Physics-Informed Neural Networks: Solving & Discovering Charge Dynamics in Gaseous High Voltage Insulation

Exploring the use of PINNs for Forward and Inverse Problems within Charge Dynamics in Air Insulation

Master's thesis in Data Science and AI

CARL-JOHAN BJÖRNSON
FELIX ÅGREN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

MASTER'S THESIS 2024

Physics-Informed Neural Networks: Solving & Discovering Charge Dynamics in Gaseous High Voltage Insulation

Exploring the use of PINNs for Forward and Inverse Problems within Charge Dynamics in Air Insulation

CARL-JOHAN BJÖRNSON
FELIX ÅGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Physics-Informed Neural Networks: Solving & Discovering Charge Dynamics in Gaseous High Voltage Insulation
Exploring the use of PINNs for Forward and Inverse Problems within Charge Dynamics in Air Insulation
CARL-JOHAN BJÖRNSON, FELIX ÅGREN

Supervisor: Olof Hjortstam, Hitachi Energy Research & Department of Electrical Engineering
Supervisor: Christian Häger, Department of Electrical Engineering
Supervisor: Thomas Hammarström, Department of Electrical Engineering
Examiner: Yuriy Serdyuk, Department of Electrical Engineering

Master's thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Physics-Informed Neural Networks: Solving & Discovering Charge Dynamics in Gaseous High Voltage Insulation
Exploring the use of PINNs for Forward and Inverse Problems within Charge Dynamics in Air Insulation
CARL-JOHAN BJÖRNSON, FELIX ÅGREN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The development of efficient high-voltage equipment is imperative for minimizing greenhouse gas emissions and saving costs within the energy system. Effective insulation plays a pivotal role in such development and requires an understanding of the performance of gaseous insulators, such as air, under high-voltage stress. Electric discharges and charge transport in gases are modeled using systems of partially differentiable equations (PDEs) and their solutions are traditionally approximated numerically with discretizing methods such as the finite element method (FEM). However, such methods have significant shortcomings including difficulty handling high-dimensional problems, non-smooth behaviors, and inverse problems with hidden physics.

An emerging, mesh-free alternative to numerical methods is physics-informed neural networks (PINNs). PINNs solve PDEs using a neural network with the PDE and associated constraints embedded into the network's loss function and are easily extended to inverse problems. Initial experiments with PINNs for the forward problems related to electric discharges and charge dynamics have shown promising advantages compared to FEM but failed to model strongly non-uniform functions and coupled equations within the domain. This thesis contributes to this research by showing how a variety of performance-enhancing techniques can address the weaknesses of previous works, improving accuracy and enabling the modeling of steeper gradients and coupled PDEs. Additionally, it demonstrates how PINNs can be used to solve inverse problems related to discharge and charge dynamics, discovering both unknown parameters and distributions.

Keywords: Physics-Informed Neural Networks, Machine learning, Discharge Physics

Acknowledgements

First and foremost, we would like to thank our team of supervisors, Olof Hjortstam, Christian Häger, Thomas Hammarström, and our examiner Yuriy Serdyuk for going above and beyond what one might expect from a supervisor. Your guidance, availability, and expertise have been pivotal to the success of our thesis. We would also like to express our gratitude to Hitachi Energy Research for making this project possible.

Carl-Johan Björnson, Felix Ågren, Gothenburg, January 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

FEM	Finite Element Method
GELU	Gaussian Error Linear Unit
HV	High Voltage
MSE	Mean Squared Error
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
RWF	Random Weight Factorization
ReLU	Rectified Linear Unit
tanh	Hyperbolic Tangent
MLP	Multilayer Perceptron

Contents

List of Figures

List of Tables

1
Introduction

The world's energy systems are changing in response to climate change. With increasing electrification and soaring demand for renewable energy, the importance of an efficient power transmission system is greater than ever. Thus, the development of efficient high-voltage equipment is imperative both for combating climate change and achieving cost savings throughout the energy system.

Effective insulation plays a pivotal role in such development and requires an understanding of the performance of gaseous insulation, e.g., air, under high-voltage stress. Charge dynamics such as moving electric discharges and corona discharges are particularly interesting for high-voltage power transmission. If the electric field close to the power line grows sufficiently strong (i.e., exceeds the discharge initiation strength of the surrounding insulation media, typically air) the rate of ionization in the surrounding medium exceeds the rate of recombination with exceedingly high concentrations of charge carriers as a result. These charge carriers will begin to drift (i.e., move) in the electric field and the kinetic energy required for them to do so will be sourced from the power grid, resulting in a loss of energy referred to as corona power loss [1]. Moreover, such electric discharges may also cause the formation of harmful gases such as ozone and NOx, noise, and insulation damage [2], [3].

Physically, these crucial electrical processes for insulation performance are described through partially differentiable equations (PDEs) and the modeling of related phenomena for different geometries, external conditions, and voltages is done by computer simulations based on their solutions [4]. PDEs play a crucial role in mathematical modeling within multiple fields and have been successfully used outside of physics in fields such as biology [5], finance [6], and sociology [7].

In general, PDEs lack closed-form solutions, requiring them to be solved numerically to obtain approximative solutions. Traditionally, numerical solutions are obtained using mesh-based methods that discretize the spatial domain, the foremost of which is the finite element method (FEM) [8], [9]. These methods are well-researched, resulting in error estimations, convergence guarantees, and easy-to-use solver software, such as COMSOL Multiphysics, being readily available [10], [11].

Nevertheless, the classic discretizing methods, such as FEM, have notable shortcomings. Firstly, the method of discretizing the spatial domain with a sufficiently fine mesh suffers from the curse of dimensionality and quickly becomes intractable for high-dimensional problems. Secondly, it is difficult to discretize highly nonlinear or non-smooth functions effectively due to the highly granular mesh required to capture such behaviors [11]. Thirdly, the methods can struggle with solving ill-posed inverse problems with hidden physics [12]. For the domain of discharge physics, the

volatile nature of the problems tends to present with steep gradients, resulting in nonlinear behavior, and the modeling of hidden physics is highly interesting.

An emerging, mesh-free alternative to traditional PDE solvers is physics-informed neural networks (PINNs). The technique is based on embedding a PDE and associated constraints into the loss function of a deep neural network, allowing training in a self-supervised fashion based on input-output pairs generated from the PDE [13]. Interestingly, PINNs have shown potential in addressing the challenges of conventional methods. For instance, neural networks have proven capable of representing strong nonlinearities in complex systems and PINNs offer differentiable outputs while avoiding mesh-related challenges of tricky problems and domains [11]. Furthermore, PINNs can easily be extended to incorporate observational data to discover hidden physics (i.e., solving the so-called inverse problem) with minimal code modification and enforce hard constraints [14].

While PINNs have shown encouraging results in a variety of domains, including quantum chemistry [15], electromagnetics [16] and molecular simulation [17], their applicability within the domain of discharge physics is relatively unexplored. Initial experiments with PINNs for the forward problem of electric field-driven charge transport in gas discharges have been conducted on Laplace's equation and the drift-diffusion equation in simple geometries [18]. In these, PINNs showed promising advantages compared to FEM but failed to model strongly non-uniform concentration profiles of charge carriers, sufficiently narrow electrodes for steamer simulation, and coupled equations for potential and charge distributions [18]. Konradsson [18] highlights the need for additional research into improvements of the forward problem within these areas as well as an initial exploration of PINNs for the inverse problem of discovering hidden physics within these equations.

1.1 Purpose

This thesis aims to contribute to the understanding of the applicability of PINNs in the domain of discharge physics by extending proposed solutions for the forward problem to include tougher models. In particular, it aims to provide techniques for solving strongly non-uniform concentration profiles for charge carriers and potential fields as well as solving coupled equations for the potential and charge distributions. Moreover, the thesis aims to explore how PINNs can be used to solve inverse problems of discovering hidden physical parameters and distribution within these problems.

The above purpose can be condensed into four research questions relating to the applicability of PINNs for Poisson's equation and the drift-diffusion equation within the domain of discharge physics:

1. How can the PINN implementation proposed in Konradsson [18] be improved to better model steep gradients and strong nonlinear concentration profiles?
2. How can the PINN implementation proposed in Konradsson [18] be improved to enable solving Poisson's equation coupled with the drift-diffusion equation?
3. Can PINNs solve inverse problems of discovering hidden physical parameters or distributions related to Poisson's equation and the drift-diffusion equation?

4. What impact do fewer and noisier observations have on the accuracy of solutions for the inverse problem?

1.2 Limitations

The proposed formulations of the PDEs in this thesis will contain several simplifications that prevent their solutions from being directly applicable to any real-world problems. Firstly, the physical models will be limited to one dimension. Secondly, only the drift and diffusion of positive ions will be considered. Ion generation and recombination processes will be ignored. Thirdly, observations of the modeled functions' values will be used for the inverse problems, neglecting the fact that these are very tedious to be directly obtained experimentally (e.g., measurements of the electric potential in an air gap).

These simplifications serve the purpose of limiting the physical complexity and allowing more time spent developing the PINNs and related techniques. Furthermore, they align the results with those of Konradsson [18] for readily available benchmarks and comparisons.

2
Theory

This chapter provides a brief introduction to discharge physics and its relevance for HV equipment design. Additionally, it covers the basics of neural networks for function approximation and introduces the concept of physics-informed neural networks.

2.1 Discharge Physics

High-voltage transmission lines generate electrical fields around power conductors that if strong enough can ionize the surrounding air [19]. The resulting electrical discharge can result in local enhancements of the electric field which impacts the performance of the insulation and may cause electric discharges including Townsend's, glow, corona, and arc discharges [2], [20]. Electric discharges are widely occurring in high-voltage (HV) power transmission lines and they are undesirable as they lead to energy losses, formation of harmful gases such as ozone and NOx, noise, and insulation damage [2], [3]. Thus, being able to model and predict electrical discharges is crucial for the design of HV equipment.

Corona discharges can occur in highly non-uniform electrical fields between one grounded electrode and one electrode subject to high potential. They are characterized by crackling sounds and glowing effects and are caused by the electrical field ionizing the air close to the high potential electrode [3].

Air consists of small densities of free electrons stemming from ionization by natural processes such as background radiation and thermal excitation [2]. The strong electrical field accelerates the free electrons causing collisions with gas atoms or molecules in which an additional electron can be knocked off, resulting in the formation of two additional charge carriers, one positive ion and one electron:

$$e^- + A \rightarrow A^+ + 2e^-$$

These newly formed electrons will also accelerate through the electrical field and collide with neutral atoms and molecules, causing further ionization and exponentially increasing free electrons [2]. This effect is referred to as an electron avalanche. As the electron avalanche progresses, the increasingly ionized air can form plasma channels with a sufficiently high density of charge carriers for the previously insulating air to become an excellent conductor of electricity, giving rise to discharges such as glowing and arcs.

Finally, electron avalanches give rise to discharges of positive ions moving toward the cathode and electrons moving toward the anode. If the avalanches are strong

enough, the resulting aggregations of charges in a particular part of space significantly alter the surrounding electrical field at the head of the wave causing further ionization. This results in self-propagating so called streamers [2].

2.2 Deep Neural Networks

A neural network is constructed from elementary nodes designed to compute a weighted sum of multiple inputs, apply a nonlinearity, and generate a real-valued output. Figure 2.1 illustrates the schematic representation of such a neuron, and the mathematical formulation is as follows,

$$\begin{aligned} z &= w^T x + b \\ a &= \phi(z) \end{aligned} \tag{2.1}$$

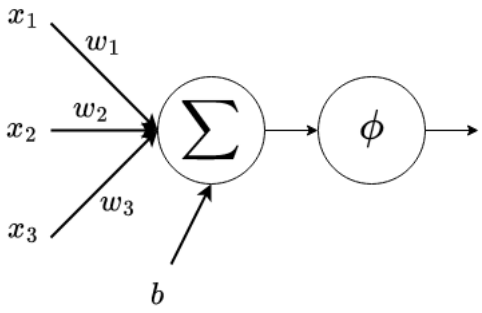

Figure 2.1: Schematic of a single neuron which computes a weighted sum of the inputs x_1, x_2, x_3, and produces an output by applying a non-linearity ϕ.

Here, x and w are vectors representing the input values and the associating weights. The symbol b is the bias term, one per neuron, and $\phi(\cdot)$ represents the non-linear function, or namely the neurons' activation function. There exists a variety of activation functions for different applications, but the most common ones are the *sigmoid*, *hyperbolic tangent* (tanh), and *rectified linear units* (ReLU) [21].

A neural network is created by arranging the neurons in a layered structure and passing the output from the previous layer as input for the subsequent layer. If the network has multiple hidden layers, it is called a deep neural network. A simple feed-forward neural network is depicted in Figure 2.2 with two input neurons, two hidden layers with three neurons each, and one output layer consisting of a single neuron.

For the formulation of a neural network, consider the following notation. Let W_k denote the weights connecting layer $k-1$ with layer k. The elements of the weight matrix W_k are denoted $w_{k,ij}$, each referring to the weight connecting the output from the i-th neuron in layer $k-1$ with the j-th neuron in layer k. The different layers are indexed as $k = 1, ..., N_L$, where N_L is the total number of layers. Note that the input layer simply contains the network input, in other words, no weighted sum is calculated in this layer. Therefore the input layer is indexed by $k = 0$. The output passed through the whole network, can recursively be defined as

$$
\begin{aligned}
a_0 &= x, \\
z_k &= W_k^T a_{k-1} + b_k, \quad k = 1, \ldots, N_L, \\
a_k &= \phi(z_k), \quad k = 1, \ldots, N_L, \\
a_{N_L} &= \hat{y}
\end{aligned}
\tag{2.2}
$$

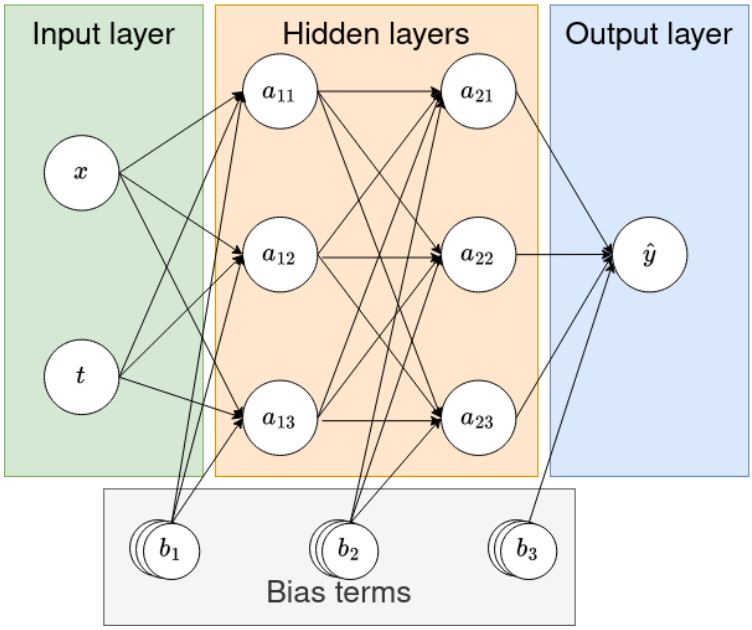

Figure 2.2: Schematic picture of a Fully-connected Neural Network.

2.2.1 Training a Neural Network

The previous section defines how a fully-connected neural network takes an input vector x and produces an output vector $\hat{y}$. This mapping of input data and output data is essentially a parameterized function with real-valued input space of dimension n and real-valued output space of dimension m,

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m \tag{2.3}$$

The parameters of the function is the neural networks weights and biases, and the goal of training a neural network is to optimize the parameters θ such that the function maps the input data to the desired output. This requires a dataset consisting of input and output pairs (x, y).

This setup of training a neural network, or any machine learning model in general, with labeled data is called supervised learning. By having a label associated with each data sample in the dataset it can be easily measured how close the networks output is to the true value described by the label. The step of feeding data through the network and calculating the output values during training is called the forward pass.

After executing the forward pass, predictions $\hat{y}$ are obtained for each input vector x. With a true value y present for each x in the dataset, the model performance can

be assessed using the current parameters θ. This evaluation involves calculating the loss. For regression problems, a commonly used loss function is the mean squared error described in Equation 2.4, where the error is computed based on the samples n.

$$\mathcal{L}_\theta(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.4}$$

By evaluating the performance of the current network parameters using the specified loss function, it becomes feasible to attain optimal parameters through the minimization of the said loss function.

$$\theta^* = \arg\min_\theta \mathcal{L}_\theta(y, \hat{y}) \tag{2.5}$$

However, considering that deep neural network models typically comprise a substantial number of parameters, an efficient approach is necessary to compute each parameter's impact on the overall loss. This is achieved by computing the partial derivatives of the loss with respect to each of the network parameters θ, which reveals the sensitivity of the loss concerning adjustments to each network parameter. After conducting the forward pass and calculating the loss, the partial derivative for each parameter is computed using the chain rule of calculus listed in Equation 2.6, starting from the end of the network and propagating backwards. Hence the name backpropagation.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z^{[l-1]}} &= [W^{[l]}]^T \cdot \frac{\partial \mathcal{L}}{\partial z^{[l]}} \cdot \phi^{[l-1]\prime}(z^{[l-1]}) \\
\frac{\partial \mathcal{L}}{\partial W^{[l]}} &= \frac{\partial \mathcal{L}}{\partial z^{[l]}} \cdot [a^{[l-1]}]^T \\
\frac{\partial \mathcal{L}}{\partial b^{[l]}} &= \frac{\partial \mathcal{L}}{\partial z^{[l]}}
\end{aligned} \tag{2.6}$$

Once all gradients detailing the impact of each parameter on the computed loss are available, the parameters can be adjusted to minimize the loss. However, the substantial number of parameters often gives rise to a highly non-convex optimization problem. Consequently, iterative gradient descent-based algorithms are the preferred choice due to their computational efficiency and their ability to leverage the derivatives computed during backpropagation. One iteration of the gradient descent algorithm is described in Equation 2.7.

