Physics-Informed Machine Learning for Modeling Unsaturated Soil Water Flow

Jule Grimm

Master's thesis
2026:E50

Lund University

Faculty of Science
Centre for Mathematical Sciences
Computational Science

Centrum Scientiarum Mathematicarum

Physics-Informed Machine Learning for Modeling Unsaturated Soil Water Flow

Jule Grimm

Thesis supervisors: Adrian Gustafson, Linda Hartman

Date: June 24, 2026

Abstract

Unsaturated soil water flow is commonly described by Richards' equation, a nonlinear PDE that is typically solved using numerical methods. In this thesis, two alternative physics-informed machine learning approaches are investigated for forward and inverse modeling of unsaturated flow. First, physics-informed neural networks (PINNs) are applied to approximate solutions of Richards' equation in a one-dimensional homogeneous soil setting. Their performance is evaluated by comparison with numerical reference solutions. In addition, an inverse PINN framework is employed to estimate the soil hydraulic parameters α, n and the saturated hydraulic conductivity K_s of the van Genuchten model. Second, a physics-informed deep operator network (DeepONet) is employed to learn the mapping between varying upper boundary flux functions and the corresponding solutions of Richards' equation. In this context, the model's ability to generalize across different boundary conditions is assessed.

Popular Scientific Summary

Water flow through soil is an important process in many environmental applications. In agriculture, for example, understanding how water infiltrates and moves through the ground is essential for efficient irrigation and sustainable water management. These processes can be described mathematically by using differential equations that represent the physical laws driving water movement in soil. One of the most important equations for this purpose is the so-called Richards' equation, a complex nonlinear partial differential equation. Solving this equation typically requires numerical methods that approximate the solution step by step on a spatial and temporal grid. While these methods can produce accurate results, they often require considerable computational effort, especially for large or complex problems.

An alternative approach is to use machine learning methods, such as artificial neural networks. Instead of solving the equation numerically, a neural network can learn to approximate its solution based on the known physical laws, i.e., the governing differential equation of the system. In this context, the term physics-informed machine learning is used because the neural network does not learn only from data but also from equations that describe the underlying physical processes. In this thesis, two different physics-informed machine learning approaches are investigated.

First, physics-informed neural networks (PINNs) are used to approximate solutions of Richards' equation for a simplified soil water flow problem. The obtained predictions are then compared with classical numerical solutions generated with a standard hydrological simulation model. In addition, PINNs are applied in an inverse modeling setup, where the goal is to estimate important soil hydraulic parameters that are part of the mathematical model. Secondly, deep operator networks (DeepONets) are investigated as a more general operator-learning approach. In this method, the neural network learns relationships between input functions and the corresponding solution functions. Unlike PINNs, which solve one specific problem setup, DeepONets aim to generalize across varying boundary conditions and approximate the corresponding solutions of the equation. This enables predictions for previously unseen infiltration scenarios, for example, due to changing rainfall patterns.

Acknowledgements

I would like to thank my supervisors, Adrian Gustafson and Linda Hartman, for their encouraging discussions, great advice, and valuable feedback throughout this project. I am also deeply grateful to my fellow students, who not only made the past two years an unforgettable experience but also contributed greatly to my motivation and understanding through the many discussions during our study sessions. Finally, I would like to thank my husband, Valentin, for his constant mental support as well as for his thoughtful and critical advice on scientific practice throughout my project.

Abbreviations

DeepONet	Deep Operator Network
FNO	Fourier Neural Operator
IC	Initial Condition
LB	Lower Boundary
MSE	Mean Squared Error
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
UB	Upper Boundary

Contents

1 Introduction **11**

2 Theoretical Background **15**
2.1 Governing Physics of Unsaturated Soil Water Flow 15
2.1.1 Hydraulic Properties and Constitutive Relationships 15
2.1.2 Derivation of Richards' Equation 18
2.2 Artificial Neural Networks 20
2.2.1 Function and Structure of Neural Networks 20
2.2.2 Training and Optimization 21
2.2.3 Capabilities and Limitations 21
2.3 Physics-Informed Neural Networks (PINNs) 22
2.3.1 The Training Process 22
2.3.2 Inverse Modeling 23
2.4 Deep Operator Networks (DeepONets) 23
2.4.1 Function and Structure of DeepONets 24
2.4.2 Physics-Informed DeepONets 24

3 Methods **27**
3.1 Problem Formulation and Simulation Setup 27
3.2 Numerical Simulations 28
3.3 General Neural Network Implementation 29
3.4 PINN Implementation and Experimental Setup 29
3.4.1 Forward Problem 29
3.4.2 Inverse Problem 34
3.5 Physics-Informed DeepONet Setup 34

4 Results **37**
4.1 PINN 37
4.1.1 Forward Problem 37
4.1.2 Inverse Problem 41
4.2 Physics-Informed DeepONet 43

5 Discussion **47**
5.1 PINN - Forward Problem 47
5.2 PINN - Inverse Problem 48
5.3 Physics-Informed DeepONet 49

6 Conclusion **51**

Chapter 1

Introduction

The understanding of water flow in unsaturated soils is of high importance across a wide range of applications. For example, improving the efficiency of agricultural irrigation systems requires detailed knowledge of water infiltration and redistribution in soil (Kandelous and Šimůnek, 2010). Further, accurate prediction of soil-related hazards such as rainfall-induced landslides depends on a reliable description of soil water dynamics (Zhang et al., 2018). In those contexts, mathematical modeling of water flow in porous media provides an essential tool for analyzing and predicting such processes.

In most real-world scenarios, water flows through the soil under unsaturated conditions, where the pore space is only partially filled with water and air occupies the remaining volume. Water movement is primarily driven by gradients in pressure and gravity, resulting in downward infiltration as water enters the soil from the surface (Bittelli et al., 2015). A schematic illustration of such a system is shown in Figure 1.1. In this thesis, a simplified one-dimensional soil column is considered, in which water infiltrates vertically from the upper boundary of the soil and moves downward through the soil profile. This setup represents a common idealization of infiltration processes and allows for a controlled investigation of the dynamics.

The conventional approach to model these dynamics is based on partial differential equations (PDEs) that describe the physical laws involved. In particular, unsaturated soil water flow is commonly modeled using Richards' equation, a highly nonlinear PDE that combines mass conservation and momentum (Bittelli et al., 2015). This equation is typically solved using numerical methods such as finite element or finite volume schemes. While these methods can provide accurate solutions, they are associated with several limitations. For instance, to provide numerical stability, the solutions are usually restricted to certain mesh sizes (Kamil et al., 2025). Moreover, they may become computationally expensive for fine spatial and temporal resolutions (Karniadakis et al., 2021), and their application can be challenging when parts of the physics are unknown, such as when incomplete initial or boundary conditions are given.

In recent years, physics-informed machine learning approaches have emerged as a promising alternative to traditional numerical methods (Faroughi et al., 2024). The idea behind those approaches is that the known physical laws, typically in the form of PDEs, are integrated into the training process of neural networks. The classical structure of a neural network is visualized in Figure 1.2. In this thesis, two different physics-informed machine learning techniques for modeling soil water dynamics are employed. First, the applicability of physics-informed neural networks (PINNs) to such problems is examined. PINNs represent a class of neural networks that incorporate the governing physical laws directly into the loss functions of the network,

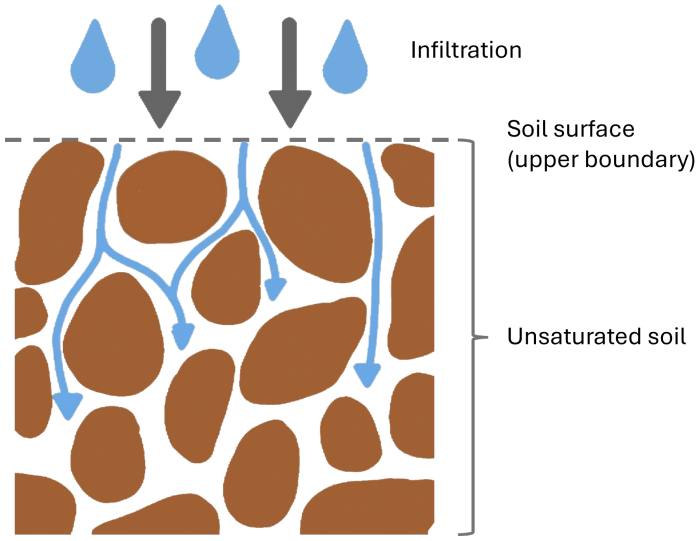

Figure 1.1: Water flow and infiltration in unsaturated soil.

allowing solutions to be obtained by enforcing the underlying PDE (Raissi et al., 2019). Using numerical solutions of the Richards equation as a reference, the ability of PINNs to accurately reproduce these solutions is assessed.

Beyond forward modeling, PINNs also offer a framework for solving inverse problems (Raissi et al., 2019). In this setup, unknown model parameters can be inferred by the PINN by using additional measured or synthetic data. This is particularly relevant for problems where poorly observed or uncertain parameters are involved, as is the case for hydraulic parameters in the soil water flow model (Depina et al., 2022). In this thesis, the potential of PINNs to estimate key parameters of the van Genuchten constitutive model (Bittelli et al., 2015) is investigated.

A common challenge in modeling is the ability to generalize to new conditions, such that flexible future predictions can be made. In the context of soil water flow, this includes predicting the system behavior under varying boundary conditions, such as different precipitation or infiltration patterns. Standard PINNs are designed to solve a single instance of a problem with fixed initial and boundary conditions, limiting their ability to generalize across different scenarios.

To address this limitation, physics-informed deep operator networks (DeepONets) are considered. Those types of neural networks are designed to learn the mapping from an input function to the corresponding output function, which is commonly the solution of the PDE (Wang et al., 2021). This framework enables DeepONets to approximate solution operators rather than individual solutions. The goal in this thesis is to learn the mapping from upper boundary flux functions to the corresponding solutions of the Richards equation. This allows the model to make predictions for previously unseen boundary conditions that describe infiltration patterns into the soil.

Based on these considerations, the following three research questions will be addressed:

(i) How well do physics-informed neural networks reproduce numerical solutions of Richards' equation?

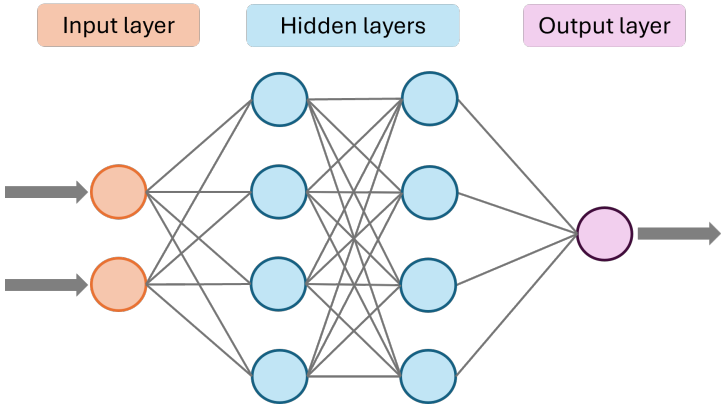

Figure 1.2: Basic structure of a feedforward neural network.

(ii) How well can PINNs estimate soil hydraulic parameters of the van Genuchten model?

(iii) Can physics-informed DeepONets learn the solution operator of Richards' equation such that accurate predictions can be made for unseen upper boundary conditions?

In this thesis, the term "accuracy" refers to the level of agreement between the predictions of the physics-informed machine learning models and the corresponding numerical reference solutions. As no real-world observational data is used, accuracy is assessed only with respect to these numerical solutions.

While PINNs have been applied to both forward and inverse problems involving Richards' equation in various contexts (Depina et al., 2022; Elkhadrawi et al., 2024; Kamil et al., 2025), the use of DeepONets for this class of problems remains limited. In particular, their capability to generalize across varying boundary conditions has not yet been extensively explored. The contribution of this work is, first, to assess the robustness of PINNs in modeling soil water dynamics and parameter estimation in simple setups, and second, to investigate the potential and limitations of physics-informed DeepONets in learning solution operators for Richards' equation and in generalizing across different upper boundary conditions.

The thesis is structured as follows. First, the theoretical background of the key components is presented, including the governing physics of unsaturated water flow, artificial neural networks, physics-informed neural networks, and deep operator networks. Then, the methodology is described, outlining the experimental setup and the implementation of the applied approaches. This is followed by a detailed presentation of the results obtained from the performed investigations. The findings are discussed in the subsequent section, and the thesis concludes with a summary of the main outcomes and potential directions for future work.

Chapter 2

Theoretical Background

2.1 Governing Physics of Unsaturated Soil Water Flow

Unsaturated water flow in porous media describes the movement of water through soil in which the pore space is not completely filled with water, but also contains air. Under such conditions, the flow is typically unsteady and therefore referred to as transient water flow. The main driving forces governing the movement of water are capillary forces, which describe the suction between water and soil particles, as well as gravity and pressure gradients (Hillel, 1998). The overall flow process is commonly described by Richards' equation, which relates these driving forces to water flux. More specifically, Richards' equation combines the two different principles of momentum and mass conservation. The former is represented by Darcy–Buckingham's law, describing the flow, while the latter is expressed through the mass balance equation (Bittelli et al., 2015).

2.1.1 Hydraulic Properties and Constitutive Relationships

Before deriving Richards' equation from Darcy–Buckingham's law and the mass balance equation, the fundamental hydraulic properties involved in these equations are introduced. In particular, the two state variables matric head ψ, a measure of how strongly water is bound in the soil, and volumetric water content θ are presented. Their relationship is described through the soil water retention curve. Among these variables, the matric head ψ serves as the primary state variable and will be the predicted output of the models throughout this thesis. Subsequently, the hydraulic conductivity K and its dependence on the matric head are examined. These constitutive relationships build fundamental components of the Richards equation, which is derived in the next section.

Matric Head and Volumetric Water Content

The matric head ψ [m] arises from the physical interaction between water and soil particles, including both capillary forces in the pore spaces and adsorption to solid particle surfaces. It represents the contribution of capillary and adsorptive forces to the total hydraulic potential of water in soil. These interactions cause water to be held under tension in unsaturated soils, resulting in negative values of ψ (Bonan, 2015; Hillel, 1998).

In some contexts, the term matric potential is used instead of matric head. Matric potential

is defined as the work required per unit mass to move an infinitesimal amount of water from the soil matrix to a reference state (Bittelli et al., 2015). The matric head expresses this potential in terms of energy per unit weight, allowing it to be interpreted as the height of a water column that would generate the same pressure. This representation makes it directly comparable to other components of the hydraulic head, such as the gravitational head.

