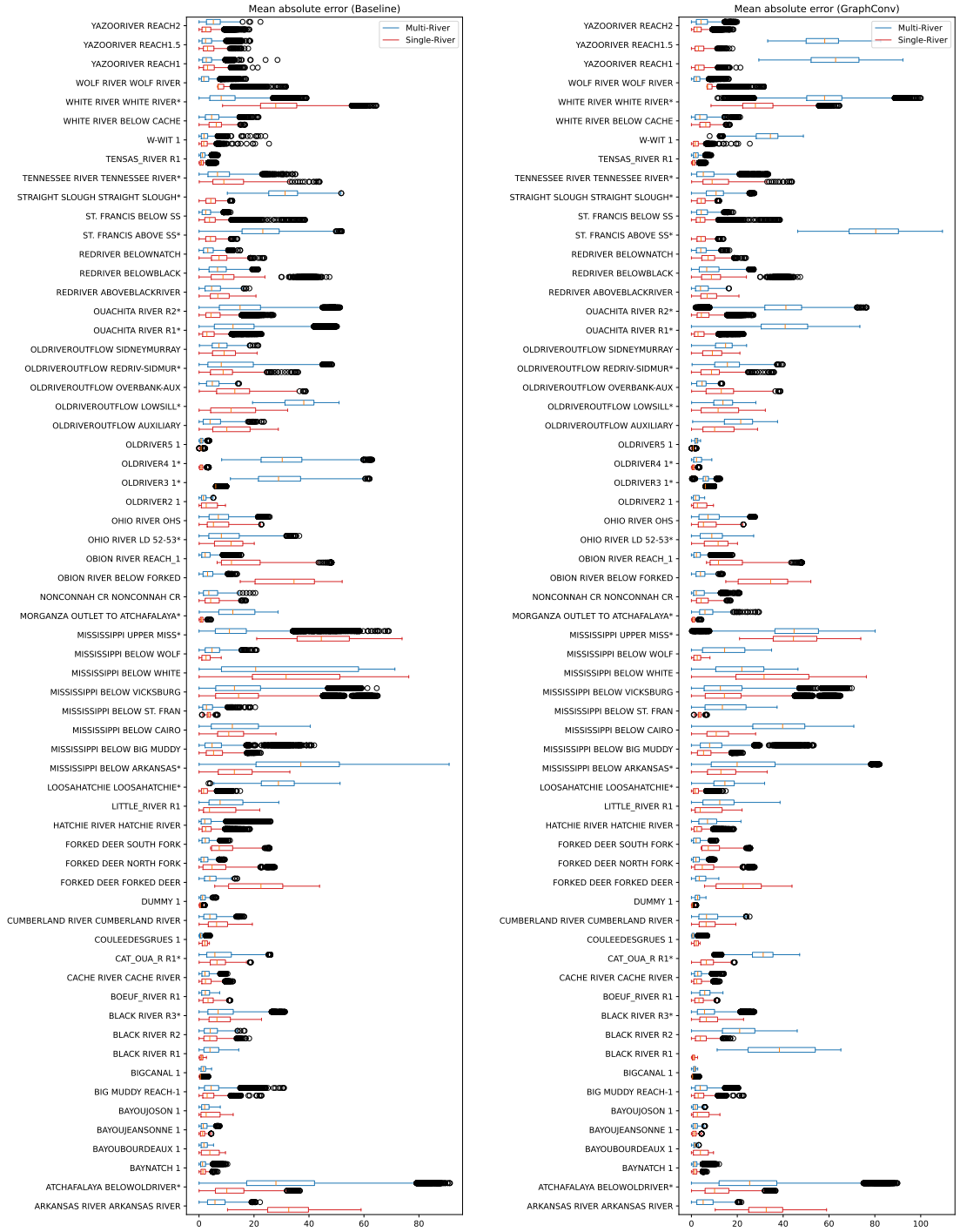
Figure 5.5.: Model predictions with different random Fourier features. When the value for the standard deviation of the random features is larger, the model learns more high-frequency features.