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}(\theta_t) \tag{2.7}$$

In this context, θ_{t+1} represents the updated set of model parameters, and α is the hyperparameter which controls the step length during optimization, commonly referred to as the learning rate. Determining an appropriate learning rate is a non-trivial task, given its impact on the updating process of the network's weights and biases. The objective is to navigate towards a global minimum, or at least a satisfactory local minimum. Selecting a learning rate that is too high may hinder convergence, causing θ to oscillate back and forth and potentially skip over the

minimum. On the other hand, opting for a learning rate that is too small results in slow convergence and increases the risk of getting stuck in a local minimum. The delicate balance in setting the learning rate is essential for achieving effective and efficient optimization in neural network training.

These four steps described above of conducting the forward pass, calculating the loss, backpropagation, and model parameter updates constitute one training iteration for a neural network. This procedure is then repeated until the loss satisfies a specific threshold, or until the maximum number of iterations is reached.

2.2.2 Activation Functions and Gradient Descent Based Methods

As previously mentioned, there exists a range of different activation functions, each with their specific characteristics and different pros and cons. The activation function serves the crucial role of introducing non-linearities to the neural network, allowing it to learn more complex solutions. Three of the most common ones are displayed in Figure 2.3, namely sigmoid, tanh, and ReLU [21].

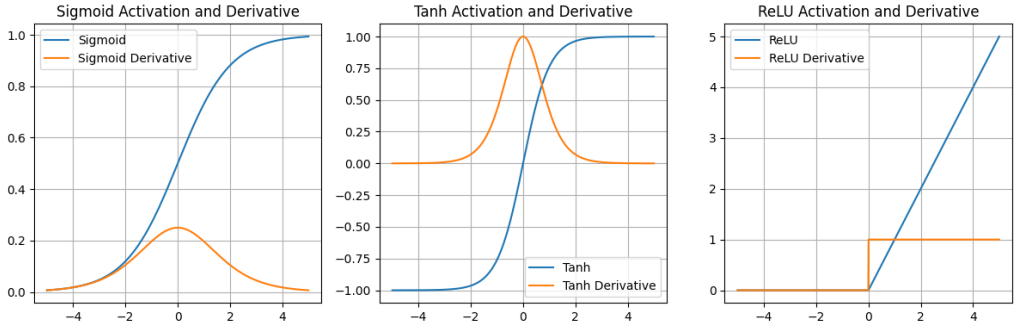

Figure 2.3: The sigmoid, tanh, and ReLU activation functions together with their derivatives.

The sigmoid and tanh activation functions have similar properties due to their similar shape, with the difference in output domain where the sigmoid ranges from 0 to 1 and the zero-centered tanh ranges from -1 to 1. The two activation functions are described in Equation 2.8 and 2.9.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.9}$$

The output range of the sigmoid activation function makes it popular in applications where the output should be in the same $[0, 1]$ range. For example, in classification applications where the output will be interpreted as a probability vector, or when the output should be strictly positive.

The ReLU activation function was originally proposed by Nair and Hinton [22] and is an efficient and simpler way of introducing the non-linearity in the neural network. The mathematical formulation of the activation function is

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.10}$$

Due to its effectiveness, ReLU has become very popular in deeper neural networks to speed up computation. Nevertheless, it becomes inapplicable in applications that require high-order derivatives of the neural network output to be computed, as the second-order derivative of the linear ReLU function is zero [23]. The Gaussian Error Linear Units (GELU) is a smoothed version of the ReLU activation function that has a non-zero second-order derivative, solving the aforementioned problem [24]. The GELU activation function is visualized in Figure 2.4 and the mathematical formulation is

$$\mathrm{GELU}(x) = x \cdot \frac{1}{2}[1 + \mathrm{erf}(x/\sqrt{2})] \tag{2.11}$$

For computational efficiency, GELU is approximated as

$$\mathrm{GELU}(x) \approx 0.5(1 + \tanh\big[\sqrt{2/\pi}(x + 0.044715x^3)]\big) \tag{2.12}$$

Hendrycks and Gimpel [24] also argue that the curvature for the positive domain of the GELU (compared to ReLU which is constant) could enable networks to approximate more complex functions.

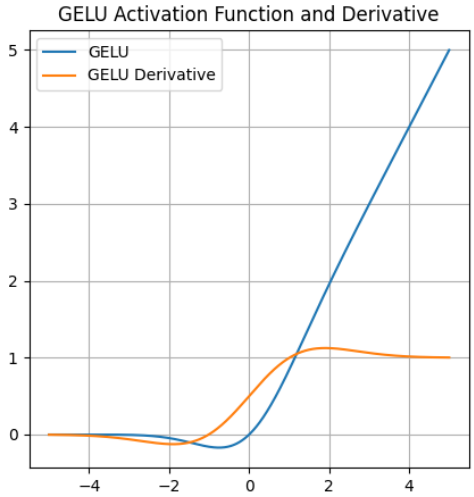

Figure 2.4: The GELU activation function and its derivative.

2.2.3 Problems with Training Neural Networks

Training a neural network is a form of implicit optimization, aiming to minimize the loss on the training data with the overarching objective of obtaining a model that generalizes well to previously unseen data. However, the assumption that a small training loss implies a correspondingly small generalization error is often overly optimistic. This phenomenon is illustrated in Figure 2.5 and is referred to as

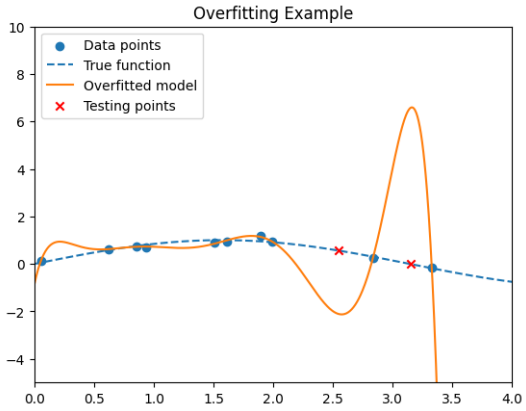

Figure 2.5: An example where overfitting occurs and the model fails to generalize.

overfitting. Here, the model performs exceptionally well on the training data but fails to generalize to new unseen data.

How well a model performs on unseen data is measured as generalization error and reducing this error is a key objective in machine learning. One solution for reducing the generalization error is to train the model on a more diverse and representative dataset to make the resulting model less prone to fit towards noisy observations and rather capture the true distribution of the dataset. Another strategy is to make use of regularization methods when training the network. A common regularization method is L^2-regularization, also known as weight decay, and operates by adding a penalty term to the loss based on the magnitude of the model weights.

$$\mathcal{L}_{L^2} = \mathcal{L}_\theta + \lambda ||\mathbf{w}||_2^2 \tag{2.13}$$

In this way, the minimization of the loss during training also penalizes model complexity where the hyperparameter λ controls the strength of the penalty. In L^2-regularization the model complexity is quantified by the sum of the squared weights, where a greater weight contributes more to the model complexity.

Another obstacle related to deep neural networks is the phenomenon known as vanishing gradients. This is related to the backpropagation step of the training cycle of neural networks where each weight's effect on the loss is calculated by computing the partial derivative of the loss with respect to each weight. The issue arises when these partial derivatives become extremely small when propagating from the final layer to the earlier ones. A small derivative results in a small weight update, and thereby slow training.

The occurrence of vanishing gradients is more common in deep network architectures using bounded smooth activation functions such as tanh and sigmoid. The sigmoid, tanh, and ReLU activation functions, together with their derivative, are shown in Figure 2.3. The sigmoid and tanh derivative is bounded on $(0, 0.25]$ and $(0, 1]$ respectively, and both derivatives go towards zero for values far away from zero. Therefore, propagating back n layers in the derivative computation by the chain rule requires n multiplications of this potentially small value, which means that the derivative decays exponentially with n. Not being able to update the weights in earlier layers appropriately may prevent the network from learning im-

portant features.

2.3 Physics-Informed Neural Networks

Traditional deep learning models rely on the availability of enormous datasets and recent advancements in computational techniques and resources. In this big data regime, deep learning models usually start from scratch and exploit the information from large and diverse datasets to find the complex non-linear mapping between input and output pairs. However, the acquisition cost of said data is prohibitive, and the quality of the data strongly affects the model's robustness and convergence. The higher the quality of the data, the better the conditions for model convergence. To overcome this, models that efficiently capture the complexity of the problem to be solved, while operating on small datasets are desired. This reduction in dataset size inevitably leads to an information loss that needs to be counteracted with other techniques.

In 2018, Raissi *et al.* [25] presented the physics-informed neural networks (PINN) to overcome the aforementioned challenge of limited data [25]. With the use of prior knowledge from governing laws of physics, the authors presented an extension of the traditional multilayer perceptron that accurately modeled complex solutions, even with only a few training examples available. By describing the physical laws using partial differential equations and evaluating the model predictions against these equations it acts as a regularization agent that quickly steers the model toward acceptable solutions.

This thesis covers two types of problems with the use of PINNs The first one is data-driven solutions of PDEs, and will here on be referred to as the forward problem. In this problem setup, the parameters for the PDE are known and constant, and data is used to compute solutions to the PDE in general form. The second one is the data-driven discovery of PDEs, which will be referred to as the inverse problem. Here, the parameters are no longer assumed to be known and constant. Instead, these are inferred using a PINN from potentially noisy observations.

In this section, the building blocks of a physics-informed neural network are described in detail. Known challenges related to PINNs will also be covered, along with methods on how they may be overcome.

2.3.1 PINN Problem Formulation

Raissi *et al.* [25] originally proposed a formulation of PINNs as follows. Assume a partial differential equation in its original form

$$u_t + \mathcal{N}[u;\lambda] = 0,\ x \in \Omega,\ t \in [0,T] \tag{2.14}$$

where $u(t,x)$ denotes the latent solution, $\mathcal{N}[\cdot;\lambda]$ is a nonlinear differential operator parameterized by λ, and Ω is a subset of $\mathbb{R}^D$.

Now define $f(t,x)$ to be given by the left hand side of Equation 2.14,

$$f := u_t + \mathcal{N}[u;\lambda] \tag{2.15}$$

For the forward problem, the parameters λ are known and constant, whereas for the inverse problem, λ are learnable parameters for the PINN. The unknown solution $u(t, x)$ is represented by a deep neural network $u_\theta(t, x)$ parameterized by θ. Here, the parameters θ refer to all the trainable weights and biases in the neural network. Combining the approximation of $u(t, x)$ from the neural network with Equation 2.15 results in another network $f(t, x)$ with shared model parameters which can be learned by minimizing the composed loss function,

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_f \tag{2.16}$$

where

$$\mathcal{L}_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \tag{2.17}$$

and

$$\mathcal{L}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \tag{2.18}$$

The notation $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denotes the training data for the initial and boundary constraints of the PDE, and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ the collocation points for the residual loss calculated from $f(t, x)$. Minimizing $\mathcal{L}_u$ ensures that the PINN satisfies the problem-specific boundary and initial constraints while minimizing $\mathcal{L}_f$ preserves the sought-after structure of the solution imposed by the physical equation. The residual loss term therefore acts as a strong regularization agent preventing the model from overfitting. Successfully minimizing this combined loss function results in model parameters for the network which satisfies both requirements for a good solution. In the context of inverse problems, an extra loss term is incorporated to compute the observational loss derived from the observation data. Consequently, the total loss is expressed as follows:

$$\mathcal{L}_{obs} = \frac{1}{N_{obs}} \sum_{i=1}^{N_{obs}} |u(t_{obs}^i, x_{obs}^i) - u^i|^2 \tag{2.19}$$

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_f + \mathcal{L}_{obs} \tag{2.20}$$

Incorporating the observational loss term allows the model to obtain the needed information from measurements and discover the unknown parameters of the PDE. The unknown parameters of the PDE are introduced in the PINN as external learnable parameters. As these physical parameters are included in the calculation of the residual loss, they may be optimized in the same way as the network's weights and biases with the use of backpropagation such that the residual loss is minimized.

2.4 Performance-enhancing techniques for PINNs

Despite the promising results exhibited by PINNs across various domains, recent research has revealed that they suffer from a number of issues that restrict their

accuracy and robustness [23]. This section covers an overview of the challenges related to PINNs and recent work regarding techniques to tackle them.

2.4.1 Non-dimensionalization

In deep learning, an important data pre-processing step is to normalize data to harmonize range, magnitude, and units across the inputs [26]—failure to do so risks increasing training time and causing convergence problems for three main reasons. Firstly, optimization algorithms based on gradient descent may encounter vanishing or exploding gradients if the initialization of the network weights is of the wrong magnitude. As most common initialization schemes (e.g., Glorot [26]) assume that the input data is close to zero mean and unit variance, rescaling is typically required. Secondly, if input variables differ in scale, one variable can dominate predictions over others and prevent learning of important feature relationships. Thirdly, rescaling allows the optimizer to take similarly sized steps when adjusting weights for all variables, stabilizing and expediting learning [23].

For PINNs in particular, this coincides with the established practice in physics of non-dimensionalizing PDEs by transforming them to a dimensionless system by removing scales and units. Importantly, however, the scale of the PDE residuals cannot be contained a priori using this model and may require reactive rescaling to avoid it taking a vastly different scale than the other loss terms [23].

2.4.2 Random Fourier Features

In general, deep neural networks tend to be biased towards learning less complex functions during training. In particular, they show a bias towards learning lower frequency functions, a pathology referred to as spectral bias [27]. As PINNs are based on neural networks, this weakness applies to them as well and constitutes a fundamental weakness complicating the accurate approximation of high-frequency functions and fine details [28].

Tancik *et al.* [29] showed that the pathology of spectral bias can be reduced by mapping the input vector $\mathbf{x}$ into a higher dimensional space using a random Fourier feature embedding before passing it into the standard MLP. The Fourier feature embedding is defined as

$$\gamma(\mathbf{x}) = \begin{bmatrix} \cos(\mathbf{B}\mathbf{x}) \\ \sin(\mathbf{B}\mathbf{x}) \end{bmatrix} \tag{2.21}$$

where each entry in $\mathbf{B} \in \mathbb{R}^{m \times d}$ is sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma > 0$ being a user-specific hyperparameter, $2m$ being the desired embedding dimension, and d being the dimensionality of the input vector $\mathbf{x}$. In practice, Wang *et al.* [23] recommend a σ between 1 and 10. For PINNs in particular, random Fourier features have been demonstrated to improve predictions of sharp gradients [28], a commonly occurring phenomenon in discharge physics.

2.4.3 Random Weight Factorization

Random weight factorization (RWF) is a technique proposed by Wang *et al.* [30] that has been empirically shown to mitigate spectral bias, enable MLPs to be less sensitive to unfavorable initialization, and find better optima. The core idea is to reparametrize the weight matrices for each layer l as $\mathbf{W}_l = \text{diag}(\mathbf{s}_l) \cdot \mathbf{V}_l$, and instead apply gradient descent on the new parameters $\mathbf{s}_l$ and $\mathbf{V}_l$ during training. Thus, it is also an interesting technique for PINNs. A brief motivation for RWF will now follow.

Consider the weight matrix $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ for the l-th layer of an L-layer MLP. Each element $w_{i,j} \in \mathbf{W}_l$ corresponds to the weight between the i-th neuron in layer $l-1$ and the j-th neuron in layer l. Consequently, the k-th row vector $\mathbf{w}_{k,l}$ in $\mathbf{W}_l$ contains the weights associated with a specific neuron in the l-th layer (namely the k-th neuron).

In RWF, the weights for each individual neuron in the network are parameterized as

$$\mathbf{w}_{k,l} = s_{k,l} \cdot \mathbf{v}_{k,l}, \quad k = 1, 2, \ldots, d_l, \quad l = 1, 2, \ldots, L+1 \tag{2.22}$$

where $\mathbf{w}_{k,l} \in \mathbb{R}^{d_{l-1}}$, $s_{k,l} \in \mathbb{R}$, and $\mathbf{v}_{k,l} \in \mathbb{R}^{d_{l-1}}$. More concisely, for a given layer l, this is equivalent to

$$\mathbf{W}_l = \text{diag}(\mathbf{s}_l) \cdot \mathbf{V}_l, \quad l = 1, 2, \ldots, L+1 \tag{2.23}$$

with $\mathbf{s}_l \in \mathbb{R}^{d_l}$.

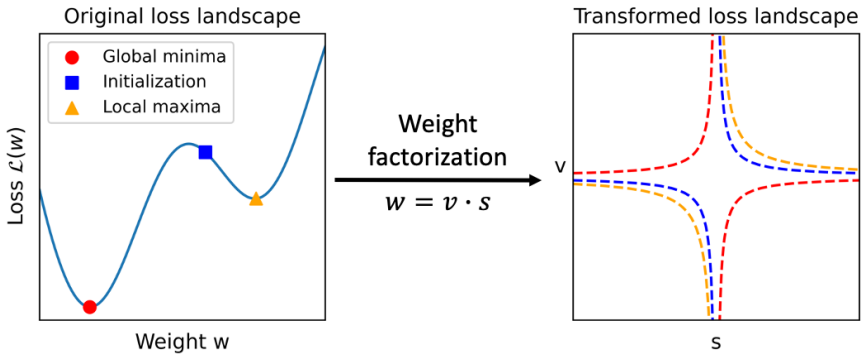

Figure 2.6: Random weight factorization transforms the loss landscapes, reducing the distance between the initialization and the global minima. The left pane shows the hyperbolas corresponding to the global minima, local minima, and initialization in the sv-plane.