The volumetric water content θ [m^3m^{-3}] is defined as the fraction of water volume relative to the total volume of the soil. It is a dimensionless quantity, and represents how much of the pore space within the soil is filled with water (Bittelli et al., 2015).

Soil Water Retention Curve

As described above, the matric head reflects the capillary and adsorptive forces exerted by the soil matrix on the pore water and is therefore directly dependent on the amount of water present in the soil. Under saturated conditions, the pore space is completely filled with water and only weak matric forces act on the fluid. As the soil begins to dry, air enters the pores and the remaining water is increasingly held by capillary and adsorptive forces. Consequently, the matric head decreases, i.e., it becomes more negative, meaning the suction force that binds water to the soil increases. This stronger binding of water to the soil matrix makes water movement more difficult, as larger pressure gradients are required to overcome these forces (Bonan, 2015). The exact water retention behavior depends strongly on soil texture and structure. Soils with larger pores, such as sand, typically exhibit a lower water retention compared to fine-textured soils, such as clay (Bittelli et al., 2015).

The functional relationship between matric head and volumetric water content is referred to as the soil water retention curve, also known as the soil moisture characteristic curve. A variety of models have been proposed to describe this functional relationship (Bittelli et al., 2015). One of the most widely used formulations is the van Genuchten model, which expresses the soil water retention curve in terms of the degree of saturation S_e:

$$S_e(\psi) = \frac{\theta - \theta_r}{\theta_s - \theta_r} = \frac{1}{[1 + (\alpha\psi)^n]^m}. \tag{2.1.1}$$

Rewriting this expression in terms of the volumetric water content yields

$$\theta(\psi) = \theta_r + (\theta_s - \theta_r)\frac{1}{[1 + (\alpha\psi)^n]^m}. \tag{2.1.2}$$

Here θ_s [m^3m^{-3}] is the saturated volumetric water content, corresponding to the water content in fully saturated soil, and θ_r [m^3m^{-3}] is the residual volumetric water content, representing the fraction of water that remains bound within the soil pores after full gravitational drainage. The parameters α [m^{-1}], n [-] and m [-] are shape parameters. Commonly, the restriction $m = 1 - 1/n$ is used in order to achieve stable results. The exact parameter values depend on the soil type. Figure 2.1 shows van Genuchten soil water retention curves for three different soil types.

Hydraulic Conductivity

Hydraulic conductivity measures the ability of a porous medium to transmit water and thus quantifies how easily water can move through the pore space. In saturated soils, all pores contribute to flow, resulting in maximum hydraulic conductivity. As the soil becomes unsaturated,

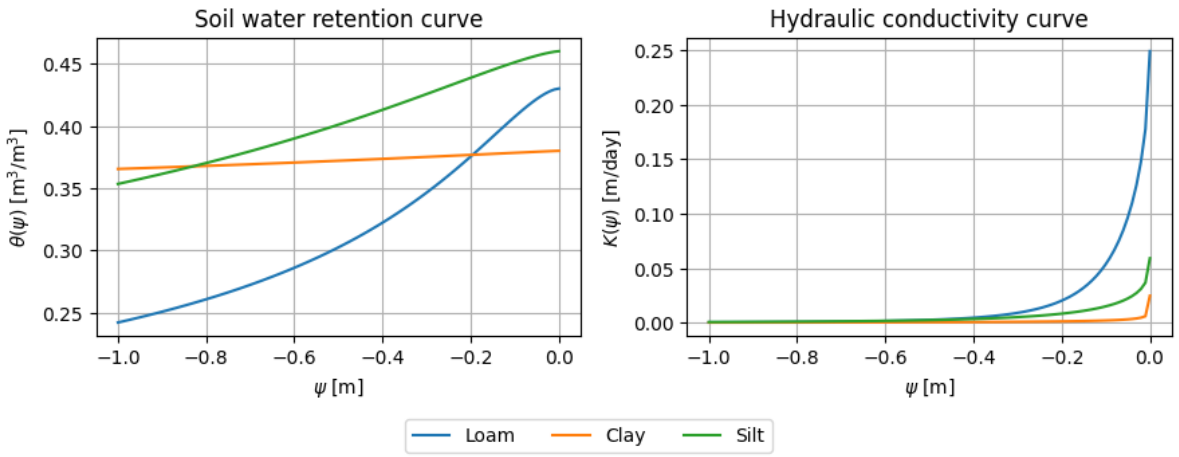

Figure 2.1: Soil water retention curve and hydraulic conductivity for different soil types using the van Genuchten and Mualem–van Genuchten model, respectively.

air replaces water in the pore space. During this process, the largest pores in the soil drain first, and because these large pores provide the most efficient flow pathways, their drainage leads to a rapid reduction in hydraulic conductivity. Consequently, hydraulic conductivity decreases strongly with decreasing water content (Bittelli et al., 2015).

As for the water retention curve introduced before, there exist a couple of different models that describe the relationship between hydraulic conductivity and matric head. In this thesis, the Mualem–van Genuchten model is employed, which is based on the van Genuchten model (2.1.1). According to this model, the hydraulic conductivity as a function of matric head is given by

$$K(\psi) = \begin{cases} \dfrac{K_s \left\{1 - (\alpha\psi)^{mn} \left[1 + (\alpha\psi)^n\right]^{-m}\right\}^2}{\left[1 + (\alpha\psi)^n\right]^{ml}} & \text{if } \psi \leq 0, \\ K_s & \text{if } \psi > 0, \end{cases} \tag{2.1.3}$$

where K_s [m day^{-1}] is the saturated hydraulic conductivity and l [-] is an empirical parameter related to the tortuosity of the soil. The parameters α [m^{-1}], n [-], and m [-] are inherited from the van Genuchten model. Again, this formulation is typically used under the restriction $m = 1 - 1/n$ (Bittelli et al., 2015). Exemplary curves of the hydraulic conductivity for three different soil types are shown in Figure 2.1.

Once the matric head ψ is known, both the volumetric water content θ and the hydraulic conductivity K can be determined from the constitutive relationships. Together with its physical significance in representing capillary forces within the soil, this makes ψ a suitable primary state variable and the chosen model output in this thesis.

Advantages and Limitations of the van Genuchten Model

The presented van Genuchten and Mualem–van Genuchten models are widely used constitutive relationships for describing soil hydraulic properties, particularly in numerical simulations of the Richards equation (Bittelli et al., 2015). One reason for their popularity is that they provide

smooth, continuously differentiable functions that enable stable numerical solutions (Ippisch et al., 2006). In addition, experimental studies have demonstrated that the van Genuchten model has a very good fit to measured soil water retention data across a wide range of soil types and textures (Lipiec et al., 2024). Another advantage is the relatively compact set of parameters, many of which are related to the physical properties of the soil, making the model both flexible and interpretable.

Despite those advantages, there are several limitations worth noting. Ippisch et al. (2006) pointed out that the Mualem–van Genuchten model does not include a well-defined air-entry value. In reality, however, every soil has a maximum pore size that corresponds to a specific air-entry pressure. By ignoring this feature, the model assumes a continuous pore-size distribution extending to zero suction, which can lead to physically unrealistic representations of the soil structure. Consequently, the derived hydraulic conductivity function may be influenced by arbitrarily large pores that do not occur in the actual soil. This issue occurs particularly when the shape parameter $n < 2$, which is the case for fine-textured soils such as loam or clay. In such cases, the model may yield unrealistic values for the hydraulic conductivity (Ippisch et al., 2006). Nevertheless, due to their robustness and analytical convenience, the van Genuchten and Mualem–van Genuchten models are applied in this thesis.

2.1.2 Derivation of Richards' Equation

In this section, Richards' equation is derived from Darcy–Buckingham's law and the mass conservation equation. Together with the constitutive relationships introduced in the previous section, Richards' equation forms the governing PDE that is solved using the physics-informed machine learning methods investigated in this thesis.

Darcy–Buckingham's Law

Water flux through porous media is commonly described by Darcy's law, which was introduced by Henry Darcy in 1856 for saturated conditions (Hillel, 1998). The law states that the water flux is proportional to the gradient of the hydraulic head, such that

$$q = -K\frac{dH}{dz}, \tag{2.1.4}$$

where q [m day^{-1}] is the water flux, K [m day^{-1}] is the hydraulic conductivity, H [m] is the hydraulic head and z [m] is the space dimension. The negative sign indicates that flow occurs in the direction of decreasing hydraulic head (Hillel, 1998). The hydraulic head H represents the total potential of the water and can be expressed as the sum of pressure head h and gravitational head (Da Silva et al., 2007). The gravitational head is typically given directly by the elevation coordinate z, such that $H = h + z$.

Based on this formulation, Darcy's law was later extended to unsaturated flow by Buckingham (1907), leading to the Darcy–Buckingham law. Under unsaturated conditions, the pressure head is typically negative due to capillary effects and is referred to as the matric head ψ. Therefore, in unsaturated soils, the hydraulic head is modified to $H = \psi + z$. In addition, the hydraulic conductivity is expressed as a function of either volumetric water content or matric head, i.e., $K(\theta)$ or $K(\psi)$ (Da Silva et al., 2007). The resulting formulation is:

$$q = -K(\psi)\nabla\left(\psi + z\right), \tag{2.1.5}$$

where ψ [m] is the matric head and $K(\psi)$ [m day^{-1}] denotes the hydraulic conductivity as a function of matric head ψ. In a one-dimensional soil column, which will be the focus of this thesis, the equation becomes

$$q = -K(\psi)\left(\frac{\partial \psi}{\partial z} + 1\right). \tag{2.1.6}$$

Darcy–Buckingham's law forms the basis for deriving Richards' equation when combined with the mass conservation equation.

Mass Conservation

The mass conservation law states that the mass of a closed system remains constant over time. When applying it to soil water flow, this principle implies that any change in water mass within a volume must be balanced by the net flux across its boundaries. Therefore, the rate of change within a representative volume must equal the difference between inflow and outflow across its surfaces, plus any internally generated or extracted mass (Bittelli et al., 2015). Neglecting internal sources and sinks (e.g., root water uptake), this leads to the mass conservation equation, also referred to as the continuity equation:

$$\frac{\partial \theta}{\partial t} + \nabla \cdot q = 0, \tag{2.1.7}$$

where θ [m^3 m^{-3}] is the volumetric water content, q [m day^{-1}] is the water flux, and t [day] is the time dimension.

Richards' Equation

To obtain the governing equation for transient water flow in unsaturated soil, mass balance is combined with the water flux relationship described by Darcy–Buckingham's law. Substituting Darcy–Buckingham's law into the continuity equation yields the Richards equation:

$$\frac{\partial \theta}{\partial t} = \nabla \cdot \left[K\left(\psi\right) \nabla \left(\psi + z\right)\right]. \tag{2.1.8}$$

Note that in this case, z is the upward-pointing vertical dimension. In the one-dimensional vertical case, the equation writes

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left[K\left(\psi\right)\left(\frac{\partial \psi}{\partial z} + 1\right)\right]. \tag{2.1.9}$$

The description and units of all variables within this equation can be found in Table 2.1. This formulation is often called the mixed form of Richards' equation because both water content and matric head appear as dependent variables. The equation represents a second-order nonlinear PDE, with the pronounced nonlinearity resulting primarily from the dependence of hydraulic conductivity on the matric head. (Bittelli et al., 2015; Hillel, 1998).

Table 2.1: Overview of the variables used in the mixed form of Richards' equation.

Variable	Description	Unit
θ	Volumetric water content	$m^3\ m^{-3}$
ψ	Matric head	m
K	Hydraulic conductivity	m day^{-1}
t	Time	days
z	Depth of the soil	m

2.2 Artificial Neural Networks

In this section, artificial neural networks are introduced, which are the foundation for the methods applied in this thesis. Artificial neural networks are models inspired by the structure and function of biological neural systems and the mechanisms responsible for learning in the human brain. In biological systems, learning is enabled by neurons that are interconnected through synapses. Similarly, artificial neural networks consist of a number of nodes, also referred to as neurons, which are connected by weighted edges (Meedeniya, 2023). The primary objective of a feedforward artificial neural network is to approximate an unknown function f that maps input features $\mathbf{x}$ to output values y, typically denoted as $y = f(\mathbf{x})$. Because of their structure, these neural networks can approximate complex and highly nonlinear functions (Goodfellow et al., 2016).

2.2.1 Function and Structure of Neural Networks

Classical feedforward neural networks are organized in a layered structure consisting of an input layer, one or more hidden layers, and an output layer. The input layer receives the feature vector $\mathbf{x}$, while the output layer produces the network's prediction of y. Each layer is composed of a set of neurons that are fully connected to the neurons in the subsequent layer. The value of each neuron is computed as the weighted sum of the outputs from the previous layer, supplemented by a bias term, and then transformed by a nonlinear activation function σ. Mathematically, this can be expressed for any layer l as

$$\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{2.2.1}$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ denote the weight matrix and bias vector of layer l, respectively, and $\mathbf{h}^{(l-1)}$ represents the output of the previous layer. By applying these transformations sequentially for each layer, the network performs a forward pass that maps inputs to outputs (Meedeniya, 2023).

Activation functions play a crucial role in enabling neural networks to model complex relationships. Without nonlinear activation functions, a neural network would reduce to a sequence of linear transformations, which is equivalent to a linear regression model regardless of its depth. Consequently, such a network would be unable to capture nonlinear and highly complex patterns in the data. By introducing nonlinear activation functions, the network can learn more expressive mappings between inputs and outputs. In addition, activation functions can help emphasize relevant features while suppressing less important ones. Commonly

used activation functions include the sigmoid function, the hyperbolic tangent (tanh), and the rectified linear unit (ReLU) (Meedeniya, 2023).

2.2.2 Training and Optimization

To approximate the target function f that maps inputs to outputs, the weights and biases, referred to as the parameters of the neural network, must be configured such that the network produces outputs that closely match the corresponding target values y for given inputs $\mathbf{x}$. To achieve this, the neural network is trained on a set of paired input-output samples $(\mathbf{x}, y)$, where the output values are observed target values that are associated with the respective input values. To quantify the discrepancy between the network's prediction $\hat{y}$ for a certain input and the true target value y, a loss function is defined. A common choice for this is the mean squared error (MSE), which measures the average squared difference between a set of predicted and observed values (Meedeniya, 2023):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2. \tag{2.2.2}$$

The training process is formulated as an optimization problem in which the objective is to minimize the loss function with respect to the network parameters, i.e., the weights and biases.