### 5.3.1. Stage Prediction Results

| Geometry Encoder | Test error | Held-out rivers error | Test error (relative) | Held-out rivers error (relative) |
|---|---|---|---|---|
| Baseline encoder | 8.661 ft (2.64m) | 18.860 ft (5.74m) | 0.204 | 0.351 |
| Point cloud encoder | 8.678 ft (2.64m) | 24.497 ft (7.46m) | 0.272 | 0.406 |
| GraphConv encoder | 8.342 ft (2.54m) | 18.663 ft (5.68m) | 0.233 | 0.322 |

Table 5.2.: Error of the multi-river model on the test- and held-out datasets

Table 5.2 outlines the results of the multi-river model. The model is evaluated on the test dataset and the held-out dataset. The test dataset contains novel stations on known river sections, while the held-out dataset contains river sections that the model was not trained on. The mean absolute error and mean




a) Baseline Encoder

b) Graph Convolution Encoder

Figure 5.6.: Mean absolute errors of the multi-river models and single-river models. Rivers marked with asterisks are part of the held-out dataset.

relative absolute error (see equation 5.1) for both datasets are calculated. From the table it can be observed that all configurations of the multi-river model outperform the single-river model architecture which achieved a mean absolute error of 9.716 ft (2.961m) on the test dataset. This is true even when using the baseline encoder which is not geometry-aware.

The choice of geometry encoder does at first glance not appear to significantly impact the model performance. On average, the multi-river model with the graph convolutional encoder does achieve roughly the same error as the baseline encoder. This and that the observed errors on the held-out dataset are higher than on the test dataset indicate that the geometry encoder overfits on the limited number of rivers. This effect persists when using the relative test error which corrects for the different sizes of the rivers. However, as depicted in figure 5.6, the graph convolutional encoder is still preferable over the baseline encoder because the distribution of errors on each river is closer to the one of the single-river model.

Figure 5.6 compares the mean absolute error of the multi-river and single-river models on each river. River sections marked with an asterisks are part of the held-out dataset. When looking at river sections of rivers in the test dataset (e.g., Yazoo river) the baseline encoder tends to outperform the graph convolutional counterpart. The same is true for rivers where all sections are part of the held-out dataset (e.g., Outachita river). Here it can be observed that both models struggle to generalize to novel rivers while the graph convolutional encoder slightly outperforms the baseline. An interesting case are rivers where some sections were part of the held-out dataset (e.g., Old river or Black river): Here the geometry-aware graph convolutional encoder tends to outperform the baseline model. The river sections of a single river naturally are very similar, which means that an encoder that is aware of the geometry is able to learn very similar latent codes. This, however, crucially does not imply that the model is able to generalize to entirely novel rivers. It only means that the encoder is capable of *storing* a discrete set of river geometries that are then used to condition the single-river model.

### 5.3.2. Visualization of Geometry Embedding Space

Figure 5.7 depicts a visualization of the latent space produced by the graph convolutional geometry encoder. To produce this visualization, each river was encoded using the geometry encoder and the resulting feature vector was reduced using T-SNE (van der Maaten & Hinton, 2008).

From the visualization it can be seen that the geometry encoder is able to learn a meaningful embedding space. Different river sections belonging to the

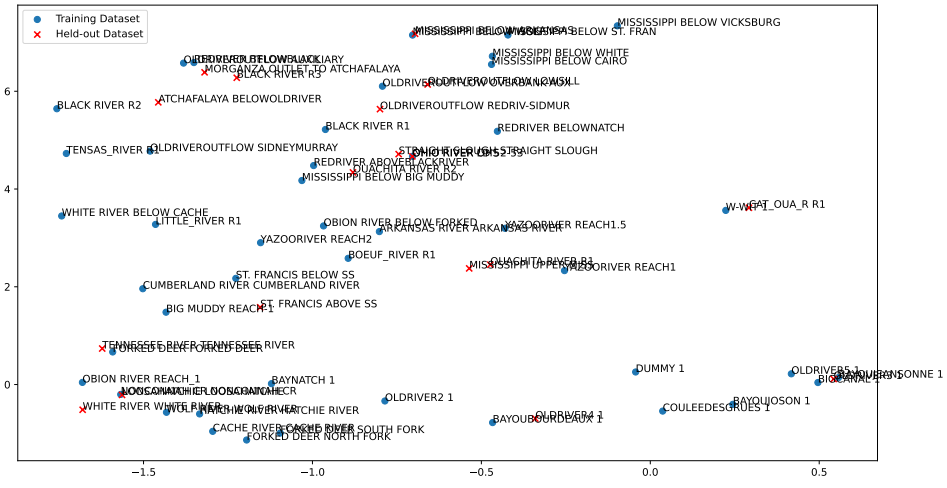



Figure 5.7.: Visualization of the latent space of the graph convolutional geometry encoder

same river tend to be clustered together in the visualization (e.g., the cluster of the sections belonging to the Mississippi river). Thus, they also map to very similar embedding vectors which can explain the behaviour discussed in section 5.3.1. As discussed there, the geometry encoder is more likely learning a mapping of river sections to a discrete set of river clusters than the actual underlying dynamic of the governing equation. The results of the river encoder is discussed in more detail in section 5.4.2.