For a geometric motivation for RWF, consider a one-parameter loss function $\mathcal{L}(w)$ for a scalar w. Then the proposed factorization is reduced to $w = s \cdot v$ for a pair of scalars (s, v) [30]. For any non-zero parameter w, there exist infinitely many (s, v) pairs for which $s \cdot v = w$. These pairs form hyperbolas in the sv-plane on which the loss function $\mathcal{L}$ is constant. The effect is illustrated in 2.6, where the left pane shows the original loss function and the right pane shows the hyperbolas in the sv-plane corresponding to the initialization and global and local minima. In particular, notice that the distance between the initialization and the global minima

becomes arbitrarily small when in the factorized sv-plane for larger values of s. In fact, it can be proven that the distance between any two points is arbitrarily small for any general loss function in arbitrary parameter dimensions (further details and a complete proof can be found in [30]):

Theorem 1 *Suppose that $\mathcal{L}(\theta)$ is the associated loss function of a neural network. For a given θ, let U_θ be the set of all possible weight factorizations*

$$U_\theta = \left\{ (\mathbf{s}_l, \mathbf{V}_l)_{l=1}^{L+1} : \text{diag}(\mathbf{s}_l) \cdot \mathbf{V}_l = \mathbf{W}_l, \quad l = 1, \ldots, L+1 \right\}$$

Then for any θ, θ',

$$\text{dist}(U_\theta, U_{\theta'}) := \min_{\mathbf{x} \in U_\theta, \mathbf{y} \in U_{\theta'}} ||\mathbf{x} - \mathbf{y}|| = 0$$

Theorem 1 thus provides an intuition for the empirical result of RWF being able to help MLPs to escape from unfavorable initializations and to find better optima.

Alternatively, RWF can be interpreted as effectively assigning a self-adaptive, trainable learning rate $s_{k,l}$ to each individual neuron $a_{l,k}$ in the network. This is realized by observing that when using RWF, the standard gradient descent with learning rate η

$$\mathbf{w}_{n+1}^{(k,l)} = \mathbf{w}_n^{(k,l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_n^{(k,l)}} \tag{2.24}$$

instead becomes

$$\mathbf{w}_{n+1}^{(k,l)} = \mathbf{w}_n^{(k,l)} - \eta \left(||[s_n^{(k,l)}]^2 + v_n^{(k,l)}||^2 \right) \frac{\partial \mathcal{L}}{\partial \mathbf{w}_n^{(k,l)}} \tag{2.25}$$

To practically implement RWF, the parameters of the MLP are first initialized as usual, using, for instance, the Glorot scheme [26]. Then, for every weight matrix $\mathbf{W}_l$, $\mathbf{s}_l$ is sampled from a multivariate normal distribution $\mathcal{N}(\mu, \sigma \mathbf{I})$. Finally, $\mathbf{V}_l$ is initialized by factorizing each weight matrix as $\mathbf{W}_l = \text{diag}(\exp(\mathbf{s}_l)) \cdot \mathbf{V}_l$. Here, the exponential parameterization expression of $\mathbf{s}_l$ is used to help avoid zeros and allow the values to span a wide range of magnitudes, as per [31].

2.4.4 Violating Temporal Causality

For time-dependent PDEs, it becomes important to consider the flow of information across the space and time of the system considered [32]. The temporal precedence principle dictates that the system's state at time t is determined by its state at time $t - \Delta t$ and, by extension, the initial conditions. The propagation of the information contained within the initial conditions through the system over time, in turn, corresponds well to the casual nature of physical systems that determines their evolution.

Such principles of basic causality are leveraged in classical numerical solvers of PDEs by discretizing the problem in time and solving for time t before attempting to solve for time $t + \Delta t$, ensuring the propagation of information [32]. However, in the classic PINN formulation by [25] such considerations are missing as all PDE residuals are minimized simultaneously. Such an approach is equivalent to attempting to learn

the solution at a time t before having an accurate solution for any previous time steps $\hat{t} < t$ which violates temporal causality and, by failing to incorporate the information in the initial conditions, is unlikely to converge at a correct solution [32]. Furthermore, Wang *et al.* [32] show that the traditional PINN formulation is actually biased towards minimizing the temporal residual for later times, aggravating this issue. They argue that this tendency to attempt to solve later timesteps before earlier is a fundamental pathology preventing PINNs from solving more complex problems.

To avoid violating temporal causality, [32] propose an alternative formulation of the PDE residual loss function $\mathcal{L}_r(\theta)$ disregards loss contributions from subsequent times if the previous times have not reached low losses. This is achieved by splitting the temporal domain into M parts with associated residual loss functions $\mathcal{L}_i(\theta)$ for $i = 1, 2, \ldots, M$. Thus, the PDE residual for the whole domain can be expressed as

$$\mathcal{L}_r(\theta) = \frac{1}{M} \sum_{i=1}^{M} w_i \mathcal{L}_r^i(\theta) \tag{2.26}$$

for some weights w_i. These weights w_i should be inversely proportional to cumulative residual loss from previous time segments in order to ensure that L_i is minimized before attempting to minimize L_j for all $i < j$, preserving temporal causality. To that end, [32] propose using

$$w_i = \exp\left(-\epsilon \sum_{k=1}^{i-1} \mathcal{L}_r^k(\theta)\right), \text{ for } i = 2, 3, \ldots, M \tag{2.27}$$

for a user-specific hyperparameter ϵ determining the degree to which causality is enforced (larger ϵ means previous residuals must be smaller before latter residuals are considered), thereby impacting both learning and convergence speed. The resulting residual loss for the whole domain then becomes

$$\mathcal{L}_r(\theta) = \frac{1}{M} \sum_{i=1}^{M} \exp\left(-\epsilon \sum_{k=1}^{i-1} \mathcal{L}_r^k(\theta)\right) \mathcal{L}_r^i(\theta) \tag{2.28}$$

This modified PDE residual has been shown to improve accuracy for several time-dependent problems [32].

2.4.5 Loss Balancing

Recall that the loss minimized when training PINNs is $\mathcal{L}(\theta) = \mathcal{L}_{ic}(\theta) + \mathcal{L}_{bc}(\theta) + \mathcal{L}_r(\theta) + \mathcal{L}_{obs}(\theta)$. As mentioned in Section 2.4.1, a challenge in training is that the loss terms may be of widely different scales which can cause some terms to dominate others and complicate the search for the optimal solution [23]. To remedy this, appropriate weights can be assigned to each loss term so that $\mathcal{L}(\theta) = w_{ic}\mathcal{L}_{ic}(\theta) + w_{bc}\mathcal{L}_{bc}(\theta) + w_r\mathcal{L}_r(\theta) + w_{obs}\mathcal{L}_{obs}(\theta)$. However, manually choosing appropriate weights is difficult as scales may change during training and are highly problem-specific.

Wang *et al.* [23] address this by introducing an algorithm for automatic loss balancing that ensures that the norm of the gradients of the different losses remain comparable. Thereby, any bias towards minimizing a specific loss term during training

is eliminated. At every N:th training iteration, new weights $\hat{w} = (\hat{w}_{ic}, \hat{w}_{bc}, \hat{w}_r, \hat{w}_{obs})$ are computed as

$$\begin{aligned}
\hat{w}_{ic} &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta) + \nabla_\theta \mathcal{L}_{bc} + \nabla_\theta \mathcal{L}_r + \nabla_\theta \mathcal{L}_{obs}||}{||\nabla_\theta \mathcal{L}_{ic}||} \\
\hat{w}_{bc} &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta) + \nabla_\theta \mathcal{L}_{bc} + \nabla_\theta \mathcal{L}_r + \nabla_\theta \mathcal{L}_{obs}||}{||\nabla_\theta \mathcal{L}_{bc}||} \\
\hat{w}_r &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta) + \nabla_\theta \mathcal{L}_{bc} + \nabla_\theta \mathcal{L}_r + \nabla_\theta \mathcal{L}_{obs}||}{||\nabla_\theta \mathcal{L}_r||} \\
\hat{w}_{obs} &= \frac{||\nabla_\theta \mathcal{L}_{ic}(\theta) + \nabla_\theta \mathcal{L}_{bc} + \nabla_\theta \mathcal{L}_r + \nabla_\theta \mathcal{L}_{obs}||}{||\nabla_\theta \mathcal{L}_{obs}||}.
\end{aligned} \tag{2.29}$$

The global weights $w = (w_{ic}, w_{bc}, w_r, w_{obs})$ are then updated using a moving average as

$$w_{new} = \alpha w_{old} + (1 - \alpha)\hat{w}_{new} \tag{2.30}$$

where the parameter α is a smoothing factor between 0 and 1 that balances stability with responsiveness to new data.

2.4.6 Hard Boundary Constraints

As mentioned in Section 2.4.5, it is problematic to minimize a composed loss function where the terms are of different scales and sizes. In the work of Lu *et al.* [33], the authors reduced the number of loss terms by strictly imposing hard Dirichlet boundary conditions through modification of the model architecture. In comparison with using soft constraints and satisfying them by minimization of a loss term, their approach satisfies the Dirichlet boundary conditions exactly and simultaneously reduces the computational cost. The authors implemented hard boundary constraints by the following,

Consider a Dirichlet boundary condition for the solution u given by,

$$u(x) = g(x), \ x \in \Gamma_D \tag{2.31}$$

where $\Gamma_D \subset \partial\Omega$ is a subset of the boundary. To force the approximate solution $\hat{u}(x;\theta)$ to satisfy this boundary condition, we construct the solution as

$$\hat{u}(x;\theta) = g(x) + \ell(x)\mathcal{N}(x;\theta_u) \tag{2.32}$$

where $\mathcal{N}(x;\theta_u)$ is the network output, and $\ell(x)$ is a function satisfying the following two conditions:

$$\begin{cases} \ell(x) = 0, \ x \in \Gamma_D, \\ \ell(x) > 0, \ x \in \Omega - \Gamma_D. \end{cases} \tag{2.33}$$

In a simpler geometry for Γ_D, it is possible to choose $\ell(x)$ analytically [34], [35]. For instance, when Γ_D is the boundary of the interval $\Omega = [a, b]$ then

$$\ell(x) = (x - a)(x - b)$$

A schematic of the hard boundary implementation is shown in Figure 2.7.

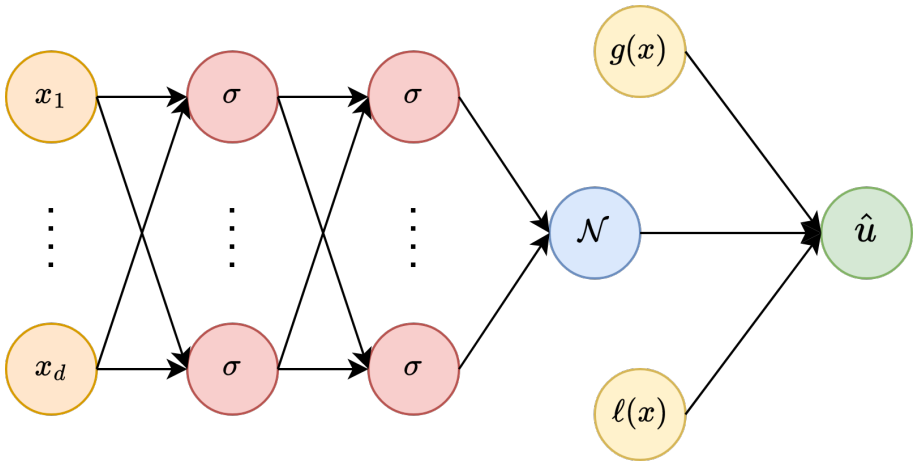

Figure 2.7: Schematic of a PINN using the hard constraint implementation proposed by Lu *et al.* [33].

3
Methods

In pursuit of understanding the potential of PINNs in addressing inverse problems, the focus centered on the exploration of their capabilities in solving Poisson's equation and the drift-diffusion equation within the realm of discharge physics. Given the broad and exploratory nature of the thesis project's objectives, it was decided to apply an iterative workflow reminiscent of the *Design Science Research* (DSR) methodology [36]. The DSR methodology focuses on obtaining insights about a problem through iterative development and evaluation of an artifact that solves the problem at hand. The artifacts in this project were the PINN models, and by iteratively evaluating the added features of our models we could determine which methods were applicable to our problem, while also reflecting on why some were unsuccessful.

To further concertize the performance and applicability of our model we continued on the three cases outlined in the previous thesis work by Konradsson [18], with the extension of solving the inverse problem for each case. The three cases, described in detail in Section 3.1, comprised Laplace's equation, the Drift-diffusion equation, and the Mono polar charge transport coupled with Poisson's equation. The third case is the combination of the two aforementioned. While studying both equations in isolation were objectives of this thesis themselves, it also gave rise to insights into how to solve the more complex problem of coupling the two equations.

3.1 Cases

To explore the applicability of PINNs in the domain of discharge physics, PINNs were implemented to solve both the forward and inverse problems associated with various cases of interest. The cases are based on those proposed in [18], providing useful benchmarks.

3.1.1 Case 1: Laplace's and Poisson's Equation

In the first case, consider a one-dimensional coaxial geometry similar to that described in [3], [20] and illustrated in Figure 3.1. This domain consists of an inner electrode with potential U_0 at the surface and radius r_0, a distance $r_1 - r_0$ of insulating air, and an outer metal cage with surface potential U_1 and radius r_1. Notably, this design with an inner electrode surrounded by an insulation media (in this case, air) and an outer metallic cage is also similar to the design of a coaxial transmission line [37].

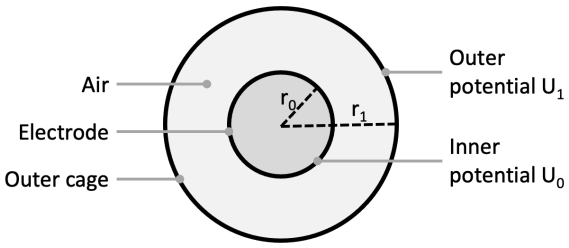

Figure 3.1: Case 1 consists of an inner conductor (radius r_0) surrounded by air insulation and an outer cage (radius r_1).

The task in case 1 is to, given the potentials and radii, determine the electric scalar potential field $U(r)$ between the inner electrode and outer cage $r \in (r_0, r_1)$. The potential in the vicinity of surfaces with boundary conditions can be determined using a PDE named Poisson's Equation

$$\nabla^2 U = -\frac{\rho}{\epsilon} \tag{3.1}$$

where ρ is the volume charge density and ϵ is the permittivity [37]. In this case, and indeed in many other applications, the charge generating the electric field lies outside of the domain, namely in the inner electrode. Therefore, while we have an electric field in the insulating region (r_0, r_1), it can be considered a source-free region free of charge (i.e., $\rho = 0$). Poisson's equation is then reduced to the PDE *Laplace's Equation*

$$\nabla^2 U = 0 \tag{3.2}$$

In cylindrical coordinates, this becomes

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial U}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 U}{\partial \phi^2}+\frac{\partial^2 U}{\partial z^2}=0 \tag{3.3}$$

for radial distance r, azimuth angle ϕ, and height z. Furthermore, radial symmetry gives $\partial U/\partial \phi = 0$ and for the interesting case of a long transmission line with a small radius r_1, every cross-section far from the edges has $\partial U/\partial z \approx 0$ [37]. This reduces Laplace's Equation to

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial U}{\partial r}\right)=\frac{\partial^2 U}{\partial r^2}+\frac{1}{r}\frac{\partial U}{\partial r}=0 \tag{3.4}$$

and when adding the boundary conditions the complete formulation of case 1 becomes

$$\begin{cases} \dfrac{\partial^2 U}{\partial r^2}+\dfrac{1}{r}\dfrac{\partial U}{\partial r}=0 \\ U(r=r_0)=U_0 \\ U(r=r_1)=U_1 \\ r\in[r_0,r_1] \end{cases}. \tag{3.5}$$

In particular, for the simple case of a grounded outer cage with $U(r_1) = 0V$ and $U(r_0) = 1V$, this problem has the analytical solution [18]

$$U(r) = \frac{\ln(r_1) - \ln(r)}{\ln(r_1) - \ln(r_0)}, \quad r \in [r_0, r_1] \tag{3.6}$$

and, consequently, the electrical field $E = -\nabla U$ in the region becomes

$$E(r) = \frac{1}{r} \frac{1}{\ln(r_1) - \ln(r_0)}, \quad r \in [r_0, r_1]. \tag{3.7}$$

Equations 3.6 and 3.7 also highlight why case 1 can be challenging to solve with conventional numerical methods, as the potential and electrical field explode for the smallest r as $r_0 \to 0$, resulting in very steep curves.

3.1.2 Inverse Problems for Case 1

Two problems where additional physical parameters, in addition to the potential U, were unknown were considered to test the data-driven discovery of PDEs associated with Poisson's Equation:

- The first problem was to infer a non-zero and unknown volume space charge ρ.
- The second problem was to infer unknown radii r_1 and r_2, given that their difference $\Delta r = r_2 - r_1 > 0$ was known. This was also done with a non-zero, but known, space charge ρ.