To solve this optimization problem, the backpropagation algorithm is applied, typically using gradient descent (Goodfellow et al., 2016; Meedeniya, 2023). In this method, the gradients of the loss function are computed with respect to all trainable parameters. By applying the chain rule of calculus, these partial derivatives are calculated layer by layer, propagating the loss from the output layer backward through the network to the input layer. Based on these gradients, the weights and biases are iteratively updated (Meedeniya, 2023). Parameters that have a stronger influence on the loss result in larger gradients and therefore undergo larger updates, while those with less impact are adjusted more slightly. This process is applied to the entire training dataset, where one full pass through all training samples is referred to as an epoch. In practice, multiple epochs are required for the network to converge towards a set of parameters that sufficiently minimize the loss function.

After training, a neural network should ideally be able to generalize, i.e., make accurate predictions on unseen input data. To assess this capability, the model is evaluated on a separate dataset, known as the test data, by computing the corresponding loss. If the network performs well on the training data but poorly on the test data, this is referred to as overfitting. In this case, the model has learned the training data too closely, including noise or irrelevant patterns, and fails to capture the underlying functional relationship.

2.2.3 Capabilities and Limitations

An important theoretical property of neural networks is the universal approximation property, which states that a sufficiently large neural network can approximate any function f, including highly nonlinear and complex mappings, to arbitrary accuracy. This result provides a strong argument for the use of neural networks as universal function approximators. However, in practice, the existence of such a representation does not guarantee that it can be found during training. The optimization process may fail to identify the desired function due to limitations of the training algorithm, such as convergence to local minima or inefficient exploration of

the parameter space. Furthermore, even if a network fits the training data well, it may still generalize poorly due to overfitting. More generally, there is no universal procedure that, based on a finite set of training examples, can guarantee perfect generalization to unseen data. In this context, deeper network architectures can be advantageous, as they often represent complex functions more efficiently, requiring fewer neurons and potentially leading to improved generalization performance (Goodfellow et al., 2016).

Overall, artificial neural networks have been successfully applied to a wide range of problems and have proven to be particularly well-suited for complex modeling tasks. The architecture and learning approach introduced in this section represent the most basic form of neural networks. Numerous extensions and modifications have been made to this in order to address specific challenges and improve performance. The basic principles outlined here also form the foundation of more specialized approaches, including physics-informed neural networks, which are introduced in the following section.

2.3 Physics-Informed Neural Networks (PINNs)

The concept of PINNs was introduced by Raissi et al. (2019). These types of neural networks are designed to learn not only from available data but also from known physical laws governing the respective system. One of the most common applications of PINNs is the approximation of solutions to PDEs, which is also the application considered in this thesis.

The governing physical laws are enforced by incorporating the PDE into the neural network's loss function. Consequently, the network is penalized when its output violates the underlying physics equations, causing it to learn solutions that satisfy these laws. In a typical PINN setup, the inputs consist of space-time coordinates (x, t), while the output represents the approximated solution u of the PDE at the corresponding point in space and time (Faroughi et al., 2024). Although PINNs predict the solution at specific spatio-temporal coordinates, they are considered mesh-independent methods, as they do not rely on a predefined computational grid (Cuomo et al., 2022; Faroughi et al., 2024). Instead, the solution is represented by a continuous neural network function that can be evaluated at arbitrary points within the domain. The architecture of PINNs is further described in Section 3.4 and illustrated in Figure 3.2.

2.3.1 The Training Process

The training process of a PINN differs fundamentally from standard supervised learning. Instead of relying primarily on paired input-output data, the network is trained by enforcing the governing physical laws at a set of sampled collocation points (Raissi et al., 2019). These points are coordinates within the spatio-temporal domain at which the model's prediction is evaluated with respect to the governing PDE. Unlike measurement points, these locations do not require associated reference data. Their purpose is to enforce the physical consistency of the solution throughout the domain (Cuomo et al., 2022).

To evaluate the PDE at the collocation points, the required partial derivatives of the network's output with respect to the input coordinates are computed using automatic differentiation. These derivatives are then used to calculate the PDE residual, which quantifies how well the neural network prediction satisfies the governing equation. The total loss function of a PINN typically splits into several individual loss terms that are added up, often with

weighting factors to control their relative importance during training. The PDE residual loss, which penalizes violations of the governing equation throughout the space-time domain, forms the central component of the total loss (Faroughi et al., 2024).

In addition, initial and boundary conditions are incorporated into the loss function. For initial conditions, available reference data is commonly used in the standard supervised learning manner by minimizing the MSE between predicted and reference values. The treatment of boundary conditions depends on their mathematical form. Dirichlet boundary conditions are typically enforced through reference values at the boundary, similar to the initial condition, whereas Neumann or flux-type boundary conditions are usually incorporated by evaluating the corresponding differential expression at the boundary points (Cuomo et al., 2022).

In certain applications, additional observational or synthetic reference data may also be included for the training process. In such cases, another loss term is introduced to measure the discrepancy between the model predictions and the data, typically again using the MSE (Cuomo et al., 2022).

2.3.2 Inverse Modeling

In addition to approximating solutions of PDEs, PINNs can also be used to solve inverse problems, in which unknown parameters of the governing equation are estimated (Faroughi et al., 2024). In this setting, the goal is not only to learn the solution of the PDE but also to infer one or multiple parameters that characterize the physical system.

In order to enable parameter estimation with PINNs, reference data from observations is required. This data does not need to be particularly extensive. In most cases, parameter estimation can be achieved using relatively small datasets with up to 100 data samples. An additional loss term, which measures the MSE of observed and predicted values, is added to the total loss. This allows the unknown parameters to be identified (Cuomo et al., 2022; Faroughi et al., 2024).

The unknown parameters of the PDE are incorporated into the neural network as learnable parameters. Instead of predicting them as outputs of the neural network, they are treated as additional trainable parameters optimized together with the neural network weights and biases during training (Cuomo et al., 2022). Through the combined minimization of the PDE residual loss and the data loss, the network learns both the PDE solution and the parameter values that best explain the observed data and the known physical laws.

2.4 Deep Operator Networks (DeepONets)

PINNs, as introduced in the previous section, are designed to approximate a solution of a PDE for a single problem instance, i.e., for fixed initial and boundary conditions. While this approach is powerful in certain setups, it has limited flexibility when the problem configuration changes. In particular, if boundary or initial conditions vary, a PINN typically needs to be retrained, which can be computationally expensive. In the context of soil water flow, this limitation becomes especially relevant. Practical applications often require predictions for a range of varying upper boundary conditions, for instance, due to changing precipitation patterns. Solving the governing equation repeatedly for each new configuration using PINNs would therefore be inefficient.

An alternative approach is to shift from learning individual solutions to learning the underlying solution operator. Instead of approximating a single function $u(x)$, the goal is to learn a mapping between function spaces. More specifically, one seeks to approximate an operator,

$$\mathcal{G} : a(x) \mapsto u(x),$$

that maps input functions, such as external forcing terms, initial conditions, or boundary conditions, to the corresponding solution function of the PDE.

DeepONets provide a framework to realize this idea (Lu et al., 2021). The theoretical foundation of DeepONets is rooted in the universal approximation theorem for operators. This theorem states that neural network architectures can approximate a broad class of nonlinear, continuous operators with arbitrary accuracy, given sufficient capacity. This property makes DeepONets a natural choice for modeling parametric PDE problems, where the solution depends on varying input functions.

2.4.1 Function and Structure of DeepONets

DeepONets are composed of two subnetworks, each processing a different type of input. The architecture of these networks is described in more detail in Section 3.5 and illustrated in Figure 3.3. The separation of the two subnetworks reflects the structure of the operator learning problem, where both the input function and the evaluation location must be taken into account. The first subnetwork, referred to as the branch net, is responsible for encoding the input function. Since functions are infinite-dimensional, they must be discretized at a fixed set of locations, commonly called sensor points, in order to be processable by a neural network. The resulting vector of function values serves as input to the branch network, which extracts a latent representation characterizing the specific input function. Intuitively, the branch network determines which particular problem instance is being considered, for example, a specific boundary or initial condition. The second subnetwork, known as the trunk net, takes, similarly to PINNs, time-space coordinates as inputs at which the solution should be evaluated. The trunk net learns a set of basis functions over the domain. In that sense, it determines where the solution is evaluated, regardless of the specific input function.

The outputs of both the branch and trunk nets are combined to produce the final predictions. This is typically done by computing the dot product between the feature vectors obtained from the last layer of both subnetworks. The separation between dependence on the input function and on the evaluation coordinates enables the network to learn complex solution operators (Lu et al., 2021).

2.4.2 Physics-Informed DeepONets

DeepONets were originally introduced by (Lu et al., 2021) as purely data-driven models. In this setup, the network is trained on datasets consisting of input-output pairs, where the outputs are obtained from numerical simulations. Therefore, the loss function is defined on the error between the data and the prediction alone. This approach can achieve high prediction accuracy given sufficiently dense data. However, generating such datasets requires repeatedly solving the underlying PDE using classical numerical methods.

Addressing this limitation, Wang et al. (2021) introduced a physics-informed extension of DeepONets. Similar to PINNs, this approach incorporates the governing physical laws directly

into the training process. The data-based loss function is extended with physics-based terms, including the PDE residual and the initial and boundary conditions. This procedure reduces dependence on labeled training data and enables training with only sparse data, or even in a completely data-free setting.

While the physics-informed DeepONet approach yields promising results in learning the underlying physics for simple PDE examples, the training process can become challenging when dealing with stiff or high-frequency dynamics. Especially, problems that exhibit steep gradients in the PDE solution may lead to reduced accuracy (Wang et al., 2021).

Chapter 3

Methods

3.1 Problem Formulation and Simulation Setup

A comparatively simple soil and simulation setup is chosen in this thesis in order to evaluate the applied machine learning methods under controlled conditions. The investigation of more complex scenarios is left for future work and lies beyond the scope of this thesis.

The mixed form of Richards' equation (2.1.8) is employed to model water flow in unsaturated soils. The equation is solved with respect to the matric head, ψ, which serves as the primary state variable. A detailed definition of all variables is provided in Table 2.1. The constitutive relationships required to close the system are given by the soil water retention function $\theta(\psi)$ and the hydraulic conductivity function $K(\psi)$. These are described using the van Genuchten model (2.1.2) and the Mualem–van Genuchten model (2.1.3), respectively. The soil is assumed to be of loamy texture and homogeneous with regard to its structure. The applied parameter values for the constitutive relationships are listed in Tab. 3.1.

A one-dimensional vertical soil column with a depth of 1 m is considered. The spatial domain, denoted by Ω, is defined such that $z = 0$ m corresponds to the soil surface and $z = -1$ m to the bottom of the column. Accordingly, $\Omega = [-1, 0]$. The simulations are performed over a total time period of one day. The temporal domain, denoted by I, is therefore given by $I = [0, 1]$.

Table 3.1: Values of van Genuchten parameters (cf. (2.1.2) and (2.1.3)) for a loamy soil.

Parameter	**Description**	**Value**	**Unit**
α	Fitting parameter	3.6	m^{-1}
n	Fitting parameter	1.56	-
m	$m = 1 - 1/n$	$1 - 1/n$	-
θ_r	Residual volumetric water content	0.078	$\mathrm{m}^3\ \mathrm{m}^{-3}$
θ_s	Saturated volumetric water content	0.43	$\mathrm{m}^3\ \mathrm{m}^{-3}$
K_s	Saturated conductivity	0.2496	$\mathrm{m\ day}^{-1}$
l	Related to tortuosity of pore space	0.5	-

Initial and Boundary Conditions

The following initial and boundary conditions are applied to obtain a unique solution of Richards' equation.

Initial condition (IC): At time $t = 0$ days, the soil column is assumed to be in a homogeneous state with a constant matric head of -1 m over the entire depth $z \in \Omega$:

$$\psi(z, 0) = -1. \tag{3.1.1}$$

Upper boundary (UB) condition: At the soil surface ($z = 0$ m), a prescribed constant or variable water flux $q(t)$ is enforced in form of a Neumann boundary condition. This flux must satisfy Darcy–Buckingham's law (2.1.6):

$$q(t) = -K\left(\psi(0, t)\right)\left(\frac{\partial \psi(0, t)}{\partial z} + 1\right), \quad t \in I. \tag{3.1.2}$$

A negative flux $q(t) < 0$ corresponds to infiltration of water into the soil column, e.g., due to precipitation, whereas a positive flux $q(t) > 0$ corresponds to water being extracted into the atmosphere, which happens due to evaporation. In this thesis, only infiltration into the soil and therefore negative fluxes are considered.

Lower boundary (LB) condition: At the bottom of the soil column ($z = -1$ m), a free drainage condition is assumed. This corresponds to a Neumann boundary condition:

$$\frac{\partial \psi(-1, t)}{\partial z} = 0, \quad t \in I. \tag{3.1.3}$$

This condition results in a total hydraulic gradient of $\left(\frac{\partial \psi}{\partial z} + 1\right) = 1$ which leads to a total flux q of the negative hydraulic conductivity $-K(\psi)$. This implies that the water can leave the domain freely under gravity without resistance at the lower boundary.

3.2 Numerical Simulations

To obtain numerical reference solutions, the HYDRUS-1D modeling software (version 4.16.0110) is used (Šimůnek et al., 2008). HYDRUS-1D is a simulation tool for modeling water flow as well as heat and solute transport in variably porous media. For the simulation of water flow, the software solves the Richards equation numerically using a Galerkin-type linear finite element scheme. To integrate in time, the model applies an implicit finite-difference method (Šimůnek et al., 2025).

The simulation setup applied here follows the tutorial example "Example 1: Infiltration of Water into a Single-Layered Soil Profile" by Rassam et al. (2018), with some modifications. Spatial and temporal units are defined in meters and days, respectively. Unless otherwise specified, the time discretization is performed using 200 print times, which corresponds to 201 time steps at which the solution is evaluated. These time points are distributed uniformly with equal distance over the simulation period. The spatial domain is discretized using a finite element mesh consisting of 1001 equidistant nodes, at which the solution is computed.

As soil type, "loam" is selected and the default values for the van Genuchten parameters corresponding to Table 3.1 are applied. The upper boundary condition is defined depending on the respective investigation. In the case of constant upper boundary flux, a constant flux value is entered, whereas for a time-dependent flux function $q(t)$, the function is discretized and specified at the corresponding time points used for the temporal discretization. After running the simulation, the numerical solution for the matric head ψ is obtained from the *Nod_Inf.out* output file.