## 5.4. Findings and Discussion

In the following section the results presented will be discussed. It first covers design considerations for the river model and discusses limitations. Further, the results of the geometry encoder are discussed in more detail.

### 5.4.1. River Model Design Considerations

The design of the model architecture has a significant impact on the model performance. This applies to both the single-river as well as the multi-river model, as the multi-river model inherits the model structure from the single-river one.

First of all, the model is trained to fit an implicit function that maps coordinates ($t$ and $x$) to predictions. The major benefit of this approach is that it does allow the most flexibility for the placement of cross-sections. An alternative design could have treated the time series akin to image data and have used a

convolutional neural network or denoising Transformer architecture (Devlin et al., 2019; Lewis et al., 2020) for the model. However, this would require a dense collection of river stations. While it might be possible to obtain those for simulated data, it is hardly possible for real-life flow data.

Random Fourier features are an essential component of the architecture that allows the model to learn high-frequency components of the curve. As demonstrated in the ablation study, the model only learns a very coarse approximation of the curve. Random Fourier features are a common architectural feature of coordinate-based generative models like neural radiance fields (Mildenhall et al., 2020). The experimental results presented in this chapter validate the results presented in literature which is predominantly focused on the computer vision domain. In fact, one can argue that the model treats the surrogate modelling problem as modelling the implicit geometry of the water surface elevation, and therefore the river, over time. This model paradigm has been successfully applied in computer vision, and provides a flexible architecture for building neural surrogate models.

Physics-informed regularization plays an equally crucial part in the design of the model. Regularization techniques like weight decay are common in Machine Learning to avoid overfitting. When training neural surrogate models, the physics-informed regularization is crucial to enforce the dynamics of the system at parts where no training data is available. Here, the advantage of learning an implicit function also becomes clear. When learning a continuous function, we can make predictions, and regularize them, at any continuous coordinate location.

However, the model architectures presented also have limitations. Most importantly, both the single-river and multi-river models exhibit a significant test error. This high error, for now, means that these models should not be used to replace numerical simulations performed in HEC-RAS. A likely reason for this is that the surrogate model is only capable of modelling parts of the HEC-RAS simulation. The surrogate model presented does not cover structures such as dams, lakes, pumps, and other parts of the river network that HEC-RAS takes into account. This is especially noteworthy since water depth predictions are sometimes wrong a near-constant offset when compared to the ground truth simulation data.

Another limitation is generalizability of the model. While both the single-river and multi-river model are able to generalize to novel station locations within known rivers, the multi-river model fails to generalize to novel rivers. As analyzed in section 5.3.1, the geometry encoder remembers the geometry but does not generalize beyond the training dataset. The geometry encode is

discussed in more detail in section 5.4.2 below.

In addition, the model is only capable of generalizing to values of $x$ and $t$ that lie within the range of $[-1, 1]$ the model was originally trained on. This limitation arises naturally from the problem setting: A dataset that would span multiple years would also have to contain multiple instances of the training example $(x, t) \rightarrow [v, d]$ (Assuming $x$ is the day of the year; if this were not the case, the model would not have access to more useful patterns to learn from, but just a longer series). The way to disentangle these examples is by conditioning the model on the actual weather conditions at or around the river station. Unfortunately, the HEC-RAS dataset does not contain this kind of weather data.

### 5.4.2. Analysis of the Geometry Encoder

The results indicate that while the multi-river model is able to outperform the single-river models on average, the graph convolutional geometry encoder does not outperform the baseline encoder. A likely explanation for the failure of the geometry encoder is that the proper training dataset only contains 63 distinct river sections. Overfitting on this very small number of rivers is thus a very likely problem.

Another explanation for the observed behavior may point to a deeper failure of neural networks to learn arithmetic operations. The aim of the geometry encoder is to provide a surrogate for the approximation of the hydraulic radius and the roughness coefficient. These two parameters then influence the flow velocity and stage of the river. There has been research that shows that neural networks struggle to learn to learn arithmetic operations and generalize beyond the range they were trained on. Special network architectures are used to solve this problem (Trask et al., 2018).

If the geometry encoder indeed fails to generalize because neural networks have an inductive bias against it, a specialized network architecture might be necessary. However, a larger and more diverse river geometry dataset may also compensate for the missing inductive bias, and allow the model to learn an effective geometry encoder. If such a dataset is not available, another option is to either use available approximation of the riverbed presented in section 2.1.3 or use an alternative geometry representation (Flanagin et al., 2007) to formulate a descriptor. This is similar to approaches in computer vision like histograms of orientated gradients that compute global descriptors for natural images. These descriptors are then used in models like support vector machines to classify images.