It is straightforward to show that the analytical solution to Poisson's equation (i.e., for the two inverse cases with non-zero space charge density ρ) in the case of boundary conditions $U(r = r_0) = 1$ and $U(r = r_1) = 0$ is given by $U(r)$ where

$$\begin{cases} U(r) = C_1 + C_2 \ln(r) - \dfrac{\rho r^2}{4\epsilon} \\ C_1 = \dfrac{\rho r_1^2}{4\epsilon} - C_2 \ln(r_1) \\ C_2 = \dfrac{1}{\ln\left(\frac{r_0}{r_1}\right)} - \dfrac{\rho(r_1^2 - r_0^2)}{4\epsilon \ln\left(\frac{r_0}{r_1}\right)} \end{cases}. \tag{3.8}$$

Thereby, this analytical solution can be used to generate observations of the potential U for the two inverse cases.

3.1.3 Case 1.5: Poisson's Equation in Cartesian Coordinates with Fixed Charge Distribution

In forward Case 1, the air between the inner and outer electrodes was assumed to be a charge-free zone. However, a corona discharge model should account for the strong electric field close to the inner electrode ionizing the surrounding air to cause discharges, as later described in Case 2. Crucially, when the charge carriers move through the domain, they influence the resulting electric field E. This results in a coupling of Poisson's equation and the drift-diffusion equation, where the potential,

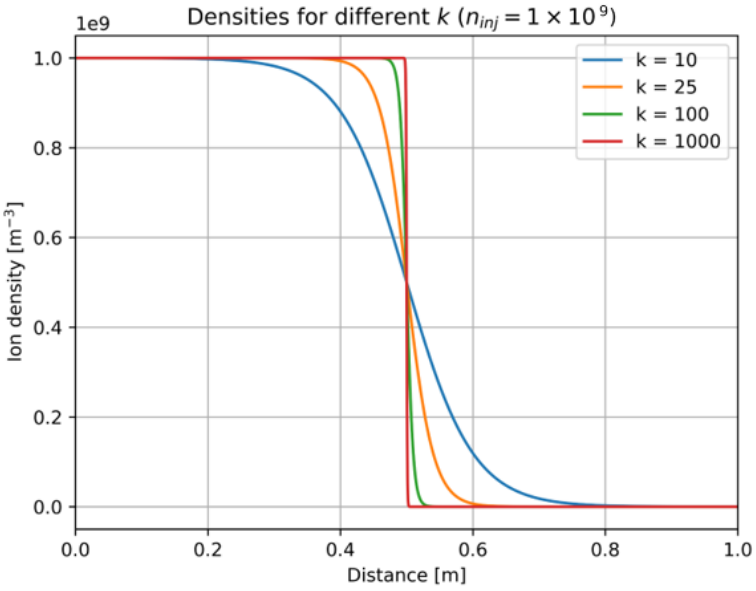

Figure 3.2: Stationary ion density for different steepness parameters k.

and, thus, the electric field, influences the density of ions, and, thus, the space charge and vice versa [3]. This complex model is covered in Case 3.

Case 1.5 is designed to serve as an intermediate step between Case 1, with no space charge, and Case 3, with a time-dependent space charge, by introducing a fixed charge distribution. Thus, it is similar to solving Case 3 for a fixed timestep. Furthermore, charge density with variable steepness was used as it was hypothesized that the steepness of the charge density profile in Case 2 complicated the learning process. The chosen charge density profile was therefore

$$n(x;k) = n_{inj}\left(1 - \frac{1}{1+\exp(-2k(x-0.5))}\right) \tag{3.9}$$

where k is the steepness parameter, illustrated in Figure 3.2. Accounting for a non-zero charge density, Poisson's equation becomes

$$\begin{cases} \dfrac{\partial^2 U}{\partial x^2} + \dfrac{qn(x;k)}{\epsilon} = 0 \\ U(x = x_0) = U_0 \\ U(x = x_1) = U_1 \\ x \in [0,1] \end{cases}, \tag{3.10}$$

with a switch from cylindrical to Cartesian coordinates.

3.1.4 Inverse Case 1.5

The inverse version of Case 1.5 considers the case when volume density $n(x)$ is unknown. The task is to determine the volume density $n(x)$ with the help of observations of the resulting potential $U(x)$.

3.1.5 Case 2: Drift-Diffusion Equation

An essential element within physical modeling of discharges is describing how charge carriers behave in an electrical field [4]. In particular, understanding their creation (due to ionization), destruction (due to recombination), and their movement in the electrical field during their lifespan is of interest. It can be described through the density function $n(t, \mathbf{x})$ for each charge species [3].

However, achieving a well-converged solution from a model containing both electrons, positive ions, and negative ions is tedious for a large-scale geometry, not least due to the short mean free path of electrons requiring very high granularity [4]. Furthermore, accounting for the continuous recombination and ionization throughout the spacial and temporal domain also complicates the model significantly. To account for this, Schiessling *et al.* [4] propose a model that only accounts for ionic charge carriers (i.e., disregards electrons) and restricts the active corona region to the surface of the inner electrode in coaxial systems like the one described in Case 1 (Section 3.1.1), enabling it to be modeled as a boundary injecting ions into the air at the surface. This model is motivated by the electrical field in Case 1 being proportional to $1/r$ and thus strongly non-uniform with a peak at the inner electrode to where the active corona region is restricted. This also supports the exclusion of electrons from the model, by modeling them as instantly forming negative ions at the boundary of the corona region.

Case 2 leverages this model, predicting the charge density for only positive ions and treating the active corona region to be only at the boundary while switching to Cartesian coordinates to simply further, as illustrated in Figure 3.3. The behavior of charge species in an electrical field can be described using the drift-diffusion equation

$$\frac{\partial n}{\partial t} + \nabla \cdot \Gamma = S \tag{3.11}$$

describing how the volume density n for a charge carrier changes over time depending on the flux Γ and the source term S [3]. The Drift-Diffusion equation describes two fundamental processes that may change the concentration of a charge species:

1. Existing charge species may move, either due to the influence of an external electric field (i.e., drift) or due to random thermal motion (i.e., diffusion). The motion (i.e., flux) caused by drift and diffusion constitutes the flux Γ and can be described further as Γ = drift flux + diffusive flux = $\mu E n - D\nabla n$, where μ is the ion electrical mobility, E is the external electric field, and D is the diffusion coefficient. The diffusion coefficient D, in turn, can be broken down into $D = \mu k_b T q^{-1}$, where k_b is the Boltzmann constant, T is the temperature, and q is the elementary charge [3].
2. Charge species can be created (through ionization) or destroyed (through recombination). The rate of these changes is contained in the source term S.

As such, the drift-diffusion equation for positive ions can be expressed as

$$\frac{\partial n}{\partial t} + \nabla \cdot \left(\mu E n - \frac{\mu k_b T}{q} \nabla n \right) = S \tag{3.12}$$

where n is the volume density of positive ions and S is the source term for positive ions. Finally, to further simplify case 2 an approximation of $S \approx 0$ was made as

Label	Meaning	Unit	Value
n_0	Initial ion density	m^{-3}	1×10^{-1}
μ	Ion mobility	$\mathrm{m}^2\mathrm{V}^{-1}\mathrm{s}^{-1}$	2×10^{-4}
E	External electric field	Vm^{-1}	1×10^{6}
k_b	The Boltzmann constant	JK^{-1}	1.38×10^{-23}
T	Temperature	K	293
q	The elementary charge	C	1.602×10^{-19}

Table 3.1: Parameter values for drift-diffusion equation.

in [18]. Together with the boundary conditions $n(t, x = 0) = n_{inj}$ as the constant injection simulating the active corona region near the high-potential plate and $n(t = 0, x \neq 0) = n_0$ as the even density across the domain before injection, the complete problem becomes

$$\begin{cases} \dfrac{\partial n}{\partial t} + \mu E \dfrac{\partial n}{\partial x} - \dfrac{\mu k_b T}{q} \dfrac{\partial n}{\partial x} = 0 \\ n(t, x = 0) = n_{inj}, \quad t \geq 0 \\ n(t = 0, x) = n_0, \quad x \in (0, 1] \\ x \in [0, 1]. \end{cases} \tag{3.13}$$

For this simple domain, the drift-diffusion equation reduces to describing the propagation of the injected ions between the plates with velocity μE as there is no ionization or recombination and a homogeneous electric field. With $E >> k_b T q^{-1}$, the analytical solution can be approximated as

$$n(t, x) \approx \begin{cases} n_{inj}, & \text{if } x \leq \mu E t \\ n_0, & \text{else.} \end{cases} \tag{3.14}$$

For the physical parameter involved, the same values as in [18] and [3] were used and are detailed in Table 3.1.

3.1.6 Inverse Case 2: Inferring Unknown Ion Mobility

In the inverse problem formulation for the drift-diffusion equation, the ion mobility μ was treated as an unknown parameter to be inferred from observation data. In the experiments, the true ion mobility was set to 2×10^{-4} $\mathrm{m}^2\mathrm{V}^{-1}\mathrm{s}^{-1}$, and the parameter was randomly initialized within the range of 2×10^{-5} to 2×10^{-3} ($\mathrm{m}^2\mathrm{V}^{-1}\mathrm{s}^{-1}$).

3.1.7 Case 3: Mono Polar Charge Transport Coupled with Poisson's Equation

As described in Section 3.1.3, when accounting for drifting charged species throughout the domain, the potential $U(t, x)$ depends both on the external electric field and space charge from the density of ions at x after time t. Likewise, the drift of the positive ions through the domain depends on the electric field $E = -\nabla U$. Therefore,

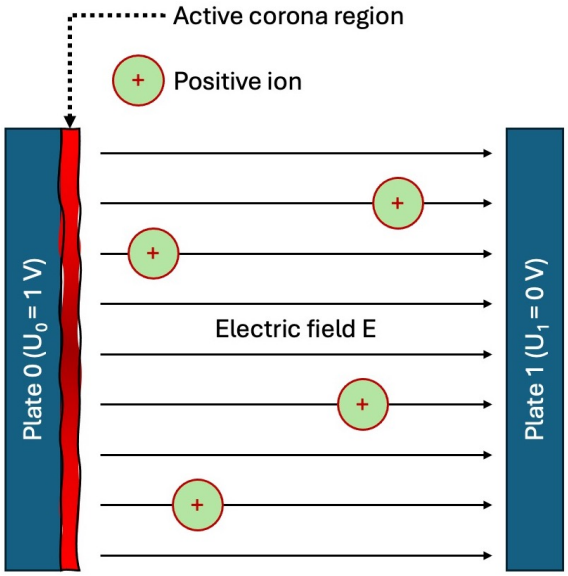

Figure 3.3: Illustration of drift-diffusion equation setup.

a complete model of positive ions behavior after injection at the high-potential plate and the resulting electric field requires the coupling of Poisson's equation and the drift-diffusion equation. While Case 1.5 corresponds to solving this coupled equation system for a fixed time t, the task in Case 3 is to solve it for the entire temporal domain. The resulting coupled equation system becomes

$$\begin{cases} \dfrac{\partial^2 U(t,x)}{\partial x^2} + \dfrac{qn(t,x)}{\epsilon} = 0 \\ \dfrac{\partial n(t,x)}{\partial t} - \mu \nabla U(t,x) \dfrac{\partial n(t,x)}{\partial x} - \dfrac{\mu k_b T}{q} \dfrac{\partial n(t,x)}{\partial x} = 0 \\ U(t,0) = U_0 \\ U(t,1) = U_1 \\ n(t,0) = n_{inj} \\ n(0,x) = n_0 \\ t \in [0, t_{end}] \\ x \in [0,1] \end{cases}, \tag{3.15}$$

by coupling Equations 3.10 and 3.13 and using the identity $E = -\nabla U$. Note that Equation 3.15 contains two PDEs that are dependent on one another, constituting the coupling.

3.1.8 Physical Parameters for the Cases

The physical parameters for various cases are detailed in Table 3.2. These specific values were employed in the final series of experiments, and the results derived from these experiments will be elaborated upon in the subsequent chapter. For the inverse

cases, the superscript *true* is used to denote the target value for the parameter that is being inferred. Furthermore, the initial values for these parameters were chosen uniformly at randomly chosen from the ranges specified in the table.

Cases	**Parameters**
Forward Case 1:	$r_0 = 10^{-4}$, $r_1 = 0.5$, $u_0 = 1$, $u_1 = 0$, $\rho = 0$
Inverse Charge Density:	$r_0 = 10^{-3}$, $r_1 = 0.5$, $u_0 = 1$, $u_1 = 0$, $\rho^{true} = 10^{-3}$, $\hat{\rho} \in [0, 1] \times 10^{-4}$
Inverse Geometry:	$\Delta r = 0.5$, $r_0^{true} = 10^{-4}$, $r_1^{true} = 0.5001$, $u_0 = 1$, $u_1 = 0$, $\rho = 5 \times 10^{-10}$, $\hat{r}_0 \in [5 \times 10^{-2}, 5 \times 10^{-1}]$
Forward Case 2:	$n_{inj} = 10^9$, $n_0 = 0.1$, $\mu = 2 \times 10^{-4}$ $E_{ext} = 10^6$
Inverse Case 2:	$n_{inj} = 10^9$, $n_0 = 0.1$, $\mu_{true} = 2 \times 10^{-4}$, $\hat{\mu} \in [2 \times 10^{-5}, 2 \times 10^{-3}]$, $E_{ext} = 10^6$
Forward Case 1.5:	$u_0 = 1 \times 10^6$, $u_1 = 0$, $k = 100$, $n_{max} = 5 \times 10^{13}$, $n_{min} = 0$
Inverse Case 1.5:	$u_0 = 1 \times 10^6$, $u_1 = 0$, $k \in \{100, 500\}$, $n_{max} = 5 \times 10^{13}$, $n_{min} = 0$
Forward Case 3:	$u_0 = 10^6$, $u_1 = 0$, $n_0 = 0.1$, $n_{inj} = 5 \times 10^{13}$

Table 3.2: Used parameter values for each of the described cases.

3.2 Model Implementation

The models were implemented in JAX [38] using Flax [39] by adapting the implementation by Wang *et al.* [23] to the domain of discharge physics. JAX is a high-performance, numerical computing library developed by Google and intended for large-scale machine learning [38]. JAX combines advanced automatic differentiation, an XLA (Accelerated Linear Algebra) powered implementation of NumPy [40] capable of running on multiple GPUs and TPUs cores, and just-in-time (JIT) compilation to achieve high computing capabilities. Flax [39], in turn, is a neural network library for JAX developed by Google Research and designed to integrate well with JAX and make full use of its performance capabilities.

The decision to modify the code by Wang *et al.* [23] to the relevant use cases, as opposed to making a completely novel implementation or using a high-level framework, such as DeepXDE [41], was made to achieve the flexibility of a low-level code base while minimizing development and testing time required to create it.

The implementation of the PINNs also used random batch sampling for the collocation points. Thereby, for each training iteration a batch consisting of between 512 and 8,192 coordinates where randomly sampled and used to compute the PDE residuals. This procedure is advocated by Wang *et al.* [23] and the resulting stochasticity introduces a greater regularization effect than using a fixed set of collocation points, as in [18]. Simultaneously, it is a simpler procedure than the more complex sample strategies such as the residual-based adaptive refinement (RAR) method [41] that actively chooses collocation points based on the domain the PDE residual is

the largest, although the impact of such methods could be interesting to study in future work.

In the upcoming subsections, we will outline specific model implementation methods and to which cases they were applied.

3.2.1 PDE Re-scaling: Parameters of Order One

To enhance numerical stability during training, the parameters of the PDEs are often rescaled to have an order of magnitude close to one. When parameters in the PDE have vastly different scales, it can complicate training, with those of larger scale having a greater impact on the loss function. This imbalance may lead to slower convergence or vanishing gradient issues, particularly when activation functions saturate such as for sigmoid and tanh. In Case 1, this involved multiplying the PDE by the radius r, yielding the transformed equation:

$$r\frac{\partial^2 U}{\partial r^2} + \frac{\partial U}{\partial r} = 0 \tag{3.16}$$

For Case 2, we instead divide the equation by the drift-speed $w = \mu E$, resulting in:

$$\frac{1}{w}\frac{\partial n}{\partial t} + \frac{\partial n}{\partial x} - \frac{1}{w}\frac{\mu k_b T}{q}\frac{\partial n}{\partial x} = 0 \tag{3.17}$$

In Case 3, these two modifications were combined for the respective PDE residual loss terms.

Furthermore, for the inverse task of discovering an unknown space density ρ for Case 1.5, both the observation and the PDE residual loss were scaled down by a factor of 10^{-9} before the mean squared error of each was calculated. This was in response to these losses generating NaN values when they grew too large and crashing training. In principle, such rescaling should not affect the result as the task of minimizing the loss function $\mathcal{L}(\theta)$ is independent of constant scaling. Nevertheless, a tendency for deteriorating performance when the loss scaling factor was too small although this pathology was not investigated further.

3.2.2 Hard Boundary Constraints

Hard boundary constraints, covered in Section 2.4.6, were implemented for the prediction of the potential U across Cases 1 and 3. Consequently, the implementation of the neural network outputs $\mathcal{N}(x;\theta)$ where modified to

$$\frac{r_1 - r}{r_1 - r_0}U_0 + \frac{r - r_0}{r_1 - r}\mathcal{N}(r;\theta) \tag{3.18}$$

for Case 1, to accommodate the boundary conditions $U(r_0) = U_0$ and $U(r_1) = U_1$, and

$$\frac{x_1 - x}{x_1 - x_0}U_0 + \frac{x - x_0}{x_1 - x}\mathcal{N}(x,t;\theta) \tag{3.19}$$

for Case 3 to reflect the shift to Cartesian coordinates.