Using this setup, HYDRUS-1D is used to generate reference solutions for the predefined soil, boundary conditions, and flux scenarios described above. These solutions serve as benchmark data for training and evaluation of the machine learning models.

3.3 General Neural Network Implementation

Both the PINN and the physics-informed DeepONet are implemented using the *PyTorch* framework. Preliminary runs were conducted to determine suitable hyperparameters, including the network architecture, learning rate, and number of training epochs. Based on these runs, the number of hidden layers is fixed to eight with 20 neurons per layer, and the learning rate is set to 10^{-3}. The number of training epochs is varied between 3000 and 4000 depending on the application.

For the PINN, a fully connected feedforward neural network is employed. The input layer comprises two neurons corresponding to the spatial and temporal coordinates (z, t), and the output layer consists of a single neuron representing the predicted matric head ψ.

The DeepONet architecture, as described in Section 2.4, consists of two sub-networks, the branch and the trunk net. Both the branch and trunk networks are implemented as fully connected feedforward networks with eight hidden layers and 20 neurons per layer as defined above. The trunk network takes the spatial and temporal coordinates (z, t) as input and therefore has an input layer with two neurons, analogous to the PINN. The branch network receives the discretized input function, i.e., the sensor values, with the number of input neurons corresponding to the number of sensors. The output of the combined architecture is a single scalar value that gives the predicted matric head ψ.

For both models, the weights of the networks are initialized using the Xavier uniform distribution, and all biases are initialized to zero. The hyperbolic tangent function is used as the activation function in all hidden layers. In order to improve numerical stability during training, the input values within a forward pass are normalized to $[-1, 1]$ using min-max normalization. Further, the output is scaled to a physically reasonable range of $\psi \in [-1, 0]$ by applying the negative sigmoid function to the unconstrained output. Training is performed using the Adam optimizer.

3.4 PINN Implementation and Experimental Setup

3.4.1 Forward Problem

The applied methods described below form the foundation of all investigations, including the sensitivity analysis, the inverse problem, and, to some extent, the setup for the physics-informed DeepONet. All deviating or additional implementations will be explained in the subsequent sections of this chapter.

Training and Test Data

As introduced in Section 2.3.1, PINNs are not primarily trained on paired input-output data, as it is common in classical supervised learning. Instead, training is based on collocation points, i.e., sampled coordinates across the spatio-temporal domain at which the governing equations and initial and boundary conditions are enforced. For the problem case in this thesis, collocation points are sampled across the spatial domain Ω and the temporal domain I.

Because each physical constraint of the problem, i.e., the governing PDE, as well as the initial and boundary conditions, is incorporated into the training process via separate loss terms, different sets of collocation points are defined for each component. Only the IC and the UB conditions require associated reference data of the matric head ψ and the flux $q(t)$, respectively. In contrast, the residual and LB conditions are enforced purely through the governing equations. Accordingly, the following four sets of training data are used:

(i) **Residual points (interior domain)** sampled over the full space–time domain:

$$(z_{\mathrm{R}}, t_{\mathrm{R}}) \in \Omega \times I. \tag{3.4.1}$$

(ii) **Initial condition points** at time $t = 0$ days, with corresponding reference values for the matric head as defined in (3.1.1):

$$((z_{\mathrm{IC}}, 0), \psi_{\mathrm{IC}}), \quad z_{\mathrm{IC}} \in \Omega. \tag{3.4.2}$$

(iii) **Upper boundary condition points** at the soil surface $z = 0$ m, with prescribed flux values as defined in (3.1.2):

$$((0, t_{\mathrm{UB}}), q_{\mathrm{UB}}), \quad t_{\mathrm{UB}} \in I. \tag{3.4.3}$$

(iv) **Lower boundary condition points** at the bottom boundary $z = -1$ m, where the free drainage condition is enforced, according to (3.1.3):

$$(-1, t_{\mathrm{LB}}), \quad t_{\mathrm{LB}} \in I. \tag{3.4.4}$$

Each of those data sets differs in size and is divided into training and test subsets using an 80-20 split, where 80% is used for training and the remaining 20% for evaluation. The exact number of data points for each set varies depending on the investigation and is specified in the corresponding results section. Figure 3.1 shows the sampled collocation points for all four data sets, with training points on the left and test points on the right, as an example.

The residual points (green) are generated by randomly sampling the spatial coordinate from a uniform distribution over the spatial domain, while the temporal coordinate is sampled with a slight bias toward earlier times $t = 0$. To achieve this, the temporal collocation points are generated using the following transformation:

$$t_{\mathrm{R}} = t_{\min} + (t_{\max} - t_{\min}) \cdot \tilde{t}^{\beta}, \quad \tilde{t} \in [0, 1], \tag{3.4.5}$$

where $t_{\min} = 0$ and $t_{\max} = 1$ correspond to the bounds of the temporal domain. The variable $\tilde{t}$ represents an unscaled uniformly sampled value, and β is a scaling factor that controls the

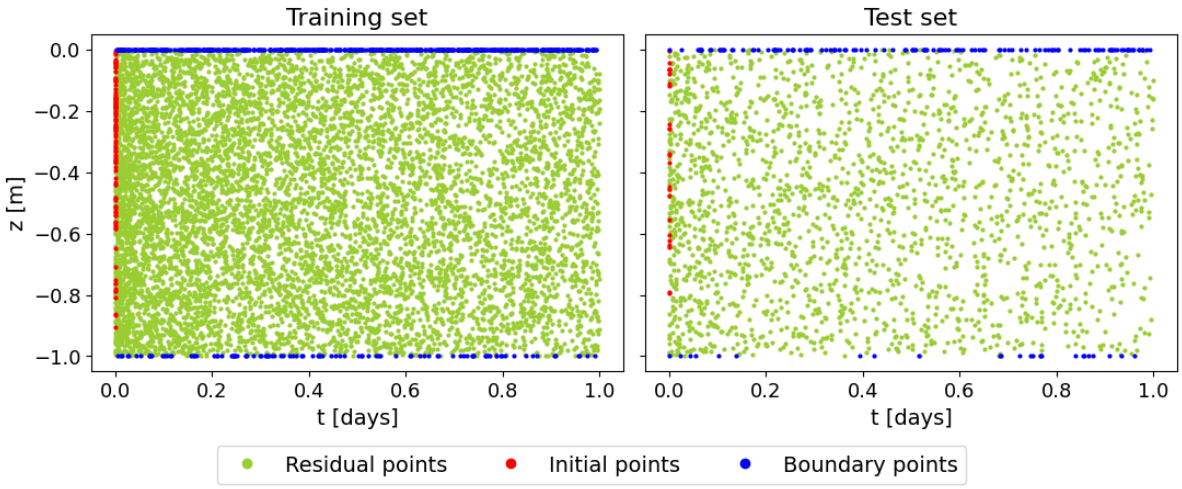

Figure 3.1: Sampled collocation points.

degree of bias. For $\beta > 1$, the sampled points are biased towards earlier times, while $\beta < 1$ results in a bias towards later times. The value of β is varied across calculations and is specified in the corresponding results section.

Similarly, the IC points (red) are slightly concentrated near the soil surface $z = 0$. The spatial collocation points for the IC are generated using the transformation:

$$z_{\mathrm{IC}} = z_{\min} + (z_{\max} - z_{\min}) \cdot \tilde{z}^{0.5}, \quad \tilde{z} \in [0, 1], \tag{3.4.6}$$

where $z_{\min} = -1$ and $z_{\max} = 0$ define the bounds of the spatial domain. In this case, a fixed scaling factor of 0.5 is used for all calculations. Since the spatial domain is defined over negative values, this transformation results in a higher density of points near $z = 0$. This non-uniform sampling strategy was found to improve the convergence during training and to enhance overall model performance.

Loss Functions

The total loss function is composed of multiple sub-loss terms, as mentioned in Section 2.3.1. This division into different loss terms stems from the requirement that the governing equation, as well as the initial and boundary conditions, must be enforced individually during training. Corresponding to the training and test datasets, the following loss functions are defined for each component of the problem:

(i) **Residual loss:** The residual loss term enforces the PDE on the interior space-time domain. It is defined as the mean of squared residuals across the residual points:

$$loss_{\mathrm{R}} = \frac{1}{n_{\mathrm{R}}} \sum_{i=1}^{n_{\mathrm{R}}} f_{\mathrm{R}} \left(z_i^{\mathrm{R}}, t_i^{\mathrm{R}} \right)^2, \tag{3.4.7}$$

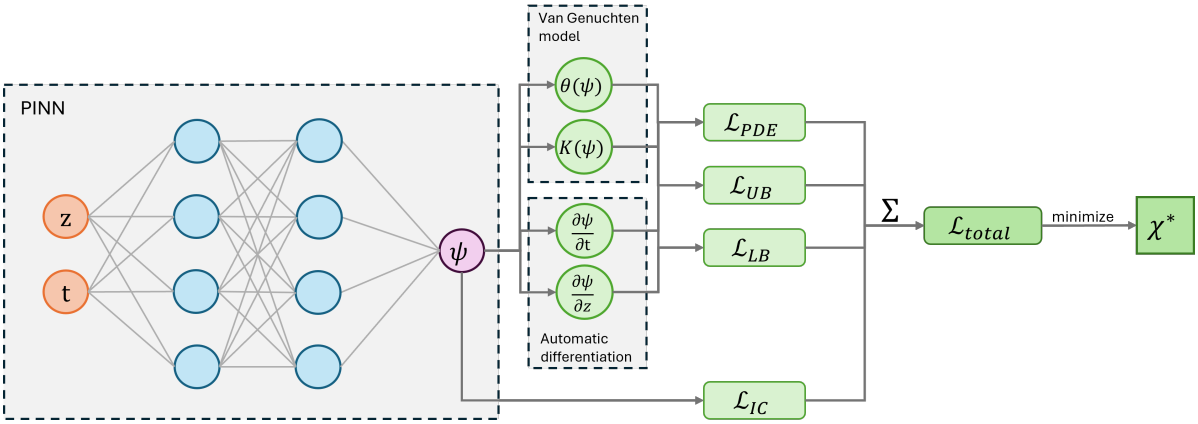

Figure 3.2: PINN architecture.

where

$$f_{\mathrm{R}}\left(z_i^{\mathrm{R}}, t_i^{\mathrm{R}}\right) = \frac{\partial \theta\left(\hat{\psi}_i^{\mathrm{R}}\right)}{\partial t} - \frac{\partial}{\partial z}\left[K\left(\hat{\psi}_i^{\mathrm{R}}\right)\left(\frac{\partial \hat{\psi}_i^{\mathrm{R}}}{\partial z} + 1\right)\right]. \tag{3.4.8}$$

(ii) **Initial condition loss:** The IC loss enforces the IC by taking the MSE between the true value of the matric head ψ and the predicted value $\hat{\psi}$ across the initial points:

$$loss_{\mathrm{IC}} = \frac{1}{n_{\mathrm{IC}}} \sum_{i=1}^{n_{\mathrm{IC}}} \left(\psi_i^{\mathrm{IC}} - \hat{\psi}_i^{\mathrm{IC}}\right)^2. \tag{3.4.9}$$

(iii) **Upper boundary condition loss:** The Neumann UB condition is enforced by a loss term that is defined as the MSE between the true flux $q(t)$ into the soil and the predicted flux $\hat{q}(t)$ at the upper boundary:

$$loss_{\mathrm{UB}} = \frac{1}{n_{\mathrm{UB}}} \sum_{i=1}^{n_{\mathrm{UB}}} \left(q\left(t_i^{\mathrm{UB}}\right) - \hat{q}\left(t_i^{\mathrm{UB}}\right)\right)^2, \tag{3.4.10}$$

where

$$\hat{q}\left(t_i^{\mathrm{UB}}\right) = -K\left(\hat{\psi}_i^{\mathrm{UB}}\right)\left(\frac{\partial \hat{\psi}_i^{\mathrm{UB}}}{\partial z} + 1\right). \tag{3.4.11}$$

(iv) **Lower boundary condition loss:** The Neumann LB condition is enforced by the mean of the squared partial derivatives of ψ with respect to z, according to (3.1.3):

$$loss_{\mathrm{LB}} = \frac{1}{n_{\mathrm{LB}}} \sum_{i=1}^{n_{\mathrm{LB}}} \left(\frac{\partial \hat{\psi}_i^{\mathrm{LB}}}{\partial z}\right)^2. \tag{3.4.12}$$

The notations $n_{\mathrm{R}}, n_{\mathrm{IC}}, n_{\mathrm{UB}}, n_{\mathrm{LB}}$ indicate the number of training data points in the corresponding data subset. Figure 3.2 illustrates the general architecture of the PINN and how the loss terms are built and combined. As mentioned before, the constitutive relationships for the volumetric water content $\theta(\psi)$ and hydraulic conductivity $K(\psi)$ are evaluated using the van

Genuchten and Mualem–van Genuchten models. The partial derivatives inside the loss terms are computed using automatic differentiation. Finally, the total loss function of the PINN combines the individual loss terms by taking the weighted sum:

$$\mathcal{L} = w_{\mathrm{R}} \cdot loss_{\mathrm{R}} + w_{\mathrm{IC}} \cdot loss_{\mathrm{IC}} + w_{\mathrm{UB}} \cdot loss_{\mathrm{UB}} + w_{\mathrm{LB}} \cdot loss_{\mathrm{LB}} \tag{3.4.13}$$

The weights $w_{\mathrm{R}}, w_{\mathrm{IC}}, w_{\mathrm{UB}}, w_{\mathrm{LB}}$ act as scaling factors for the single loss terms and are introduced to balance their relative contributions during training. This ensures that the network learns to satisfy the governing equation and the initial and boundary conditions simultaneously, rather than overemphasizing a single component. The choice of weights depends on the specific experimental setup, as different configurations yield different magnitudes for the individual loss terms. Therefore, the weights are adjusted for each calculation to achieve a well-balanced training process and a sufficiently low total loss. The exact values used in each case are specified in the corresponding results section.

Evaluation

In order to assess the training process and the extent to which the model satisfies the governing equation and associated conditions, the individual loss terms are evaluated on the corresponding test data sets. While low test losses indicate that the physical constraints are well enforced, they are not sufficient to fully assess the model's predictive performance. In particular, a model may satisfy the loss terms while still deviating from the reference numerical solution.