### 5.4.3. Limitations

While it is possible to learn a surrogate model to approximate the simulation data generated using HEC-RAS, the prediction accuracy and generalization capabilities of the model are limited. This section summarizes the limitations of the method in more detail.

**Prediction Accuracy.** While the model is able to predict the general trend of the simulation data, the evaluation has shown that the accuracy of the model is limited. The best model achieved a mean absolute error of 8.342 ft (2.54m), which may not be enough to replace the original HEC-RAS model. A major reason for this is that the surrogate model only covers a subset of the functionality that HEC-RAS provides. Given that the model is also only trained on simulation data, it inherits the accuracy of HEC-RAS itself as a ceiling. Real-world historical datasets are most likely much larger than simulated ones.

**Generalization.** While the multi-river model is shown to generalize to unseen cross-sections, it is unable to generalize to novel rivers altogether. For an open set of rivers, which would require a global model, the method may not be a good fit. However, if the set of rivers does not change between training and inference, the method can provide approximations of the data.

**Data Availability.** The underlying reason driving the limited prediction accuracy and generalization is the size of the dataset. The models were trained on synthetic data acquired using HEC-RAS. Unfortunately, generating a larger dataset would require a larger river network which was not available for this project.

## 5.5. Summary

This chapter presented an evaluation of the models introduced in the prior chapter. First, the single-river models were evaluated, which achieved an average mean absolute error of 9.716 ft (2.96m). The ablation study showed that both the random Fourier features and the physics-informed regularization play an important role to achieve good results. When modifying the standard deviation $\sigma$ for the Fourier features and regularization weight $\lambda$, the effect of both becomes visible. If the random Fourier features are not used, the model is not able to represent the high-frequency details of the data. When the model is trained without the regularization, the global shape of the curve is not modeled well.

Afterwards, the multi-river model was evaluated. All configurations of the

multi-river model achieved better results than the single-river models achieved on average. Of the multi-river models, the model configured with the graph convolutional geometry encoder performed best with an mean absolute error of 8.342 ft (2.54m). However, the model does struggle to generalize to river sections not contained in the training dataset. When visualizing the latent space learned by the graph convolutional geometry encoder, it can be seen that the latent codes of sections from the same river tend to be clustered together.

The discussion section covered design configurations for the model itself, and discussed the generalization capabilities of the geometry encoder further. It is theorized that the geometry encoder does not generalize well because of the lack of diverse training data.

The next chapters will cover lessons learned and conclude the thesis.

# 6. Lessons Learned

This chapter outlines key lessons learned from conducting the research presented in this thesis.

## 6.1. Literature Research

Gaining a deep understanding of the research domain was a crucial initial step for this project. Deep Learning research has sometimes tended to disregard domain knowledge acquired by other fields. The most well known example of this probably is linguistics in the case of language modelling ("Language models and linguistic theories beyond words," 2023). On the other hand, training relatively generic model architectures like Transformers on very large datasets has led to many breakthroughs. In the field of environmental sciences, for example, the weather prediction models presented in section 2.2.3 use adapted versions of the Transformer architecture and graph neural networks.

For this project, however, it was important to incorporate the physics as prior information. A major aspect of the literature research therefore was understanding and summarizing the derivation of the Saint-Venant equations. Further, different forms of these equations are used in literature. The equations derived in hydraulics literature and the ones used in practice like in Feng et al., 2023 are slightly different. Thus, bridging the gap between the two formulations was an important aspect of the literature research. A deeper understanding of the physics behind river analysis also helped inform the discussion of the geometry encoder.

## 6.2. Development

Building on the literature research, a surrogate model was developed. Initially, some of time was spent trying to forecasting the stage from weather data using sequential models like Transformers. However, this approach unfortunately turned out to be infeasible due to the lack of high-quality data. Therefore, the scope of the project was limited to building a surrogate model for HEC-RAS. A significant influence of the early model design was the work of Feng et al., 2023 as it demonstrated a neural network optimized with a regularization term that

implemented the Saint-Venant equations. The models described in this work can be viewed as an extension of the model described by the authors.

Given how extensive the HEC-RAS software suite is, for a project like this it is essential to limit its scope in order to avoid mission creep. HEC-RAS not only covers the rivers themselves, but also dams, lakes, and other elements. All these influence the outcome of the simulation. Therefore, working with domain experts to define the scope of the project, and to find the most influential elements is critical. Further, for future projects it may be helpful to work with simplified simulation setups.