For the charge density profile $n(t,x)$, however, no simple Dirichlet boundary condition satisfying $n(t, x = 0) = n_{inj}$ and $n(t = 0, x \neq 0) = n_0$ exists and thus no hard boundaries were implemented for the prediction of the charge density.

3.2.3 Inferring a Constant Parameter

To reiterate, when solving the inverse problems the PDE is parameterized by some unknown parameter λ. In inverse case 1 and inverse case 2, the standard network architecture had to be extended with the functionality of inferring this constant parameter from the observational data. To achieve this, the functionality was introduced in JAX by treating λ as an external learnable parameter within the network. This design enables the network to adjust and optimize the value of λ during the training process, similar to how it learns other model parameters.

3.2.4 Inferring a Function

In inverse Case 1.5, the objective shifts from inferring a constant to capturing a function that characterizes the space charge within the geometry. To accomplish this, an additional output neuron was introduced in the network dedicated to predicting the space charge. By configuring the loss calculation such that this output is treated as the predicted space charge, the true space charge can be inferred by training the network as usual.

3.2.5 Sequential Training

For both the forward and inverse problem of case 3, a PINN model consisting of two disconnected sub-networks was developed inspired by Niaki *et al.* [42]. Together with a sequential training algorithm, these two sub-networks are designed to approximate one equation each. In our case, that is one network approximating the potential $u(t, x)$, and the other network the charge density $n(t, x)$ described in Equation 3.15. This design choice aims to mitigate the challenges associated with training a single model to solve coupled equations concurrently, introducing stability concerns. In order to still solve the coupled case of the equations, the sequential training works in the following way. Start by training one of the sub-networks and treat the output from the other network as constant. After k iterations, switch and train the other network and treat the previously trained network's output as constant. This means that the parameters of the in-active network are not optimized when the active network is training.

Another advantage of having separate networks for each PDE is the ability to tailor each network for its specific equation. Specific hyperparameters and performing-enhancing methods found suitable for each of the cases can therefore be used to solve the coupled case of the two PDEs.

3.3 Model Selection and Evaluation

A neural network consists of a range of hyperparameters to be defined by the developer. The optimal set of hyperparameters is problem-specific and needs to be empirically evaluated to find the best-performing set. To ensure the suitability of found hyperparameters, a Grid search was conducted for each model. In a grid search, a predefined set of hyperparameter values is systematically explored. Each combi-

nation of hyperparameters is then used to train and evaluate a model to identify the best-performing set. However, it's important to note that the time complexity of grid searches grows exponentially with the number of hyperparameters, resulting in scalability issues. In our iterative workflow, we leveraged gained experience to identify impactful hyperparameters, allowing us to streamline the search process. By discarding hyperparameters that did not significantly impact performance, we optimized our approach given the time constraints of a thesis project and the substantial number of models to evaluate. The chosen investigated hyperparameters are shown in Table 3.3.

Hyperparameter	**Values**
Number of layers	4, 6
Number of neurons per layer	256, 512
Activation function	Tanh, GELU, Sigmoid

Table 3.3: Grid Search Hyperparameters

The relative L^2 error,

$$\text{Rel. } L^2 \text{ error} = \frac{||\text{prediction} - \text{reference value}||_2}{||\text{reference value}||_2} \tag{3.20}$$

was chosen to quantify the performance of the different models. Here, $|| \cdot ||_2$ is the L^2 norm. However, this requires that a reference solution is available. In the cases where an analytical reference solution was not available, the performance was instead assessed by comparison with a FEM solution generated in COMSOL. The relative L^2 error is suitable for this project since it encourages smooth predictions by penalizing larger errors more heavily. Additionally, relative L^2 error is particularly useful when dealing with datasets where the magnitudes of the true values vary widely. In such cases, using absolute L^2 error tends to emphasize errors in larger values. Relative L^2 error normalizes the error by the magnitudes of the true values, providing a more balanced view.

Having obtained optimal hyperparameters through grid search, an ablation study was conducted to evaluate each of the performance-enhancing methods proposed by Wang *et al.* [23]. This involved systematically excluding one method at a time, allowing us to isolate the effect of each method and assess its individual impact. The analyzed methods were *Gradient normalization* to deal with loss balancing, *Random weight factorization* and *Random Fourier Features* to mitigate spectral bias, and finally the *Causality loss re-formulation* to respect temporal bias in the cases including a time-domain.

In all inverse cases involving the inference of information from observational data, we introduced noise to the input data to assess the robustness of the models. Uncorrelated Gaussian noise was systematically added to the observational data at increasing levels: 1%, 10%, 25%, and 50%. Having a model that is robust towards noisy data is important for real-world applications where accurate measurements

are difficult to obtain. Being able to extract valuable information from noisy data therefore enhances the model's usability.

Lastly, each model was evaluated ten times with different random seeds. This was done to eliminate the dependency of a specific random initialization of model weights and biases. Additionally, this also generates a range for the model performance to aid in the discussion for model confidence.

4
Results and Discussion

This chapter will present and discuss the results from the cases described in Section 3.1. The results are organized by equation, starting with Laplace's and Poisson's equation in Case 1, followed by the drift-diffusion equation in Case 2, and finally the Coupled Case 3. Sweeps and ablation studies for forward cases were evaluated through relative L^2 error to provide a common dimensionless evaluation metric, using an analytical solution or COMSOL-generated solution as the reference. Similarly, ablation studies for the inverse cases are evaluated using relative L^2 error on the predictions of the unknown physical entity compared to the true value or distribution.

4.1 Case 1: Laplace's and Poisson's Equation

This section presents the hyperparameter sweeps, ablation studies, and sensitivity analyses for the forward Case 1 and the two inverse Case 1 problems.

4.1.1 Forward: Solving Laplace's Equation

The results for the hyperparameter sweep for the forward problem of Laplace's equation with an inner radius of 0.1 mm are illustrated in Figure 4.1. There are three notable results here. Firstly, only one set of hyperparameters, namely six layers of width 256 with GELU, converges to an accurate solution. Secondly, models with the GELU activation function consistently outperformed those with tanh. Thirdly, deeper networks outperformed more shallow ones.

These results indicate that the problem has a high sensitivity to hyperparameter choices. Therefore, a rigorous hyperparameter search would be advisable before choosing model architecture and configurations. Furthermore, the superiority of GELU over tanh is notable as [23] recommend the latter as the default activation, further highlighting the importance of hyperparameter tuning and the absence of silver bullet configurations.

The results of the ablation study on performance-enhancing techniques on the best-performing model from the hyperparameter sweep are shown in Figure 4.2. Notice that the modified loss function described in Section 2.4.4 to enforce time causality and gradient normalization are in-applicable as the problem is time-independent and only has one loss term when using hard boundary constraints. The results show that:

1. Meaningful learning is dependent on the use of random Fourier features.

2. Random weight factorization slows the learning process but significantly improves the end results when combined with random Fourier features.

Given that random Fourier features have been shown to improve the ability of PINNs to predict steep gradients it makes sense that the technique becomes important in Case 1 which is characterized by the potential having a sharp gradient close to the inner electrode. Moreover, the increased training time following the inclusion of RWF is also reasonable given that it effectively doubles the number of trainable weights in the network. The larger set of parameters, in turn, requires more training time to train much like larger networks typically require more training than smaller ones.

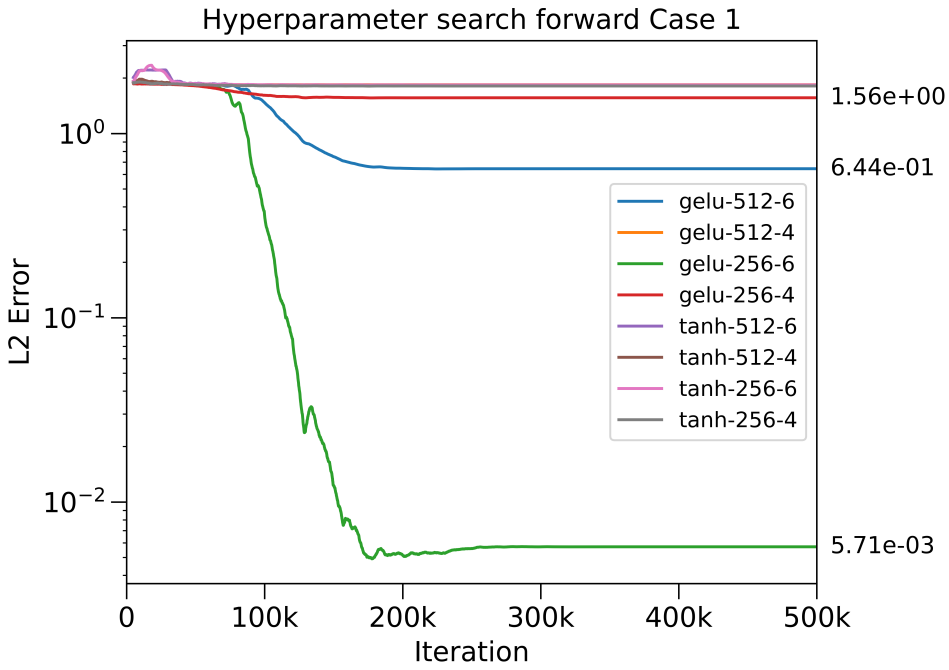

Figure 4.1: Hyperparameter search for forward Laplace's Equation. Labels are formatted as *activation function - width - depth*. The Error estimates are based on single simulations with random initializaiton.

Sample predictions and errors for forward Case 1 using both the optimal setup from the hyperparameter sweep and ablation study shown in Figure 4.3. The PINN model makes extremely accurate predictions of both the potential and the resulting electric field for a radius of 0.1 mm compared to the 5 mm used in Konradsson [18]. Thereby, a different network architecture and inclusion of Fourier features enables the modeling of a ~ 30 times stronger electrical field. Finally, running the model for ten different seeds using a batch size of 8,192 showed stable learning with an average relative L^2 error of 0.00547 and a standard deviation of 0.00555. The results of the ten runs are shown in Figure A.1 in Appendix A.

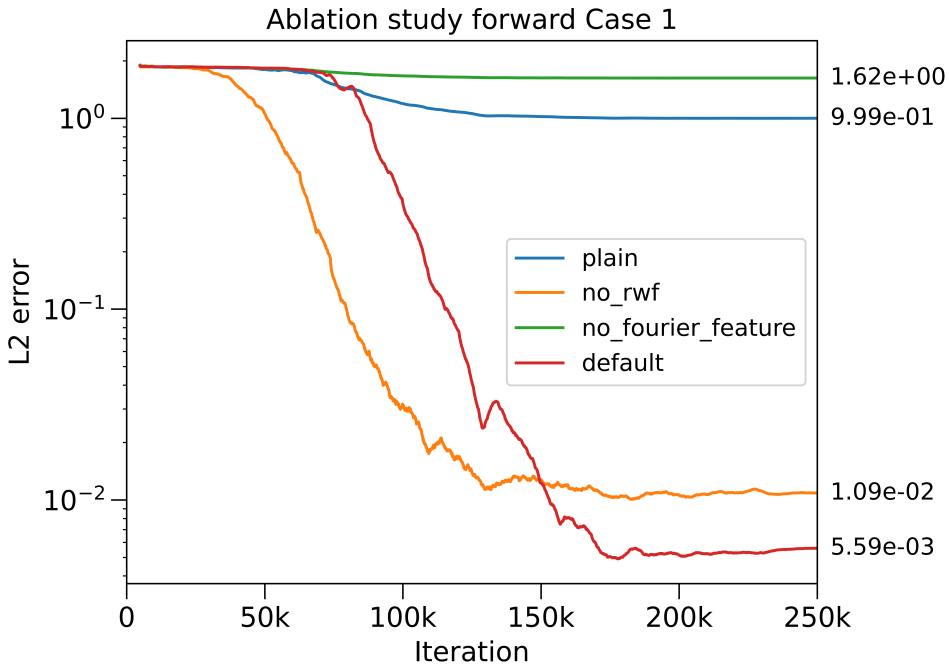

Figure 4.2: Ablation study for forward Laplace's Equation. The error estimates are based on single simulations with random initialization.

4.1.2 Inverse Charge Density

The relative parameter L^2 errors for the inverse Case 1 ablation study with unknown space charge ρ, 100 noise-free observations, and the optimal network architecture from the forward version of the problem are shown in 4.4. Notice that gradient normalization is included in the ablation study, as opposed to in the forward case, as there is now an additional observation loss term. These results indicate that gradient normalization is crucial for learning while the inclusion of RWF and random Fourier features only decreases model accuracy. When using gradient normalization, however, the model predicts the unknown parameter ρ with extreme accuracy as errors are limited to about 0.1%.

Interestingly, this indicates that the importance of the impact of the proposed performance-enhancing techniques differs significantly between the forward and inverse versions of Case 1. The importance of gradient normalization likely follows from the difference in magnitude between the residual and observation loss terms the former ends at a magnitude of 10^{-3} while the latter ends at a magnitude of 10^{-7}. Thus, without gradient normalization, minimizing the sum of the two loss terms becomes equivalent to minimizing the 10,000 times greater residual loss and the observations are ignored. With gradient normalization, however, the loss terms become comparably large and the minimization of their sum is dependent on minimizing both of them.

Another interesting result is the negative impact of RWF and random Fourier features. A possible explanation is that the inclusion of noise-free observations is sufficient to fit the model prediction to the steep gradient and that embedding it in a higher dimension only serves to complicate the learning. Regardless, it highlights

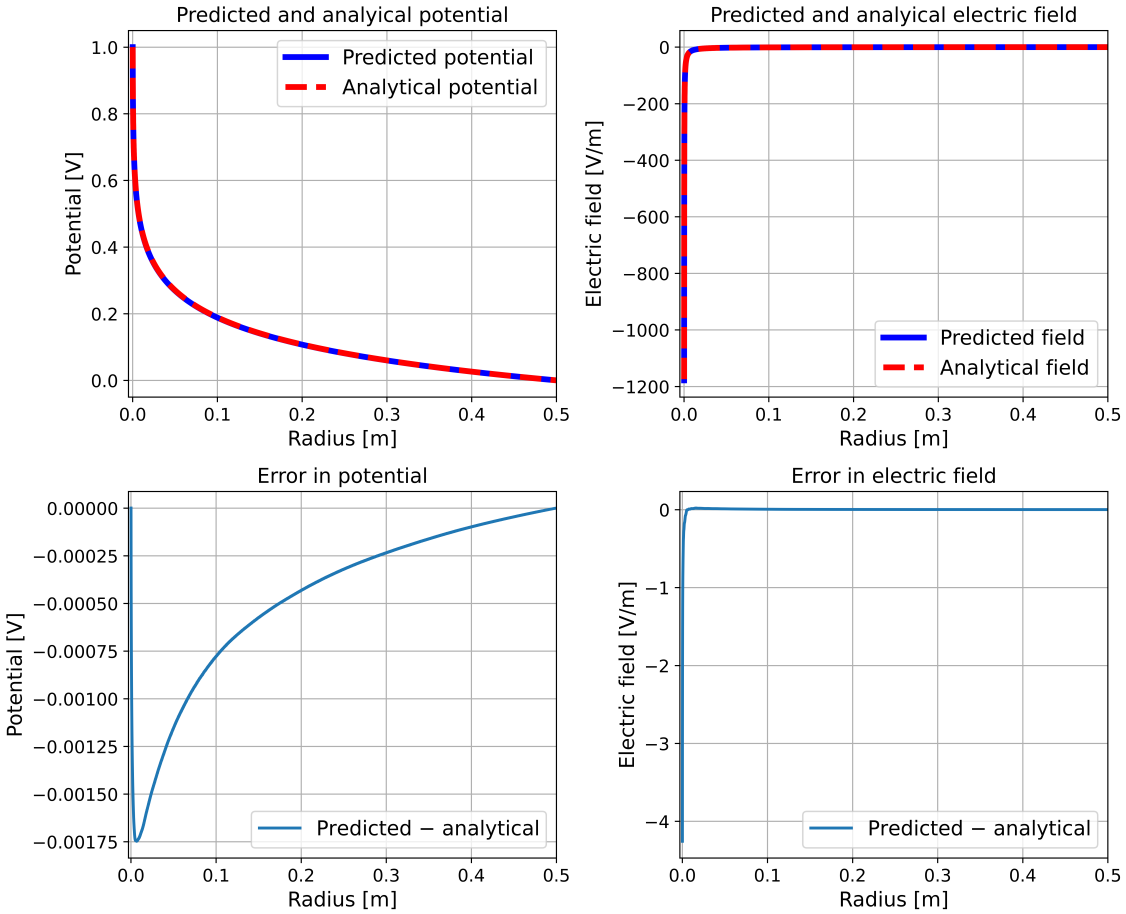

Figure 4.3: Predicted potential and electric field compared to the analytical solution for the forward case of Laplace's equation.

the lack of a silver bullet architecture for PINNs and suggests that the challenges of the forward and inverse problems differ significantly.

The sensitivity analysis for the impact of the number of observations and their noise levels when using the optimal setup, i.e., gradient normalization but neither RWF nor Fourier features, is summarized in Table 4.1. As expected, model accuracy tends to decrease with fewer observations and higher noise levels. While the required accuracy is application-dependent, the results overall suggest a high resistance to fewer and noisier observations. Additionally, the results display clear stochasticity with higher accuracy for 50% noise than 10% in the case of 1,000 observations. This stochasticity, in turn, suggests that a greater batch size than the 4098 used might be preferable to increase the stability of the training process and that multiple runs for noise level and number of observations should ideally have been done to provide a confidence interval for each accuracy.