Therefore, the primary evaluation approach in this work is based on a direct comparison between the PINN predictions and the numerical reference solutions. This evaluation is performed on an independent set of grid points that was not used during training or hyperparameter tuning, ensuring an assessment on previously unseen data. The comparison is performed both qualitatively, through graphical analysis, and quantitatively using the MSE, defined as

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(\psi_i^{\mathrm{NUM}} - \psi_i^{\mathrm{PINN}} \right)^2, \tag{3.4.14}$$

where ψ^{NUM} and ψ^{PINN} denote the numerical and PINN predictions of the matric head, respectively. To ensure a consistent comparison, the trained PINN is evaluated at the same grid points used by the numerical model.

Sensitivity Analysis

For the forward PINN case, a sensitivity analysis is conducted to investigate the influence of the van Genuchten parameters $\alpha, n, \theta_s, \theta_r$ and K_s on the predicted solution. Starting from the baseline parameter set (Tab. 3.1), each parameter is varied individually by $\pm 20\%$, while all remaining parameters are kept fixed at their baseline values. For each resulting set of parameters, the PINN is retrained and the predicted matric head ψ is evaluated at the same grid points used by the numerical model.

To assess the impact of these variations, the resulting solutions are compared to a baseline scenario in which all parameters keep their default values. The comparison focuses on the position and shape of the wetting front, defined as the region in the solution space where -0.7 m $< \psi < -0.3$ m. The analysis is performed qualitatively by comparing the wetting fronts

obtained for increased and decreased parameter values against the baseline case. The wetting front is selected as the evaluation criterion because the solution is particularly sensitive to variations in the van Genuchten parameters within this region.

3.4.2 Inverse Problem

The inverse parameter estimation is performed for the van Genuchten parameters α, n, and K_s. Each parameter is estimated in an individual setup by incorporating it as an additional trainable parameter into the neural network, while all remaining parameters stay fixed as external parameters. The parameters to be estimated are initialized with predefined initial guesses: $\alpha = 4.0$ m^{-1}, $n = 1.3$, and $K_s = 0.4$ m day^{-1}.

In contrast to the forward problem, the inverse setup requires additional reference data of the output (cf. Section 2.3.2). For this purpose, synthetic data for the matric head ψ is obtained from the numerical model and used as additional training data. A total of 150 data samples are extracted from the numerical solution on a structured grid. Specifically, 50 equidistant spatial points along the soil depth are selected at three distinct time points, $t_1 = 0.248$ days, $t_2 = 0.504$ days, and $t_3 = 0.752$ days, resulting in $3 \times 50 = 150$ data points.

To incorporate this data into the training process, an additional loss term is introduced, defined as the MSE between the PINN prediction $\hat{\psi}$ and the numerical reference data for ψ at the corresponding grid points:

$$loss_{\text{data}} = \frac{1}{n_{\text{data}}} \sum_{i=1}^{n_{\text{data}}} \left(\psi_i^{\text{data}} - \hat{\psi}_i^{\text{data}} \right)^2, \tag{3.4.15}$$

where $n_{\text{data}} = 150$ denotes the total number of data samples. The total loss function accordingly modifies to

$$\mathcal{L} = w_{\text{R}} \cdot loss_{\text{R}} + w_{\text{IC}} \cdot loss_{\text{IC}} + w_{\text{UB}} \cdot loss_{\text{UB}} + w_{\text{LB}} \cdot loss_{\text{LB}} + w_{\text{data}} \cdot loss_{\text{data}}. \tag{3.4.16}$$

To investigate the influence of numerical resolution on the parameter estimation, the procedure is done twice using two sets of reference solutions with different temporal resolutions. One reference data set is obtained from a numerical simulation with 125 time steps, and another one with 250 time steps. This allows an assessment of how the numerical model's discretization error affects the quality of the estimated parameters. In both cases and for all parameters, the model is trained for 4000 epochs to ensure convergence to stable parameter values.

3.5 Physics-Informed DeepONet Setup

The overall setup of the physics-informed DeepONet is closely related to that of PINNs, with some extensions to account for the operator-learning framework. The architecture consists of a trunk and a branch network (see Figure 3.3). The trunk network, like PINNs, takes the spatial and temporal coordinates (z, t), i.e., the collocation points, as input. The branch network receives a discretized representation of the UB flux function $q(t)$, evaluated at 100 uniformly distributed so-called sensor locations. The network output is a scalar prediction of the matric head ψ, computed as the dot product of the neurons in the last layer of both subnetworks.

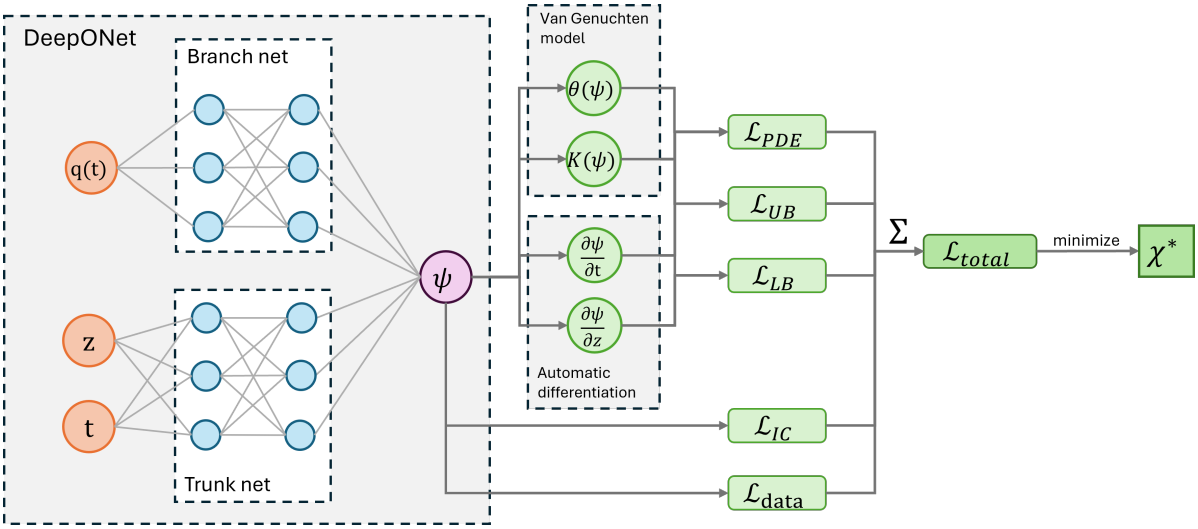

Figure 3.3: DeepONet architecture.

The DeepONet learns a mapping from an input function to the corresponding solution of the governing equation. In this thesis, the network approximates the operator

$$q(t) \mapsto \psi(z, t; q), \tag{3.5.1}$$

where $\psi(z, t; q)$ denotes the matric head corresponding to a given input function $q(t)$.

The same physics-based loss terms for the PDE residual and the initial and boundary conditions as for the PINN setup are employed. In addition, a data loss term based on numerical reference solutions is included, analogous to the inverse PINN case. The total loss function is defined as in (3.4.16). The training data consists of the previously defined sets of collocation points (see Section 3.4.1), as well as sampled reference data for the matric head ψ. The reference data is obtained from the numerical model using the same sampling strategy as in the inverse problem setup, yielding a total of 150 data points.

In this work, the DeepONet is trained on a family of Gaussian-type UB flux functions defined as

$$q(t) = -a \cdot \exp\left(-50(t-b)^2\right), \tag{3.5.2}$$

where the parameters a and b control the amplitude and temporal position of the curve, respectively. More specifically, b determines the time at which the minimum of the curve occurs, i.e., the temporal location of the infiltration peak. The parameter a controls the magnitude of this minimum value, thereby defining the strength of the infiltration flux. Depending on the setup, the model is trained on sets of 3, 5, 7, or 9 such functions with varying parameter values for a and b. During the training process, the model iterates over all defined training input functions in each epoch. For each input function, the loss is evaluated using the corresponding collocation and data points. The total loss for one epoch is then computed as the mean of the losses across all training functions.

To assess the generalization capability of the DeepONet, the trained model is evaluated on unseen UB flux functions, defined by parameter combinations of a and b that were not included in the training set. For both training and test cases, numerical reference solutions

are generated using HYDRUS-1D with the corresponding UB flux functions $q(t)$. The model performance is quantified by comparing DeepONet predictions with these reference solutions using the MSE across all training and test input functions. The training and test MSEs are then compared to assess the model's ability to generalize over the input functions used during training.

Chapter 4

Results

4.1 PINN

This chapter presents the results for the forward and inverse problems using classical PINNs and examines the performance of the applied methods in the defined use case.

4.1.1 Forward Problem

In the forward PINN setup, the model is trained using three different UB conditions, each defined by a function $q(t)$. These functions are shown in Figure 4.1. The first case corresponds to a constant flux of $q = -0.05$ m/day. In addition, two time-dependent flux functions are considered: a sinusoidal flux and a Gaussian-shaped flux.

For each of these UB condition scenarios, the training configuration is adjusted to ensure stable convergence. In particular, the loss weights (cf. (3.4.13)), the number of collocation points for each loss term, and the bias scaling factor β for the residual collocation point

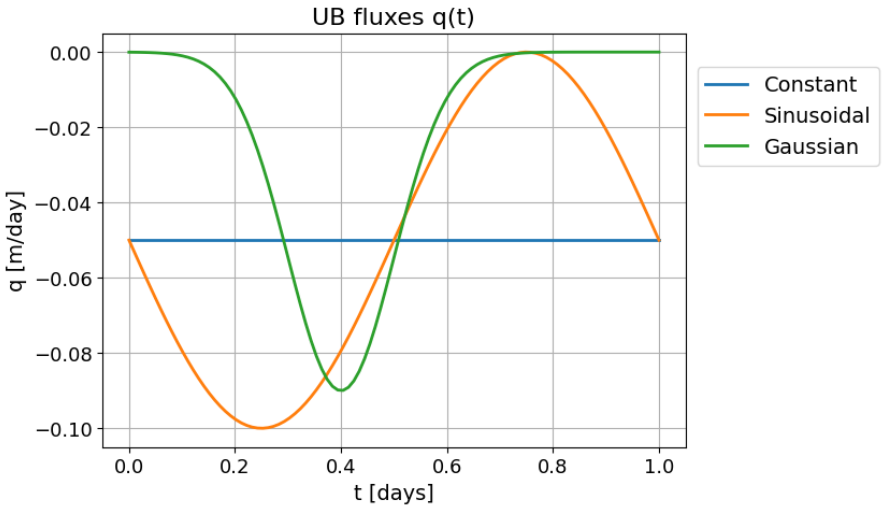

Figure 4.1: Constant, sinusoidal, and Gaussian-shaped UB fluxes, used as UB conditions in the forward PINN setups.

Table 4.1: Hyperparameter configurations of loss weights, collocation points (cf. (3.4.13)) and bias scaling factor β (cf. (3.4.5)) for the three PINN training runs with different UB fluxes. Loss weights and collocation points correspond to the residual, initial condition, upper boundary, and lower boundary terms, respectively. The number of collocation points corresponds to the number of training points.

	Loss Weights				**Collocation Points**				β
UB Flux Type	w_{R}	w_{IC}	w_{UB}	w_{LB}	n_{R}	n_{IC}	n_{UB}	n_{LB}	
Constant	1	10	20	1	8000	100	500	100	1.50
Sinusoidal	1	10	50	1	8000	100	500	100	1.75
Gaussian	1	10	50	1	8000	100	800	100	1.25

samples (cf. (3.4.5)), are varied. The corresponding hyperparameter settings for each scenario are presented in Table 4.1.

The numerical and PINN predictions of the matric head ψ for the three UB flux scenarios are shown in Figure 4.2. For all three cases, the largest differences between the numerical and

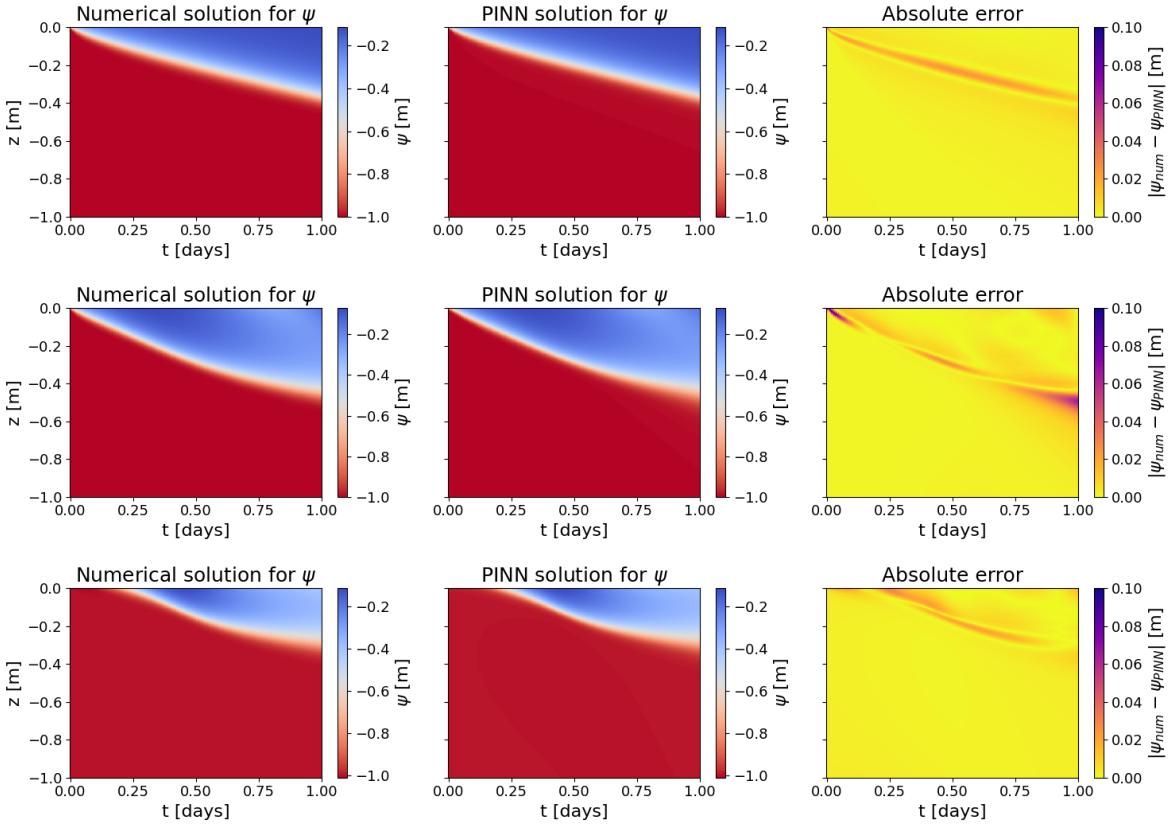

Figure 4.2: Comparison of numerical and PINN predictions of the matric head ψ, along with the corresponding absolute error, for three UB flux conditions: constant flux (top), sinusoidal flux (center), and Gaussian-shaped flux (bottom).