Another major challenge during the development of this project was that while the U.S. Army Corps of Engineers provides a good documentation for HEC-RAS itself, the file formats are less well-documented. For example, to develop the geometry encoder, the geometry of cross-sections needed to be extracted. The structure of the geometry files used to store the geometry is undocumented, which is why the files had to be reverse-engineered to work with the geometry.

Lastly, different Deep Learning frameworks have different strengths and weaknesses. For a project like this, PyTorch is preferable because of is dynamic nature. The initial experiments were developed using TensorFlow. While TensorFlow provides better performance than PyTorch in some instances, it also is very opinionated. The pythonic nature of PyTorch, on the other hand enables to implement the geometry storage mechanism which greatly reduces the amount of redundant data to be stored. Additionally, PyTorch provides the PyTorch3D library which includes many useful primitives for the geometry encoder.

## 6.3. Personal

As part of this master's thesis, I had the pleasure of spending half a year at the University of New Orleans in the United States. In addition to getting a taste of the culture of New Orleans, I was able to meet many new and exciting people, and make both personal and professional lasting connections. Before working on this master's thesis, my work was mostly focused on computer vision. Therefore, working on an entirely different research topic in a new environment greatly helped me to expand my horizon. It also helped me recognize the importance of adhering to a well-established and rigorous process when conducting high-quality research.

# 7. Conclusion and Future Work

This chapter concludes the project described in this thesis. It describes the main conclusions drawn and provides a possible road map for future work related to this project.

## 7.1. Conclusion

The aim of this thesis was to build and evaluate surrogate models for river analysis which were trained on synthetic data produced by HEC-RAS. As expected, neural networks are very well-suited for approximating the simulation data. Formulating the model as an implicit function makes it straightforward to handle the sparse nature of cross-section locations. It allows to easily regularize the model at locations where there are no training samples available. As described in the chapter 5, both the random Fourier features and the physics-informed regularization play a crucial role in learning a representation of the simulation data. The ablation study presented shows that without the Fourier features the model is only able to learn a very coarse representation of the data. While this acts as an implicit regularization of the model, it also limit its expressiveness. The model becomes more expressive when the random Fourier features are added, as the transformation helps mitigate the low frequency bias of neural networks. As the model now loses the natural regularization of the spectral bias, the physics-informed regularization helps to regularize the model and adhere to the underlying physical model. The combination of random Fourier features and physics-informed regularization results in a model that can represent high-frequency features, and adhere to the underlying physical model. This does confirm results from literature that has studied the spectral bias of neural networks.

For the surrogate model two scenarios, a model trained on a single river and one trained on multiple rivers, were studied. For the multi-river models, different model architectures to encode the geometry of the riverbed were evaluated. The single-river model achieved an average mean absolute error of 9.716 ft (2.96 m). Every configuration of the multi-river model outperforms the error achieved by the single-river model, with the configuration with the graph convolutional encoder achieving an mean absolute error of 8.342 ft (2.54 m). Even when using the baseline geometry encoder, which is not aware of the river geometry, the

model is able to benefit from the larger training dataset (MAE 8.661 ft or 2.64 m).

However, the method still has significant limitations. The surrogate models presented in this thesis do still exhibit a significant error when compared to the ground truth simulation data. This may first of all be attributed to the necessary simplifications a surrogate model has to make. Many aspects of the simulation like bridges, dams, or bodies of water, are not taken into account by the surrogate model. These effects, however, still influence the flow, and therefore also the stage, which contributes to the error of the surrogate model. Secondly, it was observed that the river geometry encoder struggles to generalize, as the system was only trained on a small set of river sections. When training surrogate models for geographically limited areas, which do not have to generalize to rivers outside the training dataset, the baseline encoder may be preferable in some instances. However, for global surrogate models, a better geometry encoder trained on a larger, more diverse training dataset, will be necessary.

## 7.2. Future Work

We identify multiple potential areas of research for future work. Firstly, the currently model architecture could be scaled up as neural network performance tends to improve with larger model sizes. As scaling up the model will require more training data, additional data sources should be considered. Many government agencies around to world collect data on river flow. Collecting and consolidating those real-world data sources would be an important aspect of scaling up the surrogate models presented here. The models could then be trained on either only real-world data or a mixture of real-world data and simulation data. Training on a larger, more diverse training dataset may already mitigate the overfitting observed for the geometry encoder.