4.1.3 Inverse Geometry

The relative parameter L^2 error for the inverse problem of Case 1 with an unknown inner radius r_0 and 100 noise-free observations is illustrated in Figure 4.5. Echoing

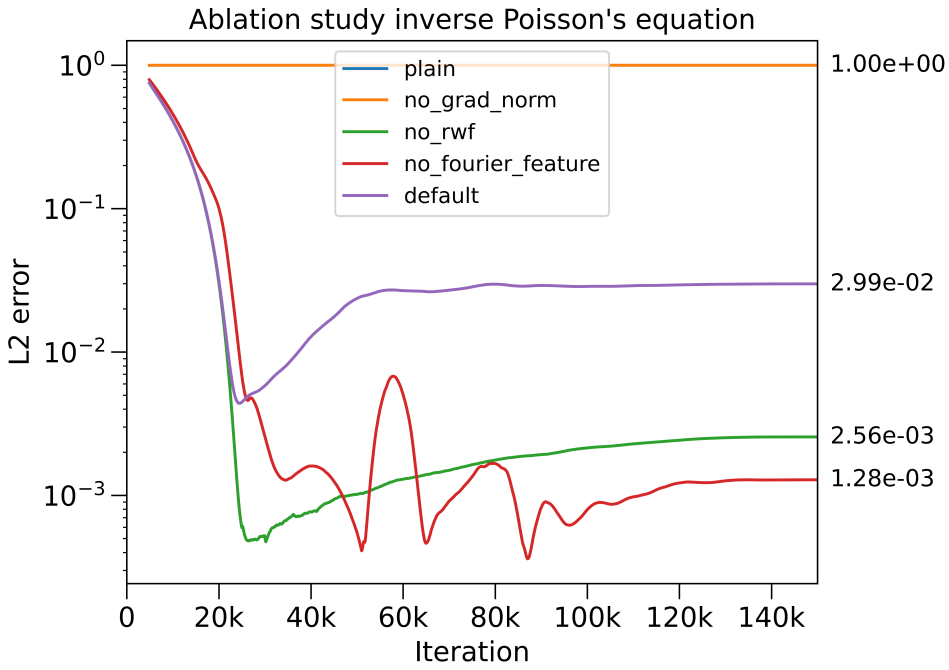

Figure 4.4: Ablation study for inverse Poisson's equation. The error estimates are based on single simulations with random initialization.

Relative parameter L^2 error			
Size \ Noise	10%	25%	50%
10	0.116	0.064	0.113
100	0.062	0.146	0.308
1,000	0.020	0.053	0.098

Table 4.1: Relative parameter L^2 error for a unknown charge density ρ for different number of observations and noise levels. The error estimates are based on single simulations with random initialization.

the above results with an unknown space charge ρ, the results once again suggest that gradient normalization is crucial for learning. The inclusion of random Fourier features and RWF, on the other side, had a negligible impact on model accuracy. Overall, the accuracy is significantly lower than that of the unknown space charge. Nevertheless, given the unfavorable initialization of r_0 at 11 cm compared to the true value of 0.1 mm, all methods were able to discover meaningful information about the true geometry of the system. The sufficiency of the accuracy of these discoveries is, once again, determined by the intended application.

The failure of the model to converge to the accurate radius of 0.1 mm could be explained by that a difference in potential between an inner radius of 0.1 mm and one of 0.2 mm radius is only seen in extreme proximity to the electrode, making the problem inherently hard when using a randomly sampled batch. This hypothesis is supported by the results from the sensitivity analysis summarized in Table 4.2, which was generated using a batch size of 8,192, instead of 4,096 for increased stability,

where a higher precision was achieved even with 10% noise.

Table 4.2 also indicates a greater noise sensitivity compared to the discovery of the space charge ρ. Model accuracy deteriorated significantly already at a 10% noise level regardless of the number of observations, indicating the increased difficulty of this task.

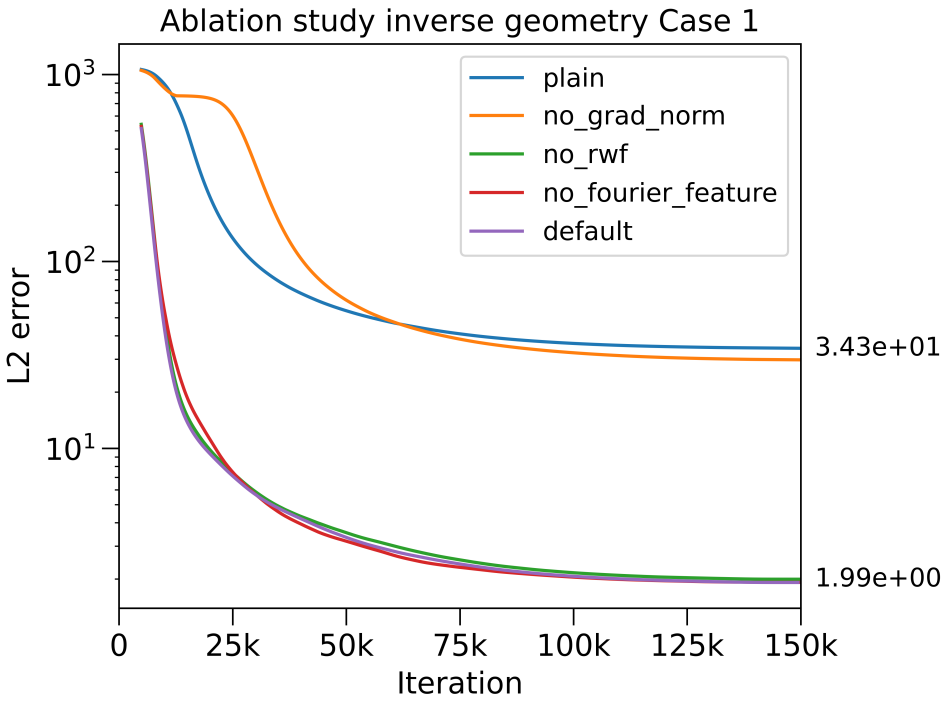

Figure 4.5: Ablation study for Case 1 with unknown geometry. The error estimates are based on single simulations with random initialization. *Default* is using the combination gradient normalization, RWF, and random Fourier features while *plain* is using neither.

Relative parameter L^2 error			
Size \ Noise	1%	5%	10%
10	4.23	8.02	20.15
100	1.72	5.73	21.52
1,000	1.07	4.64	19.85

Table 4.2: Relative L^2 parameter error for different observation dataset sizes and noise levels. The error estimates are based on single simulations with random initialization.

4.2 Case 1.5: Poisson's Equation

This section presents the outcomes of Case 1.5, as outlined in Section 3.1.3. To reiterate the problem at hand, Case 1.5 involves solving Poisson's equation with

a non-constant space charge. The forward problem entails predicting the potential distribution, while the inverse problem focuses on inferring the space-charge distribution based on observations derived from the potential distribution. These observations were generated from the PINN solving the forward problem. To verify the correctness of the PINN's solution the relative L^2 error was calculated, referencing a provided solution from COMSOL. This error metric quantifies the alignment between the two solutions, with a lower error indicating a better fit. However, a more accurate solution found by the PINN model, compared to the COMSOL solution, could yield a greater L^2 error. To complement the relative L^2 error, the PINNs PDE residual loss was also examined to compare the different setups' performance, where a lower loss indicates a better obtained solution for the PINN.

4.2.1 Forward: Solving Poisson's equation with non-constant space charge

The resulting PDE residual loss and relative L^2 error for each evaluated hyperparameter setup for the forward case are listed in Table 4.3.

Case 1.5 Grid Search Results				
#neurons	#Layers	Activation	Rel. L^2 error	PDE Res. Loss
256	4	Tanh	3.3×10^{-5}	1.93×10^{3}
256	4	Gelu	3.3×10^{-5}	7.99×10^{2}
256	4	Sigmoid	3.3×10^{-5}	1.41×10^{3}
256	6	Tanh	3.3×10^{-5}	6.44×10^{3}
256	6	Gelu	2.8×10^{-5}	$\mathbf{6.56 \times 10^{2}}$
256	6	Sigmoid	3.3×10^{-5}	1.68×10^{3}
512	4	Tanh	3.3×10^{-5}	1.11×10^{4}
512	4	Gelu	$\mathbf{1.7 \times 10^{-5}}$	1.99×10^{4}
512	4	Sigmoid	3.3×10^{-5}	1.51×10^{3}
512	6	Tanh	3.3×10^{-5}	1.10×10^{3}
512	6	Gelu	2.8×10^{-5}	9.41×10^{2}
512	6	Sigmoid	3.3×10^{-5}	1.26×10^{3}

Table 4.3: Resulting L^2 error and residual loss for each Case 1.5 hyperparameter setup. The lowest obtained L^2 error and PDE residual loss are marked with bold font. The error estimates are based on single simulations.

By studying the L^2 error it is observed that consistently low L^2 errors were obtained by all models. This means that all setups manage to approximate the solution fairly well. The lowest error, and thereby best replication of the COMSOL solution, was obtained with four hidden layers, 512 neurons per layer, and GELU as the activation function.

In contrast, there is a considerable variation in the residual loss. Given that the residual loss quantifies the degree to which the identified solution satisfies the PDE, and considering that all configurations yielded an acceptable L^2 error, the optimal

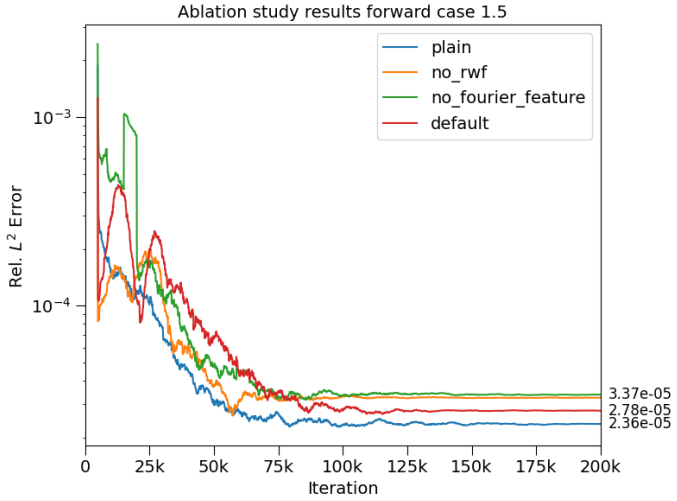

Figure 4.6: Convergence of Rel. L^2 error for the evaluated performance-enhancing methods. The error estimates are based on single simulations with random initialization. *Default* is using both random Fourier features and RWF while *plain* is using neither.

architecture was selected based on the lowest residual loss. This was the setup with six hidden layers, 256 neurons per layer, and GELU as the activation function.

After determining the optimal network architecture, the impact of performance-enhancing methods was assessed. The resulting relative L^2 error and residual loss are visualized in Figure 4.6 and 4.7, respectively. As in the hyperparameter search, all models demonstrated equivalent performance in terms of L^2 error, while the residual loss better showed the different methods impact on the obtained solution.

The result from the ablation study demonstrates that both random weight factorization and Fourier feature embedding indeed improve the model's performance and allow for further minimization of the residual loss. Examining the graph reveals that Fourier features have the most substantial impact on improving PINN, as evidenced by a more significant increase in the residual loss when this method is excluded. This agrees with the impact Fourier features had in the results of the forward Case 1 as detailed in 4.1.1. Note that loss balancing with gradient normalization was not evaluated in this case since the residual loss was the only loss term in the otherwise compound loss function. This, as a result of using hard boundary conditions for the potential. Using the optimal hyperparameters from the hyperparameter search, together with both random weight factorization and Fourier feature embedding, the resulting predictions of the Poisson's equation with non-constant space-charge is visualized in Figure 4.8.

The figure shows how the PINN model captures the complexity of the PDE and performs equally well as the COMSOL solution for both the potential and the resulting electrical field (recall that $E = -\nabla U$). This highlights a key strength of PINNs in contrast to traditional neural networks. In PINNs, the inclusion of the partial differential equation and residual loss functions serves as a regularization mechanism for the network. This regularization prevents the model from identifying overly intricate, high-frequency solutions to the potential, which could otherwise

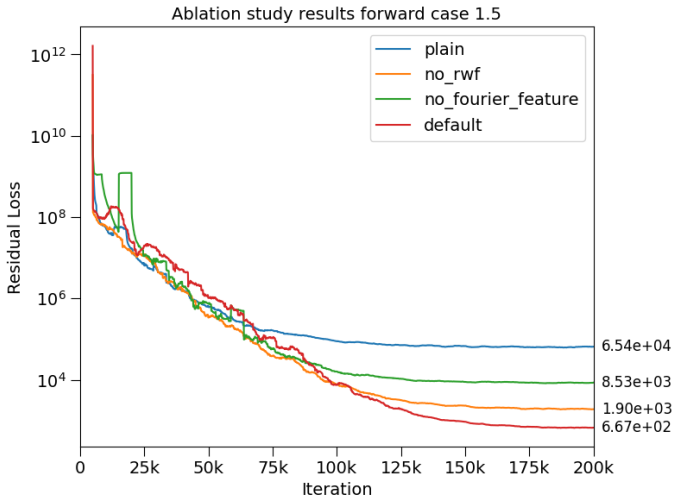

Figure 4.7: Convergence of PDE Residual loss for the evaluated performance enhancing methods. The error estimates are based on single simulations with random initialization. *Default* is using both random Fourier features and RWF while *plain* is using neither.

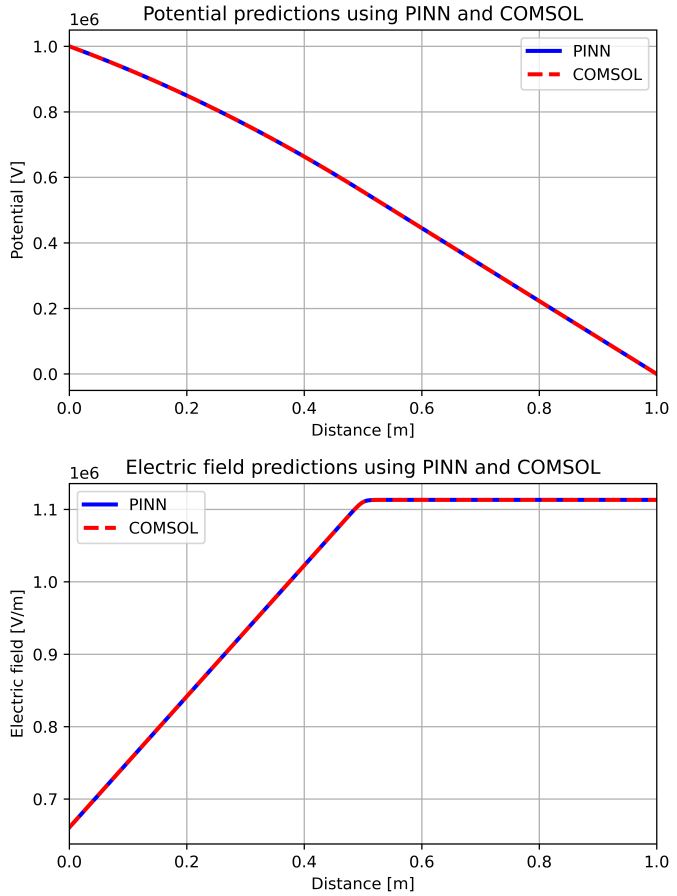

Figure 4.8: Comparison of PINN solution with COMSOL solution.

lead to inaccurate predictions of the electrical field. Finally, the evaluation runs with ten different initialization seeds showed the PINNs robustness towards random initialization with a mean L^2 error of 3.1×10^{-5} with a standard deviation of 7.1×10^{-6}.

4.2.2 Inverse: Inferring non-constant space-charge

The objective in the inverse case was to infer a non-constant space charge based on measurement from the true potential distribution. The dataset comprised 1,000 points generated from the PINN model developed in 4.2.1.

In this particular experiment, where the goal is to infer a function $n(x)$, the model output has been modified into a two-dimensional vector, as detailed in Section 3.2.4, enabling the prediction of both the potential and the underlying space-charge. Evaluation of the accuracy of the inferred charge profile was performed using the relative L^2 error against the true density profile, which was the smoothed step-function outlined in Equation 3.9, with a steepness parameter $k = 100$.

The ablation study, conducted with the hyperparameter configuration obtained from the forward case, produced the results listed in Table 4.4.

Inverse Case 1.5 Ablation Study Results					
RWF	Gradient Norm.	Fourier Features	Rel. L^2 error	Obs. Loss	PDE Res. Loss
✓	✓	✓	2.1×10^{-2}	7.7×10^{-1}	1.8×10^{4}
✗	✓	✓	3.0×10^{-2}	2.3	9.0×10^{2}
✓	✗	✓	1.5×10^{-1}	2.3×10^{4}	1.5×10^{1}
✓	✓	✗	4.5×10^{-2}	2.2×10^{1}	7.8×10^{4}
✗	✗	✗	2.5×10^{-1}	1.2×10^{5}	8.6×10^{2}

Table 4.4: Resulting relative L^2 error, observation loss, and residual loss for each Inverse Case 1.5 ablation setup. The error estimates are based on single simulations.