Table 4.2: MSEs between numerical and PINN predictions of the matric head ψ, for three different UB flux conditions.

UB Flux Type	**MSE (m^2)**
Constant	1.9×10^{-5}
Sinusoidal	4.25×10^{-5}
Gaussian	1.56×10^{-5}

PINN solutions occur near the wetting front, where $\psi \approx -0.5$ m. In these regions, the absolute error reaches values of up to 0.05 for the constant and Gaussian-shaped fluxes, and up to 0.08 for the sinusoidal flux. These deviations correspond to relative differences of approximately 10% for the constant and Gaussian cases and up to 16% for the sinusoidal case. However, these larger errors are localized in small regions of the domain, while the overall agreement between the PINN and numerical solutions remains high.

The MSE values, computed across the entire space-time domain, are reported in Table 4.2. The lowest MSE is obtained for the Gaussian flux scenario, followed by the constant flux. The highest MSE is observed for the sinusoidal flux, which is consistent with the larger absolute errors near the wetting front. This indicates that increasing complexity and variability in the UB flux pose additional challenges for the PINN in accurately resolving the solution.

The evolution of the total training and test losses over the epochs is shown in Figure 4.3. For the case with constant UB flux, a divergence between training and test loss is observed after approximately 1000 epochs, with the training loss decreasing more rapidly. However, the test loss continues to decrease throughout training, indicating that the model does not exhibit severe overfitting despite this gap. In contrast, for the sinusoidal and Gaussian UB flux cases, the training and test losses decrease more consistently and remain of similar magnitude over the entire training process.

Among the three scenarios, the lowest test loss after 3000 epochs is achieved for the Gaussian flux case, followed by the constant flux, while the sinusoidal scenario exhibits the highest test loss. This trend is consistent with the MSE results and suggests that lower loss values,

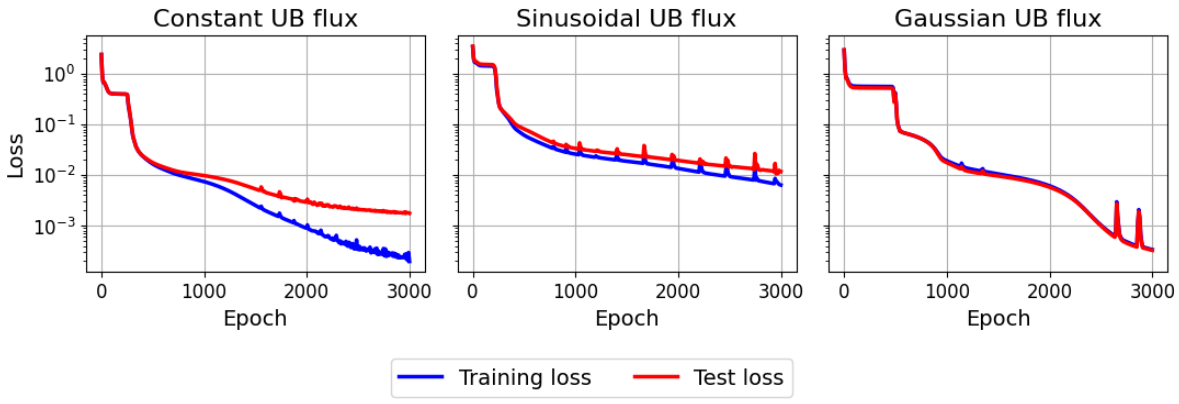

Figure 4.3: Evolution of total training and test losses over the epochs for three different UB fluxes.

Table 4.3: Comparison of run times for the numerical model (HYDRUS-1D) and the trained PINN for different UB flux conditions. The speedup is defined as the ratio of numerical model runtime to PINN runtime.

UB Flux Type	**Numerical (s)**	**PINN (s)**	**Speedup Factor**
Constant	1.06	0.038	27.89
Sinusoidal	1.11	0.04	27.75
Gaussian	1.14	0.043	26.51

indicating better satisfaction of the governing equations and boundary conditions, correspond to more consistent predictions of the PINN with the numerical model.

Computational Efficiency

The different computation times of the numerical model and the trained PINN for the different UB flux scenarios are summarized in Table 4.3. Both the numerical simulation and the PINN evaluations were executed on a CPU. In all three cases, the PINN achieves substantially lower evaluation times, resulting in a speedup of around 27 times compared to the numerical model. Even though the total runtimes of the numerical simulations are relatively short, due to the simplicity of the considered soil infiltration setup, the results clearly demonstrate the superior computational efficiency of PINNs compared to conventional numerical methods. It should be noted, however, that this efficiency advantage applies only after the training phase is complete. On the same CPU-based machine used for the inference time measurements, training the PINN took approximately 5-6 minutes.

Sensitivity Analysis

The sensitivity of the shape and position of the wetting front to variations in the van Genuchten parameters $\alpha, n, \theta_s, \theta_r$ and K_s is shown in Figure 4.4. The analysis is performed using a constant UB flux of $q = -0.05$ m/day. Among the investigated parameters, the shape parameter n exhibits the strongest influence on the wetting front. In particular, a decrease in n leads to a pronounced downward shift of the wetting front, which becomes more significant over time. Furthermore, the wetting front is thinner in that case. An increase in n also results in a slight downward shift, although the effect is less pronounced.

Variations in α primarily affect the spread of the wetting front. Decreasing α leads to a broader wetting front, with infiltration starting at shallower depths at early times and reaching deeper into the soil at later times compared to the baseline case. In contrast, changes in the saturated hydraulic conductivity K_s result in only minor variations in the wetting front, with slightly more noticeable effects for parameter decreases than for increases.

The parameters θ_s and θ_r do not show substantial influence on the shape and position of the wetting front within the considered variation range. Due to this very low sensitivity, θ_s and θ_r are excluded from the inverse parameter estimation. Consequently, only α, n and K_s are considered in that context.

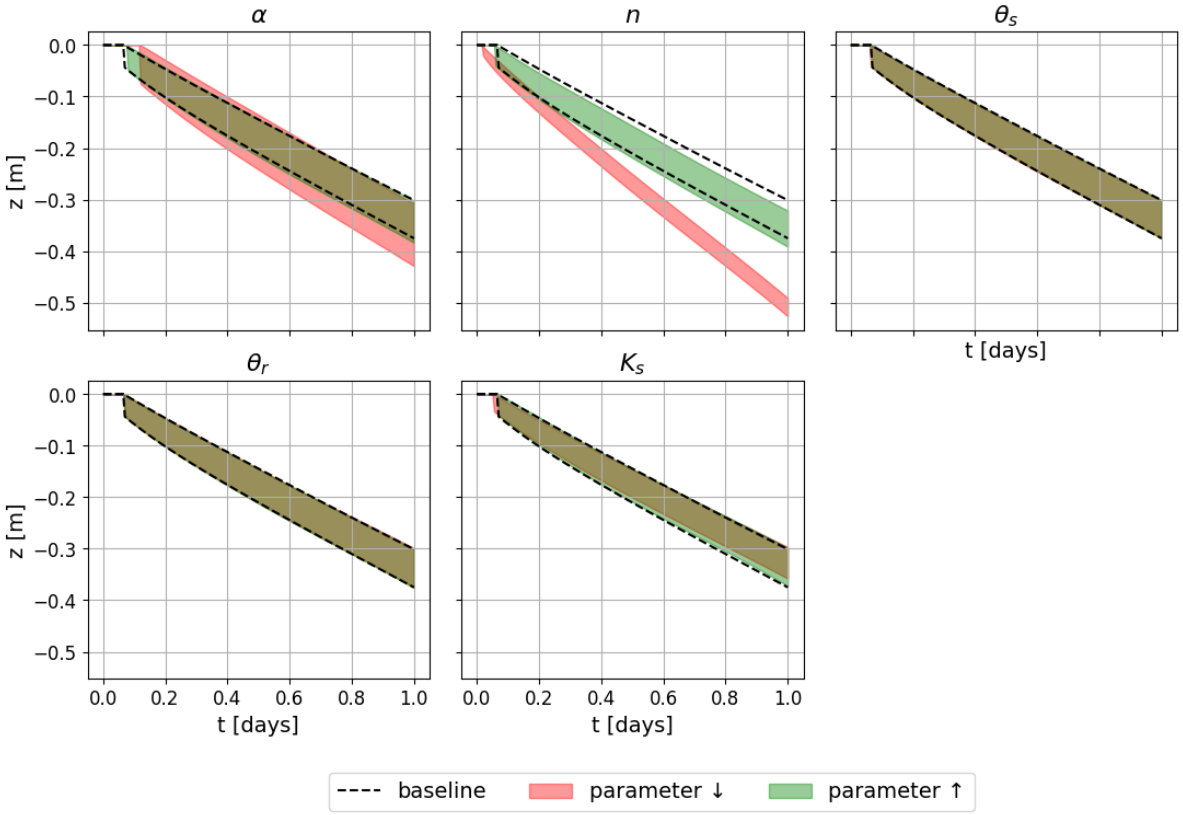

Figure 4.4: Sensitivity of the wetting front to increases and decreases of 20% in van Genuchten parameters, compared to a baseline wetting front.

4.1.2 Inverse Problem

In the inverse parameter estimation setup, a constant UB flux of $q = -0.05$ m/day is assumed. The corresponding training configuration, including loss weights, the number of collocation points, and the bias scaling factor β for the residual points, is defined as given in Table 4.4, accounting for the inclusion of additional data loss terms in this setup.

As described in Section 3.4.2, the three van Genuchten parameters α, n, and K_s are es-

Table 4.4: Hyperparameter configurations of loss weights, collocation points (cf. (3.4.16)) and bias scaling factor β (cf. (3.4.5)) for the inverse PINN setup and the DeepONet. Loss weights and collocation points correspond to the residual, initial condition, upper boundary, lower boundary, and data terms, respectively. The number of collocation points corresponds to the number of training points.

	Loss Weights					**Collocation Points**					β
Model	w_{R}	w_{IC}	w_{UB}	w_{LB}	w_{data}	n_{R}	n_{IC}	n_{UB}	n_{LB}	n_{data}	
Inverse PINN	1	10	20	1	10	8000	100	800	100	150	1.50
DeepONet	20	1	50	1	1	8000	100	800	100	150	1.25

Table 4.5: Estimated van Genuchten parameters obtained using reference data with different temporal resolutions (125 and 250 time steps). The MSE refers to the deviation between PINN predictions and numerical reference solutions for the matric head ψ.

	Estimate		**Ref.**	**MSE**	
Parameter	125	250		125	250
α (m^{-1})	3.538	3.557	3.600	1.3×10^{-5}	1.2×10^{-5}
n (-)	1.565	1.564	1.560	1.9×10^{-5}	1.6×10^{-5}
K_s (m day^{-1})	0.259	0.256	0.2496	9.6×10^{-6}	2.6×10^{-5}

timated individually, resulting in three different trained PINNs. For each parameter, two calculations are conducted using numerical reference data with different temporal resolutions (125 and 250 time steps), which serve as additional training data. The results, presented in Table 4.5, show that even for the lower temporal resolution (125 time steps), the estimated parameters converge close to the reference values, although deviations up to 0.06 in the case of α are observed. When using the higher temporal resolution (250 time steps), all estimated parameters exhibit improved agreement with the reference values, indicating that increased training data resolution enhances parameter estimation performance.

Figure 4.5 visualizes the evolution of the estimated parameters across training epochs. For all three parameters, convergence toward the final estimated values, corresponding to the values after 4000 epochs, is achieved within approximately the first 2500 epochs. Beyond this point, the parameter values remain relatively stable, indicating that the optimization with respect to the parameters has reached a steady state and that the loss function no longer drives significant updates.

The overall model performance in predicting the matric head ψ, quantified by the MSE, is reported in Table 4.5. It should be noted that each parameter was estimated in an individual training setup, resulting in separate trained models and, consequently, separate MSE values for each parameter estimation. In most cases, the MSE is comparable to 1.9×10^{-5}, the value obtained in the forward PINN setup with a constant UB flux. For the parameters α and n, lower MSE values are achieved when higher-resolution numerical reference data is used.

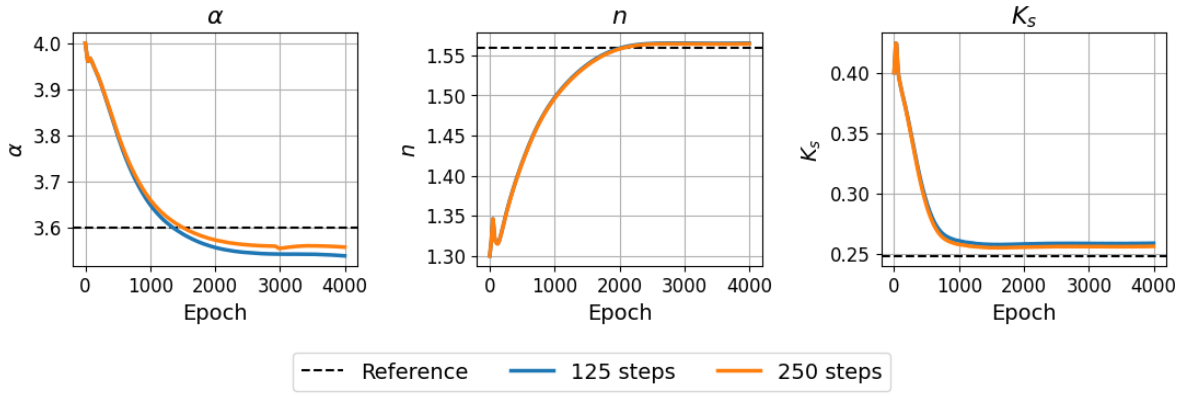

Figure 4.5: Convergence of the estimated parameters over the epochs.

In contrast, for K_s, a slightly higher MSE is observed in the case with increased temporal resolution.

4.2 Physics-Informed DeepONet

In the investigations with the physics-informed DeepONet, the applied hyperparameter configuration, including loss weights, number of collocation points, and the bias scaling factor β, is defined in Table 4.4. As described in Section 3.5, the DeepONet is trained to learn a mapping from UB flux functions $q(t)$, used as input functions, to the corresponding solution $\psi(z, t; q)$. The input functions are defined as Gaussian-shaped fluxes parameterized by a and b (cf. (3.5.2)).