Another possible line of research would concern the estimation of the friction force (see chapter 2.1.2). While most forces acting on the fluid are physically motivated, the friction force is estimated using empirical formulas. The values for the Manning's $n$ coefficient have been determined from experimental data. Deep Learning, however, could use a combination of satellite images, elevation data, and other measurements to directly predict the friction force. A system like this could be trained in similar manner as the surrogate model of this project, except that the model would also have to predict the force. This is also known as partial differential equation discovery (Raissi et al., 2019). The estimation of the friction coefficient could be used in HEC-RAS itself, improving the predictions of the system, and saving modellers time. In this case, it may be beneficial to train models on simplified river environments that have been

simulated using HEC-RAS, before moving to real-world data.

And finally, all of these strains of research could lead to an end-to-end surrogate model that may even surpass the performance of HEC-RAS. For this, in addition to collecting river flow data and riverbed geometry, the data would need to be connected with corresponding weather data. For weather simulation (see section 2.2.3) models like this already exist, and similar models for river analysis could have an enormous impact for flood prevention, help respond to natural disasters, and drive foundational research.

# Bibliography


Arjovsky, M., Chintala, S., & Bottou, L. (2017, June). Wasserstein generative adversarial networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 214–223, Vol. 70). PMLR. https://proceedings.mlr.press/v70/arjovsky17a.html (cit. on p. 22).

Augustin, J., Andrees, V., Czerniejewski, A., Dallner, R., Schulz, C. M., & Mezger, N. C. S. (2024). Auswirkungen des ahrtal-hochwassers auf die gesundheit der lokalen bevölkerung – eine analyse auf grundlage von gkv-routinedaten. *Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz*, *67*(1), 5–13. https://doi.org/10.1007/s00103-023-03809-x (cit. on p. 1).

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. https://arxiv.org/abs/1607.06450 (cit. on p. 34).

Bai, X.-d., Wang, Y., & Zhang, W. (2020). Applying physics informed neural network for flow data assimilation. *Journal of Hydrodynamics*, *32*(6), 1050–1058. https://doi.org/10.1007/s42241-020-0077-2 (cit. on p. 22).

Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., & Kritchman, S. (2020, 13–18 Jul). Frequency bias in neural networks for input of non-uniform density. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (pp. 685–694, Vol. 119). PMLR. https://proceedings.mlr.press/v119/basri20a.html (cit. on p. 32).

Benner, P., Gugercin, S., & Willcox, K. (2015). A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, *57*(4), 483–531 (cit. on p. 21).

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., & Tian, Q. (2023). Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, *619*(7970), 533–538. https://doi.org/10.1038/s41586-023-06185-3 (cit. on p. 22).

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax (cit. on p. 19).

Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. *International Conference on Learning Representations*. https://openreview.net/forum?id=B1xsqj09Fm (cit. on p. 21).

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*(Volume 52, 2020), 477–508. https://doi.org/https://doi.org/10.1146/annurev-fluid-010719-060214 (cit. on p. 21).

Chanson, H. (2004). *Hydraulics of open channel flow*. Butterworth-Heinemann. (Cit. on pp. 6–13).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423 (cit. on p. 53).

Donnelly, J., Daneshkhah, A., & Abolfathi, S. (2024). Physics-informed neural networks as surrogate models of hydrodynamic simulators. *Science of The Total Environment*, *912*, 168814. https://doi.org/https://doi.org/10.1016/j.scitotenv.2023.168814 (cit. on p. 23).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021a). An image is worth 16x16 words: Transformers for image recognition at scale. https://arxiv.org/abs/2010.11929 (cit. on p. 21).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021b). An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR* (cit. on p. 31).

Feng, D., Tan, Z., & He, Q. (2023). Physics-informed neural networks of the saint-venant equations for downscaling a large-scale river model. *Water Resources Research*, *59*(2). https://doi.org/10.1029/2022wr033168 (cit. on pp. 6, 23, 30, 39, 57).

Flanagin, M., Grenotton, A., Ratcliff, J., Shaw, K. B., Sample, J., & Abdelguerfi, M. (2007). Hydraulic splines: A hybrid approach to modeling river channel geometries. *Computing in Science  Engineering*, *9*(5), 4–15. https://doi.org/10.1109/MCSE.2007.99 (cit. on p. 54).

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, June). Neural message passing for quantum chemistry. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 1263–1272, Vol. 70). PMLR. https://proceedings.mlr.press/v70/gilmer17a.html (cit. on p. 22).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 27). Curran

Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (cit. on p. 22).

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf (cit. on p. 22).

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., & Cohen-Or, D. (2019). Meshcnn: A network with an edge. *ACM Transactions on Graphics*, *38*(4), 1–12. https://doi.org/10.1145/3306346.3322959 (cit. on p. 36).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* (cit. on p. 31).