This study shows that all three methods indeed improved the performance of the PINN, with gradient normalization emerging as the most impactful, followed by Fourier feature embedding and, lastly, random weight factorization. As previously mentioned, incorporating loss balancing with gradient normalization proves crucial in optimizing the model with loss terms of different scales. A more in-depth analysis of the ablation study results reveals a correlation: models with low L^2 error also exhibit low observation loss (calculated based on the observation set). Conversely, models with higher L^2 error tend to concentrate on minimizing the residual loss, which, by its formulation, is significantly larger than the observation loss.

The resulting inferred charge profile, together with the approximated solution is shown on the right-hand side in Figure 4.9. The PINN model struggles at $x = 0.5$ where the density drop occurs resulting in small over- and undershoot compared to the true charge profile. Otherwise, it manages to model both the potential and electrical field with the same accuracy as the reference COMSOL solution. The resulting mean relative L^2 error from ten different runs was 0.027 with a standard deviation of 0.0084, demonstrating great robustness.

However, as the steepness of the charge profile increases, so does the error in the PINN's inference of the charge-density profile. The left-hand side of Figure 4.9 illustrates the outcomes with an elevated steepness with $k = 500$. A comparison with the original experiment employing $k = 100$ reveals more pronounced deviations from the true charge profile, and interestingly, the predicted steepness for $k = 500$ closely resembles that of $k = 100$.

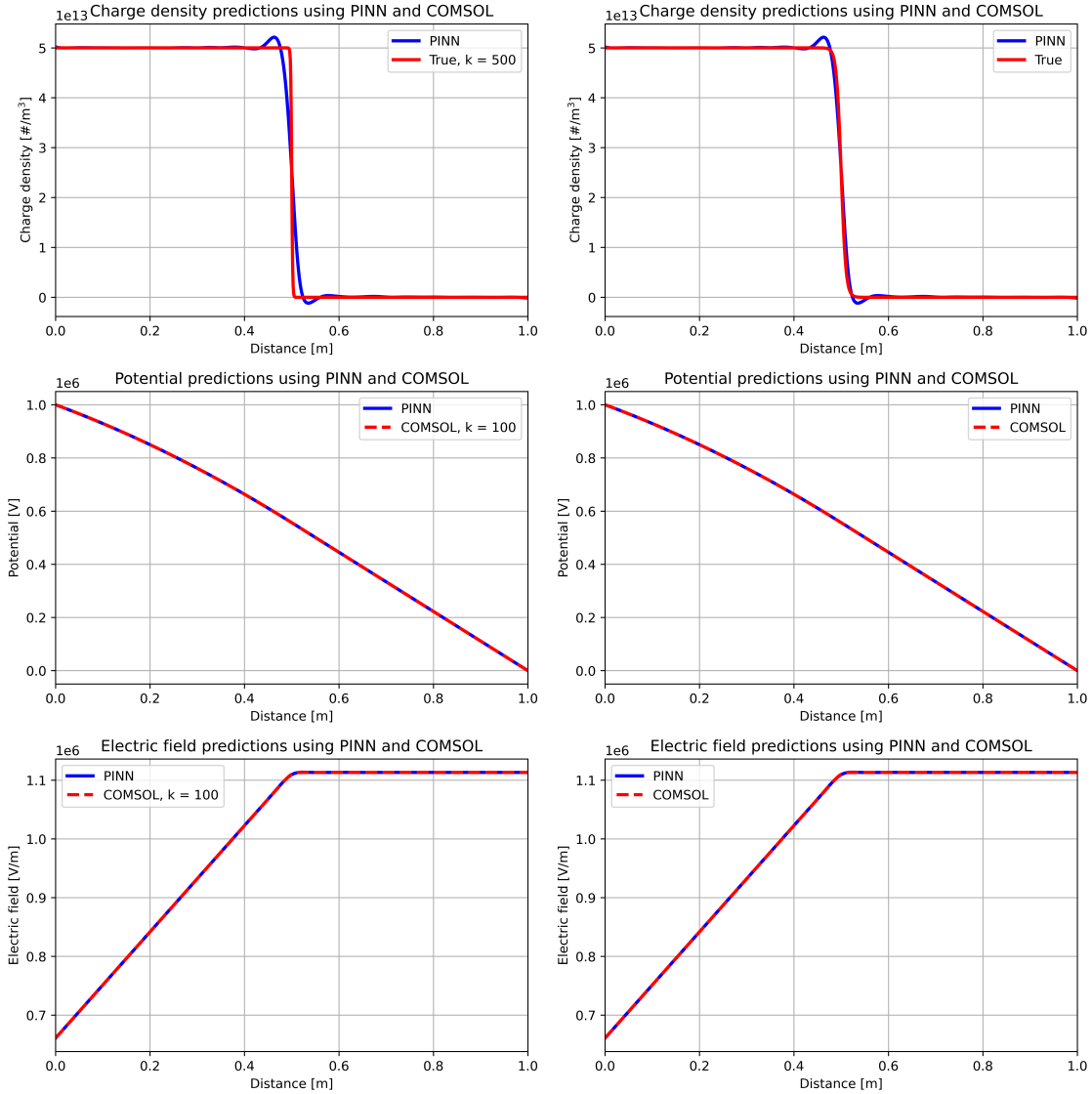

Figure 4.9: Inferred charge density, Predicted potential, and electrical field from the PINN and COMSOL solution. The left-hand side shows the PINN predictions with $k = 500$, and the results on the right-hand side are with $k = 100$. The COMSOL solutions for both sides are with $k = 100$.

One primary reason is that, while an increase in k significantly alters the charge profile, its impact on the potential distribution is comparatively small. Consequently, when training the PINN using potential data generated with $k = 500$, there is minimal distinction compared to data generated with $k = 100$. Additionally, the PINN models the potential distribution for $k = 100$ when trained on data

with $k = 500$, which further supports this claim. This implies that the PINN essentially learns from the same observational dataset in both cases, despite the variation in the steepness parameter, resulting in predictions that closely resemble each other.

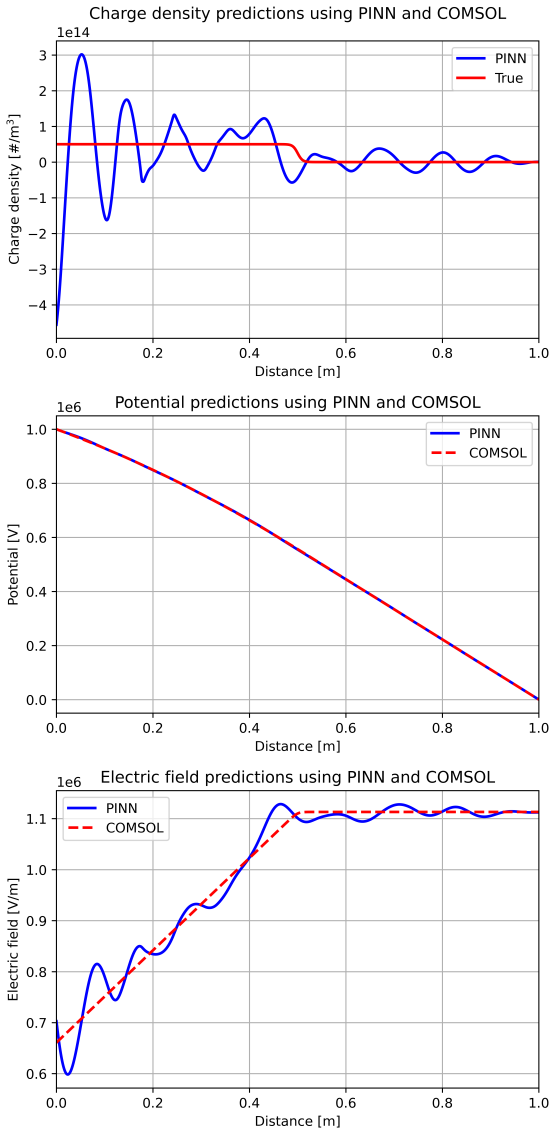

Figure 4.10: Inferred charge density, Predicted potential, and electrical field from the PINN operating on noisy observation data with 1% noise together with COMSOL solution.

A similar relationship between the charge profile and potential was shown in experiments involving noisy observation data. Training the PINN with only 1% added noise not only resulted in a failure to infer the underlying charge profile, but also worsened the predictions of the potential and resulting electrical field. This outcome, illustrated in Figure 4.10 reveals that the PINN predicts a high frequency for the potential graph rather than a smooth curve, indicative of overfitting to the observation data. Using loss balancing with gradient normalization, crucial for non-noisy data, now contributes to overfitting and generates unrealistic predictions for

the potential and electrical field. We hypothesize that the differing scales of our two loss terms, namely the observation loss and the residual loss, are too disparate. We also suggest for future work to analyze if this overfitting could be countered with the use of additional regularization methods, such as L^2 regularization. Additionally, we also suggest constraining the output of the PINN when predicting the charge profile to be strictly positive with the use of a different activation function in the output layer of the network. For example, by using the sigmoid activation function.

4.3 Case 2: Drift-Diffusion Equation

This section presents and analyzes the results for Case 2 (i.e., the drift-diffusion equation). The reference distribution for the relative L^2 error is calculated using the approximate analytical solution in Equation 3.14.

4.3.1 Forward: Solving the Drift-diffusion Equation

The results of the hyperparameter search for Case 2 are listed in Table 4.5. In Table 4.5, the activation column denotes the used activation function in the hidden layers. For the output layer, all models used the sigmoid activation function to exclude negative outputs. The hyperparameter configuration obtaining the lowest L^2 error was the sigmoid activation function, six hidden layers, and 512 neurons per layer, achieving an error of just under 5%. However, six other configurations performed similarly with an error of around 5%. The error was measured after 200,000 iterations of training, and at this stage, all the well-performing models oscillated slightly around the 5% mark. The graph of relative L^2 error per iteration for each model configuration is shown in Figure 4.11. Since all well-performing models converged to the same errors, their performance is considered equivalent.

Drift-Diffusion Grid Search Results			
#neurons	#Layers	Activation	Rel. L^2 error
256	4	Tanh	**0.052**
256	4	Gelu	0.746
256	4	Sigmoid	**0.051**
256	6	Tanh	0.624
256	6	Gelu	1.000
256	6	Sigmoid	**0.052**
512	4	Tanh	**0.053**
512	4	Gelu	**0.052**
512	4	Sigmoid	**0.050**
512	6	Tanh	0.747
512	6	Gelu	0.747
512	6	Sigmoid	**0.049**

Table 4.5: Resulting relative L^2 error for each drift-diffusion hyperparameter setup. The error estimates are based on single simulations with random initialization.

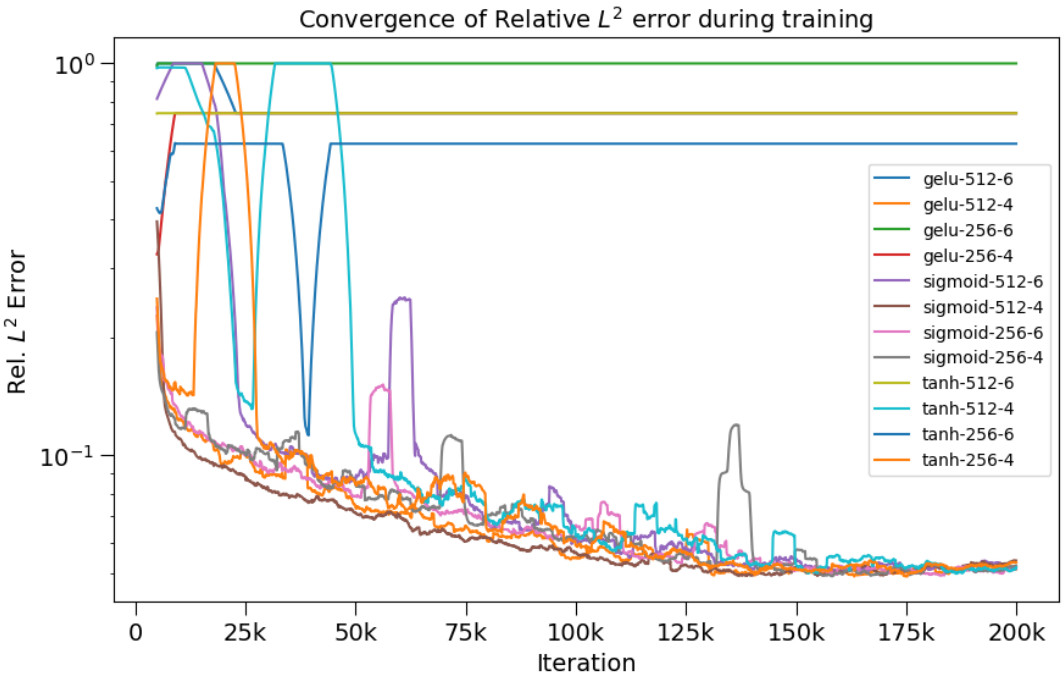

Figure 4.11: Convergence of relative L^2 error for different hyperparameters for the drift-diffusion equation. Labels are formatted as *activation function - width - depth*.

Another interesting finding was the effect of the activation function, where the sigmoid performs well for all layer and neuron setups. Tanh and GELU on the other hand were shown to be more sensitive towards the used number of layers and neurons per layer. Given that seven out of the twelve hyperparameter configurations performed well, the decision on the best setup was based on network size. Opting for a smaller network translates to fewer model parameters to optimize during training. By selecting the configuration with sigmoid, four hidden layers, and 256 neurons the computational overhead is reduced while still having the robustness shown by the sigmoid activation function.

In similarity with the finding from the hyperparameter search, the ablation study yielded similar relative L^2 errors for all evaluated methods. The result is shown in Table 4.6.

Interestingly, the standard PINN without additional methods performed slightly better than the others. Moreover, a smaller error was observed when applying all methods compared to when excluding Fourier features. This suggests that the inclusion of Fourier feature embedding indeed improved training. However, the results from the standard model seemingly contradict this finding. To further assess the impact of Fourier feature embedding, a dedicated run exclusively using this method was conducted and is shown in the second last row of Table 4.6. The increased error in this setup indicated that the method did not enhance performance. These results further strengthen the idea that the standard PINN, without any additional functionality, is sufficient to solve the drift-diffusion equation.

One performance-enhancing method that was shown not to be applicable to the drift-diffusion equation was loss balancing with gradient normalization. We hypoth-

Drift-Diffusion Ablation Study Results				
RWF	Causality Loss	Fourier Features	Grad. Norm.	Rel. L^2 Error
✓	✓	✓	✗	0.055
✗	✓	✓	✗	0.052
✓	✗	✓	✗	0.053
✓	✓	✗	✗	0.057
✗	✗	✗	✗	0.049
✗	✗	✓	✗	0.051
✗	✗	✗	✓	No convergence

Table 4.6: Resulting relative L^2 error for each drift-diffusion ablation setup. Each row corresponds to a separate run. A tick (✓) means that the method was used, and the cross (✗) means that it was excluded. The error estimates are based on single simulations with the same random initialization.

esize that the underlying reason for its inapplicability was caused by the fact that a straight line also satisfied the drift-diffusion partial differential equation. As a result of this, the PINN's residual loss could be minimized to exactly zero as the model predicts a straight line. However, as the residual loss term goes towards zero, the associated weight used in gradient normalization loss balancing goes towards infinity resulting in NaNs during training.

The resulting approximation of the drift-diffusion equation from the PINN is shown in Figure 4.12. The graph shows the predicted charge density for seven different time steps. Figure 4.13 compares the model predictions with the approximated analytical solution. The model accurately predicts each charge density drop for the equation with extreme accuracy and without any over-shoot or under-shoot (i.e. predicting charge densities greater than the injection density or negative densities), as a result of using the sigmoid activation function in the output layer. It should be noted that constraining the predictions from the PINN with the use of the sigmoid activation function in this specific case is possible since the upper bound of the output is known. Since the output from sigmoid is in the range $[0, 1]$, it can simply be scaled with the maximum value of the output domain to get the PINN predictions in the desired range. For a more general case, where the upper bound is not known, we suggest using a scaling factor well above the estimated maximum range. This would still constrain the output to be strictly positive and thereby eliminate under-shoot. However, this strategy has not been evaluated in this thesis and is left as future work.

To further exclude the impact of random initialization of the model parameters when training, the PINN model was trained ten times with different initialization seeds. The result showed great robustness towards different random initializations with a mean relative L^2 error of 0.049 and a standard deviation of only 0.0002.

The absence of metrics in [18] regarding the PINN solution quality complicates benchmarking against our model. Nevertheless, when compared to the findings in [18], the PINN demonstrates notable robustness and consistently converges to a

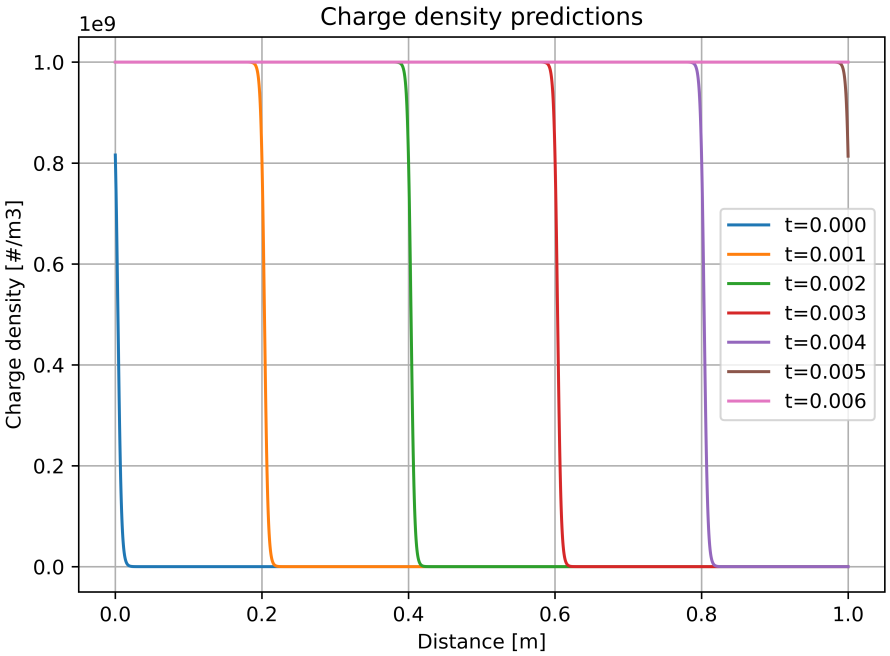

Figure 4.12: Predictions of the drift-diffusion equation with the best performing model setup.