To investigate the generalization capability of the model, four different training setups are considered, varying in the number and distribution of training input functions. In all cases, the model is evaluated on the same set of four test functions. The setups are defined as follows:

- **Setup 1:** Training on 3 input functions
- **Setup 2:** Training on 5 input functions
- **Setup 3:** Training on 7 input functions
- **Setup 4:** Training on 9 input functions

The corresponding training and test functions are shown in Figure 4.6 for each setup. The test functions remain identical across all setups, enabling a consistent comparison of generalization performance. The training and test functions differ by the parameterization of a and b in (3.5.2). Table 4.6 provides an overview of the parameter values used for each function and setup.

After training, the DeepONet is evaluated on both the training functions and the four unseen test functions. The resulting MSEs for each setup are visualized in Figure 4.7, where each point corresponds to a specific training or test input function. The location of the point is determined by the parameters a and b and indicates the minimum of the respective input function. The marker size indicates the magnitude of the MSE.

In the first setup, all three training functions share the same temporal parameter $b = 0.5$. Consequently, the model achieves low MSE values for test functions with the same b, but exhibits significantly higher errors for the test functions with different b values. This indicates limited generalization to unseen temporal positions of the infiltration peak. In the second setup, two additional functions are introduced that match the amplitude parameter $a = -0.08$ of the two test functions with higher MSEs in the first setup. This leads to partial improvement in generalization. The MSE is reduced for one test case ($b = 0.35$) but remains high for the other ($b = 0.65$). Overall, the results in this setup suggest that the model generalizes better when temporal infiltration characteristics are matching (parameter b) than when the amplitude is matching (parameter a).

This observation is further supported in setup 3, where additional training functions are applied at $b = 0.65$. As a result, the MSE for the corresponding test function at that b value decreases substantially. However, the error for the test function at $b = 0.35$ increases again, indicating that the model struggles to generalize in regions of the input space that are underrepresented in the training data. When introducing two additional training functions at

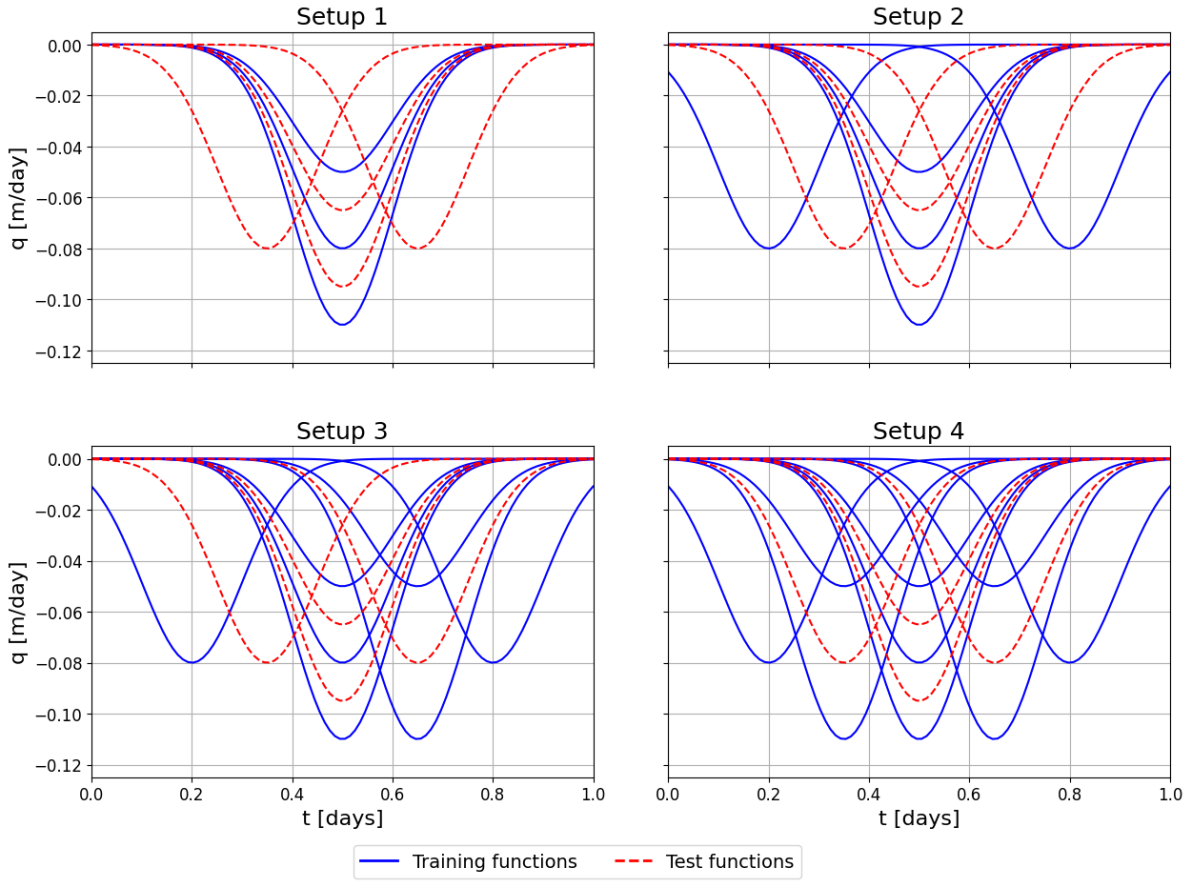

Figure 4.6: Training and test UB fluxes $q(t)$ used in the different setups.

$b = 0.35$ in setup 4, the MSE for the test function at that value of b is reduced again. In this configuration, the model achieves lower, more balanced MSE values across all test functions, demonstrating that a more balanced and dense coverage of the input function space improves generalization. Another observation is that the MSE values for the training functions themselves vary with the position of the function's minimum and across different setups, indicating that the difficulty of learning specific functions depends on their location in the parameter

Table 4.6: Parameterization of Gaussian input functions used for training and testing of the DeepONet. Each column corresponds to one function defined by parameters a and b. The row Setup indicates the experimental setup in which the function is used. All test functions are used for evaluation in every setup.

	Training Functions									**Test Functions**			
	1	2	3	4	5	6	7	8	9	1	2	3	4
a	0.05	0.08	0.11	0.08	0.08	0.05	0.11	0.05	0.11	0.065	0.095	0.08	0.08
b	0.5	0.5	0.5	0.2	0.8	0.65	0.65	0.35	0.35	0.5	0.5	0.35	0.65
Setup	1–4	1–4	1–4	2–4	2–4	3–4	3–4	4	4	1–4	1–4	1–4	1–4

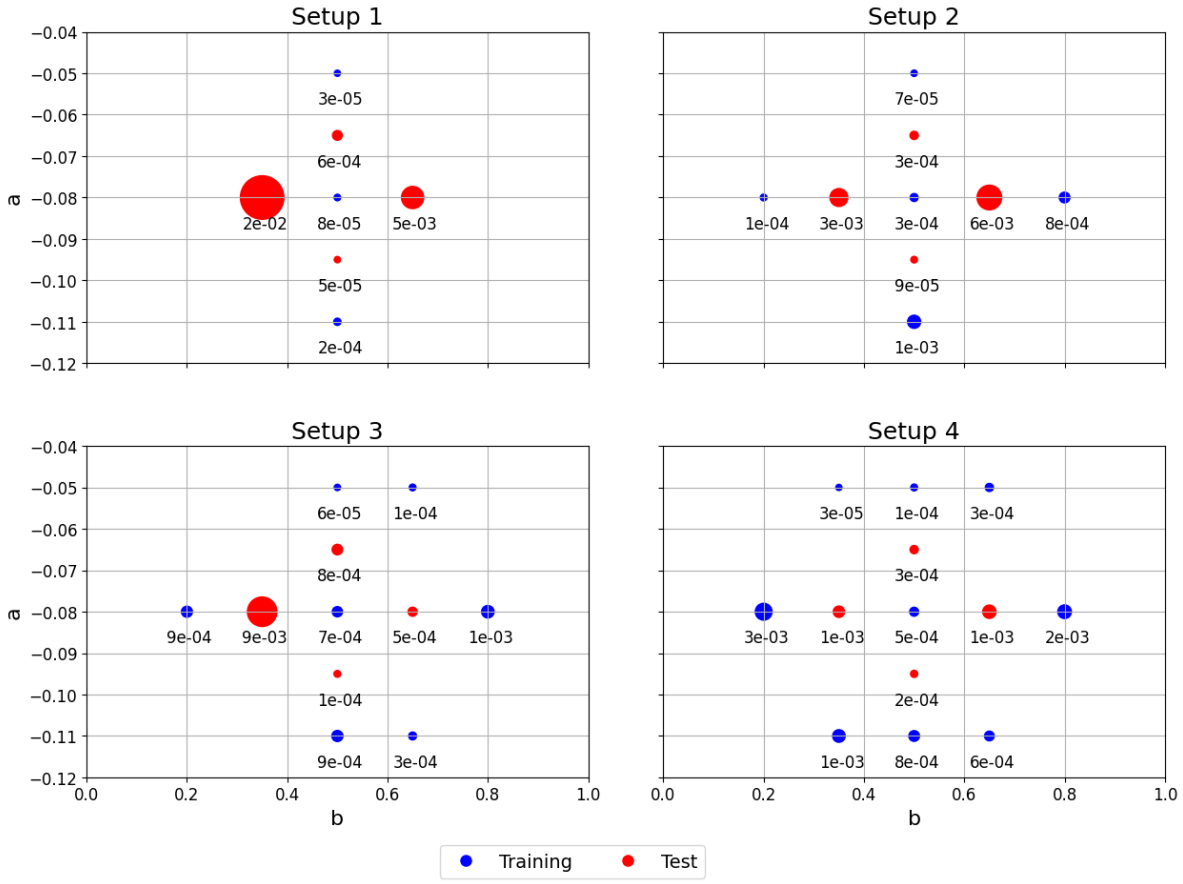

Figure 4.7: MSEs for training and test UB fluxes in different setups.

space.

Overall, the results indicate that the DeepONet generalizes more effectively across variations in amplitude (a) than across temporal shifts (b) of the input functions. To achieve robust generalization over the full input space, a sufficiently dense and balanced set of training functions is required.

Computational Efficiency

To evaluate the computational efficiency of the physics-informed DeepONet, the runtime required to generate predictions for the matric head ψ with the trained model (Setup 1) is measured for the four test input functions. For comparison, the same test flux functions $q(t)$ are used as UB conditions to the numerical HYDRUS-1D model, and the corresponding computation times are recorded. Both the DeepONet predictions and the numerical simulation were executed on a CPU. The average runtimes for predictions across all four test cases are reported in Table 4.7. The physics-informed DeepONet achieves substantially lower evaluation times compared to the numerical model, resulting in a speedup of approximately 16 times. Although this speedup is lower than that observed for the PINN (cf. Tab. 4.3), the DeepONet still demonstrates a significant improvement in computational efficiency during inference compared to the numerical method. However, as in the PINN case, this efficiency advantage

Table 4.7: Comparison of average run times for the numerical model (HYDRUS-1D) and the trained DeepONet for four test input functions. The speedup is defined as the ratio of numerical model runtime to DeepONet runtime.

Numerical (s)	**DeepONet (s)**	**Speedup Factor**
1.107	0.068	16.28

applies only after the training phase is completed. Depending on the number of training input functions, the training time ranged from approximately 13 minutes for the setup with three training functions to around 58 minutes for the setup with nine training functions on a CPU-based machine.

Chapter 5

Discussion

5.1 PINN - Forward Problem

The results obtained in the forward PINN case demonstrate that the model is capable of approximating the solution of Richards' equation with strong alignment to the numerical model, using only physical constraints without relying on paired input-output data. This highlights one of the key strengths of PINNs, namely their ability to incorporate governing equations directly into the training process and to learn physically consistent solutions in a data-scarce setting. It should be noted, however, that the problem setup considered in this thesis is relatively simple. The simulations are restricted to a one-dimensional, homogeneous soil column, and only infiltration processes are considered, neglecting additional complexities such as evaporation, heterogeneous soil properties, or internal sink terms (e.g., root water uptake). While this simplified setting allows for a clear assessment of the PINN's capabilities, it may also contribute to the good performance observed. Extending the approach to more complex and realistic scenarios is likely to increase the difficulty of the learning task.

Especially when steep gradients are involved, a good performance of the PINN is not guaranteed. In the present case, the largest errors occur near the wetting front, indicating challenges in capturing sharp nonlinear transitions. However, such characteristics are also known to be challenging for conventional numerical solvers, where steep gradients often require additional stabilization techniques such as adaptive time-stepping. Even with these interventions, achieving stable and accurate solutions is not always guaranteed. Another important observation is that the training behavior of the PINN is sensitive to changes in boundary conditions. When the UB flux is varied, adjustments to hyperparameters, such as the number of collocation points, loss weights, and the bias factor of residual collocation points, are required to maintain comparable performance. This indicates that PINNs are not entirely robust to changes in the problem setup and may require tuning for different scenarios.

Despite these limitations, the PINN approach offers several advantages compared to conventional numerical methods. In particular, once trained, the model provides significantly faster evaluations, resulting in computational speedups of up to 27 times compared to the numerical solver. Although the training process of a PINN involves a high computational cost, this initial effort is offset by the efficiency during inference. Furthermore, PINNs are mesh-independent, allowing the solution to be evaluated at arbitrary spatio–temporal coordinates within the domain. In contrast, numerical methods rely on predefined discretizations, constraining the solution to a fixed grid.

At the same time, classical numerical models are well established and robust for complex, highly nonlinear problems. Their underlying principles are well understood, and their behavior is generally predictable. In contrast, PINNs retain characteristics of machine learning models, including limited interpretability, as the internal mechanisms that lead to a prediction are not fully transparent. Overall, while PINNs show considerable strengths in terms of computational efficiency during inference, their scalability and robustness in higher-dimensional or more complex settings remain open questions.

5.2 PINN - Inverse Problem

The results of the inverse parameter estimation suggest that the PINN framework is capable of inferring unknown soil parameters from sparse data while enforcing governing physical laws. In all cases considered, the estimated parameters converge to stable values during training, indicating that the optimization process successfully identifies parameter values consistent with both the reference data and the imposed physical constraints.

Although the estimated parameters converge close to the reference values used in the numerical simulations, deviations remain for all three parameters α, n, and K_s. It is observed that the difference between the estimated parameter and the reference value is reduced when higher-resolution numerical data is used for training. This can be explained by the reduced discretization error associated with smaller time steps, resulting in numerical solutions that more closely approximate the true underlying physical process. Consequently, part of the discrepancy in the parameter estimates can be attributed to errors inherent in the numerical reference data itself. In addition, it should be considered that the parameter values identified by the PINN are those that best satisfy the combined constraints of the governing equation, as well as the initial and boundary conditions, in a global sense. This may lead to parameter estimates that differ slightly from the reference values but still yield solutions that are physically consistent and accurate for the predicted state variable ψ.