Helmore, E. (2024). Hurricane helene's 'historic flooding' made worse by global heating, fema says. *The Guardian*. https://www.theguardian.com/world/2024/sep/16/dramatic-flooding-in-central-europe-continues (cit. on p. 25).

Henley, J. (2024). Death toll reaches 16 as 'dramatic' flooding in central europe continues. *The Guardian*. Retrieved March 10, 2024, from https://www.theguardian.com/world/2024/sep/16/dramatic-flooding-in-central-europe-continues (cit. on pp. 1, 25).

Hesthaven, J. S., Rozza, G., & Stamm, B. (2016). *Certified reduced basis methods for parametrized partial differential equations* (Vol. 590). Springer. (Cit. on pp. 20, 21).

Holmes, P., Lumley, J. L., Berkooz, G., & Rowley, C. W. (2012). Galerkin projection. In *Turbulence, coherent structures, dynamical systems and symmetry* (pp. 106–129). Cambridge University Press. (Cit. on p. 20).

IPCC. (2022). Summary for policymakers [In Press]. In H. O. Pörtner, D. C. Roberts, M. Tignor, E. S. Poloczanska, K. Mintenbeck, A. Alegría, M. Craig, S. Langsdorf, S. Löschke, V. Möller, A. Okem, & B. Rama (Eds.), *Climate change 2022: Impacts, adaptation, and vulnerability. contribution of working group ii to the sixth assessment report of the intergovernmental panel on climate change* (In Press). Cambridge University Press. (Cit. on p. 1).

Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, *426*, 109951. https://doi.org/https://doi.org/10.1016/j.jcp.2020.109951 (cit. on p. 22).

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. *International Conference on Learning Representations*. https://openreview.net/forum?id=Hk99zCeAb (cit. on p. 21).

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980 (cit. on pp. 39, 41).

Kong, J., Kim, J., & Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 17022–17033, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/c5d736809766d46260d816d8dbc9eb44-Paper.pdf (cit. on p. 21).

Krell, M. M., Kosec, M., Perez, S. P., & Fitzgibbon, A. (2022). Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. https://arxiv.org/abs/2107.02027 (cit. on p. 41).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 21).

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. (2023). Learning skillful medium-range global weather forecasting. *Science*, *382*(6677), 1416–1421 (cit. on p. 22).

Language models and linguistic theories beyond words. (2023). *Nature Machine Intelligence*, *5*(7), 677–678. https://doi.org/10.1038/s42256-023-00703-8 (cit. on p. 57).

Lee, S., & You, D. (2019). Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, *879*, 217–254. https://doi.org/10.1017/jfm.2019.700 (cit. on p. 21).

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020, July). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7871–7880). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703 (cit. on p. 53).

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on p. 22).

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 31).

Malcherek, A. (2019). *Fließgewässer: Hydraulik, hydrologie, morphologie und wasserbau* (1st ed.). Springer Vieweg. in Springer Fachmedien Wiesbaden GmbH. (Cit. on pp. 12–15).

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV* (cit. on pp. 30, 32, 53).

Padula, G., Girfoglio, M., & Rozza, G. (2024). A brief review of reduced order models using intrusive and non-intrusive techniques. https://arxiv.org/abs/2406.00559 (cit. on p. 20).

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 165–174 (cit. on p. 30).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. https://arxiv.org/abs/1912.01703 (cit. on pp. 19, 41).

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., & Anandkumar, A. (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214* (cit. on p. 22).

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660 (cit. on pp. 33, 34).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* (cit. on p. 34).

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., & Courville, A. (2019). On the spectral bias of neural networks. *International conference on machine learning*, 5301–5310 (cit. on p. 32).

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045 (cit. on pp. 18, 19, 60).

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. https://arxiv.org/abs/2007.08501 (cit. on p. 41).

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. W. (2020). Learning to simulate complex physics with graph networks. *International Conference on Machine Learning* (cit. on p. 22).

Shi, J., Yin, Z., Myana, R., Ishtiaq, K., John, A., Obeysekera, J., Leon, A., & Narasimhan, G. (2023). Deep learning models for flood predictions in south florida. https://arxiv.org/abs/2306.15907 (cit. on p. 23).

Takbiri-Borujeni, A., Kazemi, H., & Nasrabadi, N. (2020). A data-driven surrogate to image-based flow simulations in porous media. *Computers Fluids*, *201*, 104475. https://doi.org/https://doi.org/10.1016/j.compfluid.2020.104475 (cit. on p. 21).

Te Chow, V. (1959). *Open-channel hydraulics*. McGraw-Hill. https://books.google.at/books?id=JG9_PwAACAAJ (cit. on p. 12).