It is also important to note that the reference parameter values used in this work are derived from the numerical model and do not necessarily represent "true" values for a real soil system. In practical applications, where observational data may be used instead of synthetic reference data, the PINN approach offers a promising alternative for parameter estimation. In such cases, the inferred parameters may provide a meaningful representation of the physical system, particularly for parameters such as α and n, which are usually treated as fitting parameters, and for the saturated hydraulic conductivity K_s, which can be difficult to measure directly. Overall, these results highlight the potential of PINNs for inverse modeling tasks, while also emphasizing the importance of high-quality reference or observational data and the need for careful interpretation of the inferred parameters.

When considering the application of these methods to real-world observational data, a further challenge arises. In practice, soil moisture or matric head measurements are often limited to shallow depths (e.g., within the upper 30 cm of the soil profile) and are typically available at relatively sparse spatial and temporal resolutions compared to the continuous space assumed in the model. On the one hand, this highlights a key advantage of PINNs, as they can incorporate physical constraints to approximate solutions in regions where no data is available. On the other hand, the limited availability and resolution of observational data make both training and, in particular, validation challenging in cases where data is needed. This raises important questions regarding the reliability of the inferred solutions and parameters

when applied to real-world conditions.

5.3 Physics-Informed DeepONet

The results obtained for the physics-informed DeepONet indicate that the model can approximate a mapping from input functions to the corresponding solutions of the Richards equation, provided that a sufficiently broad and representative set of training functions is applied. In particular, the investigations show that the model is able to generalize to unseen input functions to some extent when the training data cover a wide range of functions. A key observation is that the generalization performance strongly depends on the similarity between training and test functions. When test cases exhibit temporal characteristics similar to those in the training set, especially regarding the timing of the infiltration peak, lower errors between the DeepONet and the numerical approximation are achieved. In contrast, test functions with temporal patterns that are not well represented in the training data result in significantly higher errors. This highlights the importance of sufficient coverage of the input function space, especially with respect to temporal variability.

However, compared to the forward PINN results, the predictive accuracy (in terms of the similarity to the numerical prediction) of the DeepONet is notably lower. While the PINN achieves MSE values on the order of 10^{-5}, the DeepONet typically yields MSE values between 10^{-3} and 10^{-4} with only a small number of cases approaching the accuracy of the PINN. This indicates that in the present setup, the physics-informed DeepONet is less effective in accurately reproducing the solution of the governing equation. One possible explanation for these higher errors is the limited coverage of the input function space provided by the training data. Since the DeepONet is designed to learn an operator mapping from input functions to solutions, insufficient representation of this function space can restrict its generalization capability and lead to increased prediction errors.

This observation reflects a fundamental trade-off between PINNs and physics-informed DeepONets. While PINNs are designed to solve a single problem instance with high accuracy, DeepONets aim to learn a family of solutions across varying input functions. This broader generalization capability comes at the cost of reduced accuracy for individual instances, particularly when the training data is limited, as in the present case.

It is important to note that the experimental setup applied here is relatively limited. Only a single class of input functions (Gaussian-type fluxes) is used, and the number of training functions remains small. While this setup allows for a controlled investigation of the model's behavior, it also restricts its ability to learn a more general operator. Given the observed results, it is questionable whether the current approach can be effectively applied to more complex scenarios involving a wider variety of input functions or more realistic conditions, particularly when considering noisy observational data.

DeepONets were originally proposed as data-driven models designed to learn from large datasets (Lu et al., 2021). The physics-informed variant considered in this thesis combines data and physical constraints but may still require a sufficiently rich dataset to achieve higher accuracy. A purely data-driven DeepONet approach could potentially improve performance, but would necessitate the generation of large amounts of training data, for example through extensive generation of numerical approximations. The primary objective of this thesis, however, is to investigate physics-informed approaches that can work effectively with limited data while maintaining consistency with underlying physical laws. Given this context, the Deep-

ONet approach, as implemented here, offers some potential for operator learning but does not yet achieve the same level of robustness as the PINN in this low-data, physics-constrained setting.

Beyond the domain of physics-informed approaches investigated in this thesis, another promising class of operator learning methods is given by Fourier Neural Operators (FNOs) (Li et al., 2021). These models aim to learn mappings between function spaces in a purely data-driven setting and have gained attention for solving PDEs efficiently in recent years. FNOs are based on the idea of performing computations in the Fourier space. In a typical Fourier layer, the input function is first transformed using a fast Fourier transform, followed by a linear transformation applied to the Fourier coefficients. The result is then mapped back to the physical domain with an inverse Fourier transform. This procedure contributes to the high computational efficiency of the method.

A key difference between FNOs and DeepONets lies in their discretization properties. While DeepONets rely on a fixed discretization of the input function (through sensor points), FNOs are discretization-invariant (Kovachki et al., 2021). This means that they can operate on input functions defined on different grids or resolutions without requiring architectural changes. This property is particularly useful in real-world applications, where data might be available at varying resolutions or irregular sampling patterns.

Besides that, a major advantage of FNOs is that they have been shown to achieve lower errors relative to numerical methods compared to other deep learning-based operator learning methods. For instance, Kovachki et al. (2021) demonstrated that FNOs can outperform DeepONets in terms of relative error across different benchmark problems. One reason for this improved performance is that FNOs are considered nonlinear operator approximation methods that combine integral operators with nonlinear activation functions. In contrast, DeepONets are characterized as linear approximation methods, as they approximate operators through a linear combination of learned basis functions through the trunk net. This makes FNOs more expressive and better suited for capturing complex, nonlinear dynamics.

Despite these advantages, FNOs typically require large amounts of training data to achieve high accuracy (Li et al., 2021), which can be a significant drawback in applications where data is scarce or expensive to generate. Nevertheless, FNOs as a whole represent a promising alternative for solving PDEs using deep learning. Their strong performance and flexibility suggest that they could be a valuable direction for future research within the application of unsaturated soil water flow.

Chapter 6

Conclusion

In this thesis, two physics-informed machine learning approaches, PINNs and physics-informed DeepONets, were investigated for approximating solutions of Richards' equation in terms of the matric head ψ to model unsaturated soil water flow. First, PINNs were applied to a simplified one-dimensional, homogeneous soil setup to solve a single problem instance of the Richards equation. The resulting solutions were compared to numerical reference solutions obtained from HYDRUS-1D. Within this framework, the computational efficiency of the trained PINN during inference was evaluated, and a sensitivity analysis of the van Genuchten parameters was conducted. Furthermore, an inverse PINN approach was employed to estimate the soil hydraulic parameters α, n and K_s.

Subsequently, a physics-informed DeepONet was implemented with the aim of learning a solution operator for Richards' equation that generalizes across UB conditions. In this context, the model was trained on Gaussian-type UB flux functions and evaluated on unseen input functions. In addition, its computational speed during inference was compared with that of the numerical model.

Based on the obtained results, the answers to the research questions can be summarized as follows:

(i) *How well do physics-informed neural networks reproduce numerical solutions of Richards' equation?*
In the considered one-dimensional homogeneous soil setup, the PINN achieved relatively low MSE values when compared to numerical reference solutions, indicating a strong overall alignment. The largest errors are observed near the wetting front, where steep hydraulic gradients occur.

(ii) *How well can PINNs estimate soil hydraulic parameters of the van Genuchten model?*
The inverse PINN setup successfully identified parameter values that converged to stable solutions during training. While the estimated parameters were close to the reference values, slight deviations remained. Further validation using observational data would be required to assess the reliability of the inferred parameters in real-world applications.

(iii) *Can physics-informed DeepONets learn the solution operator of Richards' equation such that accurate predictions can be made for unseen upper boundary conditions?*
In the simplified setup considered in this thesis, the physics-informed DeepONet was able to learn an operator that generalizes to unseen Gaussian-type upper boundary fluxes,

provided that a sufficiently wide range of training functions is available. However, the achieved accuracy was lower than that of the PINN.

Despite these results, several limitations must be acknowledged. First of all, the problem setup considered is highly simplified, as water flow is modeled in a single spatial dimension and the soil is assumed to be homogeneous. Furthermore, additional processes such as evaporation and internal sink terms are neglected. Moreover, all calculations were conducted using synthetic data derived from numerical simulations, and no validation with real-world observational data was performed. In the case of the DeepONet, only a single class of input functions and a limited number of training samples were considered, restricting its ability to fully capture the underlying operator.

Future work could focus on extending the investigated approaches to more complex and realistic scenarios, including higher-dimensional soil systems, heterogeneous soil conditions, and the incorporation of observational data. In addition, alternative operator learning methods, such as FNOs, could be explored as a data-driven approach to learn the mapping from UB flux functions to the corresponding solutions of Richards' equation.

Overall, this thesis demonstrates that physics-informed machine learning methods provide a promising framework for solving forward and inverse problems in unsaturated soil water flow. While PINNs show strong performance for single-instance problems with limited data, operator learning approaches such as DeepONets offer potential for generalization across varying boundary conditions; however, they incur higher relative errors. Compared with classical numerical methods, both approaches offer substantial advantages in computational efficiency and flexibility, particularly due to their mesh-independent nature. Further research is required to fully exploit the strengths of these methods and to assess their applicability in more realistic hydrological modeling, especially for operator learning approaches.

Bibliography

Bittelli, M., Campbell, G. S., and Tomei, F. (2015). *Soil Physics with Python: Transport in the Soil–Plant–Atmosphere System*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199683093.001.0001.

Bonan, G. (2015). *Ecological Climatology: Concepts and Applications*. Cambridge University Press, Cambridge, 3 edition. https://doi.org/10.1017/CBO9781107339200.

Buckingham, E. (1907). Studies on the movement of soil moisture. *US Dept. Agic. Bur. Soils Bull.*, 38.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88. https://doi.org/10.1007/s10915-022-01939-z.

Da Silva, A. L., Reichardt, K., Roveratti, R., Bacchi, O. O., Timm, L. C., Oliveira, J. C. M., and Dourado-Neto, D. (2007). On the use of soil hydraulic conductivity functions in the field. *Soil and Tillage Research*, 93(1):162–170. https://doi.org/10.1016/j.still.2006.03.024.

Depina, I., Jain, S., Mar Valsson, S., and Gotovac, H. (2022). Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 16(1):21–36. https://doi.org/10.1080/17499518.2021.1971251.

Elkhadrawi, M., Ng, C., Bain, D. J., Sargent, E. E., Stearsman, E. V., Gray, K. A., and Akcakaya, M. (2024). Novel physics informed-neural networks for estimation of hydraulic conductivity of green infrastructure as a performance metric by solving Richards–Richardson PDE. *Neural Computing and Applications*, 36(10):5555–5569. https://doi.org/10.1007/s00521-023-09378-z.

Faroughi, S. A., Pawar, N. M., Fernandes, C., Raissi, M., Das, S., Kalantari, N. K., and Kourosh Mahjour, S. (2024). Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics. *Journal of Computing and Information Science in Engineering*, 24(040802). https://doi.org/10.1115/1.4064449.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hillel, D. (1998). *Environmental Soil Physics: fundamentals, applications, and environmental considerations*. Academic Press, San Diego, CA.

Ippisch, O., Vogel, H. J., and Bastian, P. (2006). Validity limits for the van Genuchten–Mualem model and implications for parameter estimation and numerical simulation. *Advances in Water Resources*, 29(12):1780–1789. https://doi.org/10.1016/j.advwatres.2005.12.011.

Kamil, H., Soulaïmani, A., and Beljadid, A. (2025). A comparative study of physics-informed neural network strategies for modeling water and nitrogen transport in unsaturated soils. *Journal of Hydrology*, 661:133624. https://doi.org/10.1016/j.jhydrol.2025.133624.

Kandelous, M. M. and Šimůnek, J. (2010). Numerical simulations of water movement in a subsurface drip irrigation system under field and laboratory conditions using HYDRUS-2D. *Agricultural Water Management*, 97(7):1070–1076. https://doi.org/10.1016/j.agwat.2010.02.012.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440. https://doi.org/10.1038/s42254-021-00314-5.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Neural Operator: Learning Maps Between Function Spaces. arXiv:2108.08481v6 [cs]. https://arxiv.org/abs/2108.08481v6.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895 [cs]. http://arxiv.org/abs/2010.08895.

Lipiec, J., Usowicz, B., and Siczek, A. (2024). Fitting the van Genuchten model to the measured hydraulic parameters in soils of different genesis and texture at the regional scale. *International Agrophysics*, 38(4):373–382. https://doi.org/10.31545/intagr/191380.

Lu, L., Jin, P., and Karniadakis, G. E. (2021). DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229. arXiv:1910.03193 [cs]. http://arxiv.org/abs/1910.03193.

Meedeniya, D. (2023). *Deep Learning: A Beginners' Guide*. Chapman and Hall/CRC, New York. https://doi.org/10.1201/9781003390824.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (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/10.1016/j.jcp.2018.10.045.

Rassam, D. W., Šimůnek, J., Mallants, D., and van Genuchten, M. T. (2018). *The HYDRUS-1D Software Package for Simulating the Movement of Water, Heat, and Multiple Solutes in Variably Saturated Media*. Australia.

Šimůnek, J., Van Genuchten, M. T., and Šejna, M. (2008). Development and Applications of the HYDRUS and STANMOD Software Packages and Related Codes. *Vadose Zone Journal*, 7(2):587–600. https://doi.org/10.2136/vzj2007.0077.

Šimůnek, J., Šejna, M., Brunetti, G., and van Genuchten, M. T. (2025). *The HYDRUS Software Package for Simulating One-, Two-, and Three-Dimensional Movement of Water, Heat, and Multiple Solutes in Variably-Saturated Porous Media*. Prague, Czech Republic.

Wang, S., Wang, H., and Perdikaris, P. (2021). Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605. https://doi.org/10.1126/sciadv.abi8605.

Zhang, L., Wu, F., Zheng, Y., Chen, L., Zhang, J., and Li, X. (2018). Probabilistic calibration of a coupled hydro-mechanical slope stability model with integration of multiple observations. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 12(3):169–182. https://doi.org/10.1080/17499518.2018.1440317.

Master's Theses in Mathematical Sciences 2026:E50
ISSN 1404-6342

LUNFBV-3009-2026

Computational Science
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/