Thieken, A. H., Bubeck, P., Heidenreich, A., von Keyserlingk, J., Dillenardt, L., & Otto, A. (2023). Performance of the flood warning system in germany in july 2021 – insights from affected residents. *Natural Hazards and Earth System Sciences*, *23*(2), 973–990. https://doi.org/10.5194/nhess-23-973-2023 (cit. on p. 1).

Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., & Blunsom, P. (2018). Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/0e64a7b00c83e3d22ce6b3acf2c582b6-Paper.pdf (cit. on p. 54).

USACE Hydrologic Engineering Center. (n.d.). Hec-ras hydraulic reference manual [Accessed: 2024-08-24]. (Cit. on p. 7).

USACE Hydrologic Engineering Center. (2024). Hec-ras user's manual [Accessed: 2024-08-29]. (Cit. on pp. 15–18, 40).

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html (cit. on p. 51).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 31, 32).

Wang, H., Cao, Y., Huang, Z., Liu, Y., Hu, P., Luo, X., Song, Z., Zhao, W., Liu, J., Sun, J., Zhang, S., Wei, L., Wang, Y., Wu, T., Ma, Z.-M., & Sun, Y. (2024). Recent advances on machine learning for computational fluid dynamics: A survey. https://arxiv.org/abs/2408.12171 (cit. on p. 21).

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., & Jiang, Y.-G. (2018). Pixel2mesh: Generating 3d mesh models from single rgb images. *ECCV* (cit. on p. 36).

Wang, R., Walters, R., & Yu, R. (2021). Incorporating symmetry into deep dynamics models for improved generalization. *International Conference*

*on Learning Representations*. https://openreview.net/forum?id=wta_8Hx2KD (cit. on p. 23).

Wang, R., Walters, R., & Yu, R. (2022). Approximately equivariant networks for imperfectly symmetric dynamics. *International Conference on Machine Learning* (cit. on p. 23).

Wen, C., Zhang, Y., Li, Z., & Fu, Y. (2019). Pixel2mesh++: Multi-view 3d mesh generation via deformation. *ICCV* (cit. on p. 36).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. https://doi.org/10.1109/tnnls.2020.2978386 (cit. on p. 35).

Xu, Q., Wang, W., Ceylan, D., Mech, R., & Neumann, U. (2019). Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 492–502). Curran Associates, Inc. http://papers.nips.cc/paper/8340-disn-deep-implicit-surface-network-for-high-quality-single-view-3d-reconstruction.pdf (cit. on p. 30).

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks. *International conference on machine learning*, 7354–7363 (cit. on p. 21).

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: A comprehensive review. *Computational Social Networks*, *6*(1), 11. https://doi.org/10.1186/s40649-019-0069-y (cit. on p. 35).

# Appendix

# Appendix A.

# Simplification of Saint-Venant Equations

The Saint-Venant equations (equations 2.16 and 2.19) presented can be simplified under the assumption that the channel is rectangular. This leads to the equations used in the regularization term in 4.4.3.

## Continuity Equation

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \qquad Q = vA, A = Wd \tag{A.1}$$

$$\frac{\partial Wd}{\partial t} + \frac{\partial vWd}{\partial x} = 0 \tag{A.2}$$

$$W\frac{\partial d}{\partial t} + Wv\frac{\partial d}{\partial x} = 0 \tag{A.3}$$

$$\frac{\partial d}{\partial t} + v\frac{\partial d}{\partial x} = 0 \tag{A.4}$$

# Momentum Equation

$$\frac{\partial vA}{\partial t} + \frac{\partial v^2 A}{\partial x} + gA\frac{\partial d}{\partial x} = gA(S_0 - S_f) \qquad Q = vA, A = Wd \tag{A.5}$$

$$\frac{\partial vA}{\partial t} + \frac{\partial v^2 A}{\partial x} + gA\frac{\partial d}{\partial x} = gAS_0 - gAS_f \tag{A.6}$$

$$\frac{\partial vA}{\partial t} + \frac{\partial v^2 A}{\partial x} + gA\frac{\partial d}{\partial x} - gAS_0 + gAS_f = 0 \tag{A.7}$$

$$\frac{\partial vA}{\partial t} + \frac{\partial v^2 A}{\partial x} + gA\frac{\partial d}{\partial x} + gA(S_f - S_0) = 0 \tag{A.8}$$

$$A\frac{\partial v}{\partial t} + A\frac{\partial v^2}{\partial x} + gA\frac{\partial d}{\partial x} + gA(S_f - S_0) = 0 \tag{A.9}$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + g\frac{\partial d}{\partial x} + g(S_f - S_0) = 0 \tag{A.10}$$