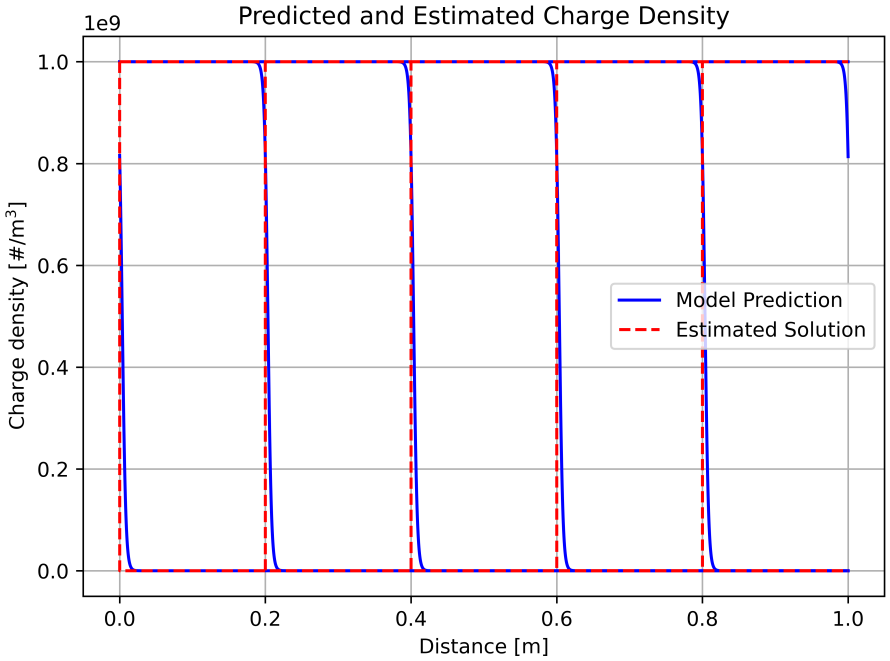

Figure 4.13: PINN predictions compared with the approximated analytical solution for the drift-diffusion equation.

satisfactory solution. This is likely the result of using significantly larger and thereby more complex networks than in [18]. However, such an increase in network size also increases the models' training times considerably.

4.3.2 Inverse: Inferring a Constant Ion Mobility

The goal of the inverse problem formulation for the drift-diffusion equation was to infer the ion mobility μ from observational data. The observation data was sampled from the approximated solution outlined in Equation 3.14. The dataset consisted of 6,400 points in total, 64 points for each of the 100 uniformly spaced time-steps. Employing the hyperparameter setup identified as the best-performing in Section 4.3.1, the ablation study produced the results documented in Table 4.7.

Inverse Drift-Diffusion Ablation Study Results				
RWF	Causality Loss	Fourier Features	Rel. L^2 Error	Rel. L^2 Param. Error
✓	✓	✓	2.01×10^{-2}	5.0×10^{-3}
✗	✓	✓	2.38×10^{-2}	5.0×10^{-3}
✓	✗	✓	1.98×10^{-2}	5.0×10^{-3}
✓	✓	✗	1.99×10^{-2}	5.0×10^{-3}
✗	✗	✗	3.45×10^{-2}	5.0×10^{-3}

Table 4.7: Resulting relative L^2 error and parameter error for each inverse drift-diffusion ablation setup. Each row corresponds to a separate run. A tick (✓) means that the method was used, and the cross (✗) means that it was excluded. The error estimates are from single experiments.

Following the results of the forward case, it can be observed that all models accurately infer the unknown ion mobility, with all models obtaining a parameter error of only 0.5% as displayed in the last column. Consequently, none of the performance-enhancing methods can be concluded to drastically increase or decrease the accuracy of the μ predictions. This ablation study was conducted once with one specific random initialization seed, and the effect of this specific seed may make the model infer the same value for the unknown ion mobility. For future work, this evaluation should be done more rigorously to make a more reliable assessment of the performance regarding inferring ion mobility. On the other hand, it can be observed that all performance-enhancing techniques indeed reduce the relative L^2 error for the predicted charge density. However, the obtained improvement from the methods is yet too subtle to affect the inferred ion mobilities.

A noteworthy observation is that the attained L^2 errors, which measure the solutions fit to the approximated analytical solution, for all inverse models are consistently lower than that observed in the forward case. One plausible explanation is that the observation data used to approximate the analytical solution is sampled from said solution, lacking additional noise. Consequently, this dataset provides precise values for each data point, supplying the model with information about the sought-after solution. Furthermore, the absence of added noise ensures a cohesive

relationship between the two loss terms. In real-world scenarios with actual measurements, it's challenging to achieve the precision observed in our simulated scenario due to the inherent variability introduced by measurements.

Relative L^2 parameter error				
Size \ Noise	1%	10%	25%	50%
100	0.024	0.023	0.025	0.019
6400	0.003	0.004	0.131	0.001

Table 4.8: Relative L^2 parameter error for different observation dataset sizes and noise levels. Error estimates are based on a single simulation with random initialization for each noise level and dataset size.

The sensitivity analysis for observation dataset size and noise levels is presented in Table 4.8, showing that the model can infer the parameter even at 50% uncorrelated Gaussian noise. While decreasing the number of observations worsens the predictions, the accuracy is still at an acceptable level. Thereby, the model demonstrates an ability to maintain a reliable performance despite restricting it to small datasets and low-quality data. However, the metrics shown in Table 4.8 should be considered as proof of concept rather than hard benchmark metrics. This is because the noise evaluation runs were conducted only once. Consequently, the surprisingly lower error with noisy data compared to noise-free data with a dataset size of 6400 might not reliably represent the model's performance under different runs or varied conditions. The resulting mean relative L^2 error of 0.034 and a standard deviation of 0.0005, and all with an inferred ion mobility error of only 0.5%.

To explore the boundaries of the robustness, the model was trained with data featuring 55% added noise, leading to a critical point where the model struggled to both infer the parameter and solve the PDE. In terms of dataset size, a reduction in the accuracy of inferred ion mobility was observed as the dataset size decreased. Yet, even with a minimal dataset comprising only 25 points, the relative L^2 parameter error stood at 15%, which could still be deemed acceptable depending on the application.

To summarize, the combined findings from the forward and inverse problems for the drift-diffusion equation show that PINNs can accurately solve the equation in isolation, both in forward and inverse mode. For the inverse formulation, the inference of the ion mobility proved accurate even when using few and noisy observations. Furthermore, given a sufficiently large network architecture—in our case, four layers with 256 neurons per layer—the performance-enhancing methods outlined in [23] are not required to solve the equation or infer the unknown constant. They may, however, have apparent impacts if a smaller network is employed.

4.4 Case 3: Sequential Coupled Case

The sequential model consists of two separate models; one U-model to predict the potential and one n-model to predict the charge density. In general, the optimal

configurations and architecture are likely to differ between the two models, and thus hyperparameter searches for the two should ideally be done separately. However, drawing from the results from Case 1 and Case 2 the distinction between the two models where limited to the activation function to avoid a combinatorial explosion. Consequently, the U-model was evaluated for GELU and tanh while the N-model was evaluated for the tanh and sigmoid activation functions. GELU was excluded from the N-model as it showed convergence problems in Case 2.

The results from this hyperparameter search are illustrated in Tables 4.9 and 4.10 for potential and charge density respectively. In these, the relative L^2 errors are calculated using the corresponding COMSOL solutions as reference points for the potential and the approximate analytical solution for the charge density (as the COMSOL solution displays significant numerical diffusion and thus is therefore inferior). As the COMSOL potential solution still shows some numerical diffusion, it should be noted that neither of these reference values is an exact solution. Nonetheless, they are believed to be expedient values for differentiating between hyperparameter values. The resulting best-performing model for the task of predicting the potential and charge density respectively was determined to be six layers of 256 neurons with GELU activation function (0.3% relative error) and six layers with 512 neurons and sigmoid activation function (4.8% relative error) respectively.

For the task of predicting the potential, Table 4.9 indicates that it is crucial for learning to have GELU as the activation function for the U-model. Furthermore, it seems as though using the sigmoid activation function for the n-model and a more narrow layer width achieves optimal performance. By contrast, Table 4.10 shows that the task of predicting the charge density n is more robust with regards to the choice of hyperparameters, with all but three configurations converging to a relative L^2-error between 4.6% and 5.8%. Nevertheless, the N-models had a significantly higher variance and slower convergence during training.

These results are reminiscent of those from Cases 1 and 2, where Laplace's equation showed greater sensitivity to hyperparameter tuning with a dependence on GELU while the drift-diffusion equation exhibited interchangeable loss values for most setups.

The results ablation study of proposed performing-enhancing techniques are exhibited in Figure 4.14 and 4.15. Notably, gradient normalization was excluded as it generated NaN values following minimizing the PDE residual loss for the drift-diffusion equation to zero, similar to in Case 2. Additionally, the modified loss function for respecting time causality was excluded from the ablation study and added to the default configuration with all other techniques as a separate run, following poor previous results indicating that it deteriorated performance rather than enhancing it.

The results of the ablation study are similar to those from Case 1 and Case 2, where random Fourier features significantly improved results for Poisson's equation, as visible in Figure 4.14, and RWF provided additional improvements when combined with the Fourier features. Notice also how random Fourier features significantly speed up the convergence time, displaying almost immediate convergence while the plain model requires some hundred thousand iterations before converging. By contrast, the drift-diffusion equation showed interchangeable results regardless

Case 3: Sequential Model Grid Search Results (Potential)				
U-model activation	n-model activation	#Layers	#Neurons	Rel. L^2 error
tanh	sigmoid	256	4	0.076
tanh	sigmoid	256	6	0.072
tanh	sigmoid	512	4	0.063
tanh	sigmoid	512	6	0.060
tanh	tanh	256	4	0.079
tanh	tanh	256	6	0.072
tanh	tanh	512	4	0.056
tanh	tanh	512	6	0.100
gelu	sigmoid	256	4	0.005
gelu	**sigmoid**	**256**	**6**	**0.003**
gelu	sigmoid	512	4	N/A
gelu	sigmoid	512	6	0.060
gelu	tanh	256	4	0.006
gelu	tanh	256	6	0.028
gelu	tanh	512	4	0.072
gelu	tanh	512	6	0.100

Table 4.9: Hyperparameter sweep for the potential of the sequential model for Case 3. Relative L^2 errors are calculated with COMSOL potential solution as a reference. Lowest error is bolded and runs with errors marked N/A failed to terminate.

of technique setups.

Sample predictions using the optimal architecture and both RWF and random Fourier features, but not gradient normalization as it caused NaNs for n-model and it is inapplicable for U-model as it only has one loss term, are shown in Figures 4.16 and 4.17. The results show that the sequential models manage to solve the coupled case, although with some inaccuracies for the potential. Figure 4.16 shows slight deviations between the PINN and COMSOL potential prediction that propagates to clear differences in the electric field. In particular, it seems as though the COMSOL electric field prediction is a better solution with straighter lines and less smearing than that of the PINN model. By contrast, Figure 4.17 shows that the PINN model is vastly superior in predicting the charge density profiles by almost completely eliminating numerical diffusion while having no over- or undershooting.

Given that the PINN formulation proposed by Konradsson [18] was unable to solve the coupled case for injections above 10^9 m^{-3} and therefore failed to model scenarios where the charge density profile significantly impacts the external electric field, these results constitute an important advancement. Moreover, solutions of equivalent quality were successfully obtained for injection upwards of 10^{16} m^{-3} indicating the strong numerical stability of the sequential PINN formulation for the coupled problem. Nevertheless, these results are omitted as larger injections reverse the electric field, suppressing further injections and breaking the physical interpretability of the model.

When running the sequential model for multiple seeds, however, it displayed significant sensitivity to initialization. Firstly, in three of the ten random initializa-

Case 3: Sequential Model Grid Search Results (Charge Density)				
U-model activation	n-model activation	#Layers	#Neurons	Rel. L^2 error
tanh	sigmoid	256	4	0.054
tanh	sigmoid	256	6	0.054
tanh	sigmoid	512	4	0.052
tanh	sigmoid	512	6	0.054
tanh	tanh	256	4	0.057
tanh	tanh	256	6	0.055
tanh	tanh	512	4	0.747
tanh	tanh	512	6	0.747
gelu	sigmoid	256	4	0.055
gelu	sigmoid	256	6	0.053
gelu	sigmoid	512	4	N/A
gelu	**sigmoid**	**512**	**6**	**0.048**
gelu	tanh	256	4	0.058
gelu	tanh	256	6	0.054
gelu	tanh	512	4	0.747
gelu	tanh	512	6	0.055

Table 4.10: Hyperparameter sweep for the charge density of the sequential model for Case 3. Relative L^2 errors are calculated with the approximative analytical solution as reference. Lowest error is bolded and runs with errors marked N/A failed to terminate.

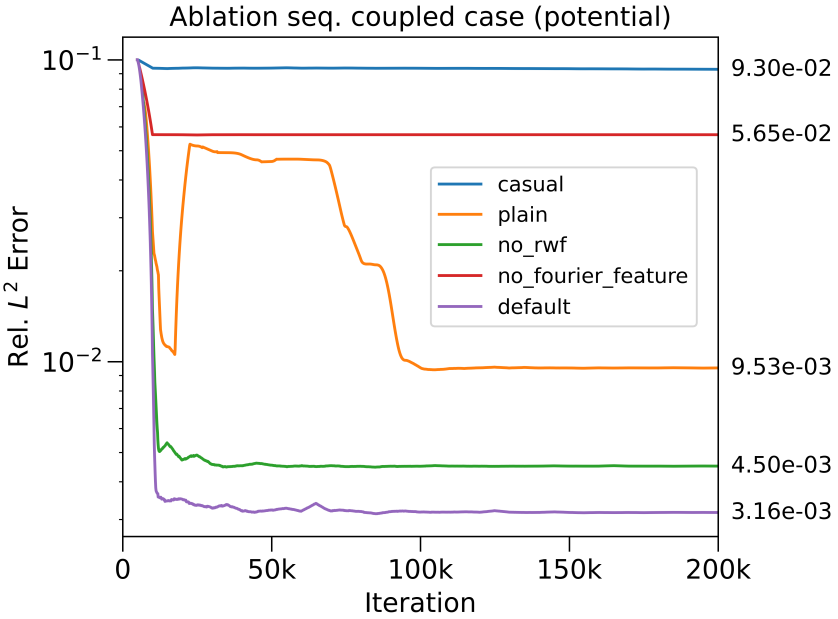

Figure 4.14: Ablation study results for the potential compared to COMSOL solution for sequential case 3. *Default* is using both RWF and random Fourier features, *plain* is using no performance enchancing techniques, and *casual* is *default* combined with enforcing time causality.

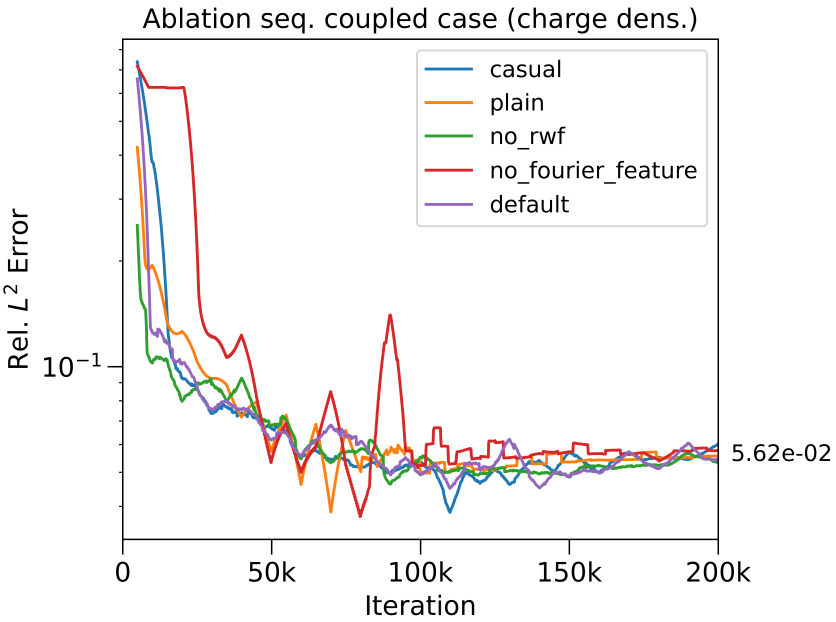

Figure 4.15: Ablation study results for the charge density compared to COMSOL solution for sequential case 3. *Default* is using both RWF and random Fourier features, *plain* is using no performance enchancing techniques, and *casual* is *default* combined with enforcing time causality.

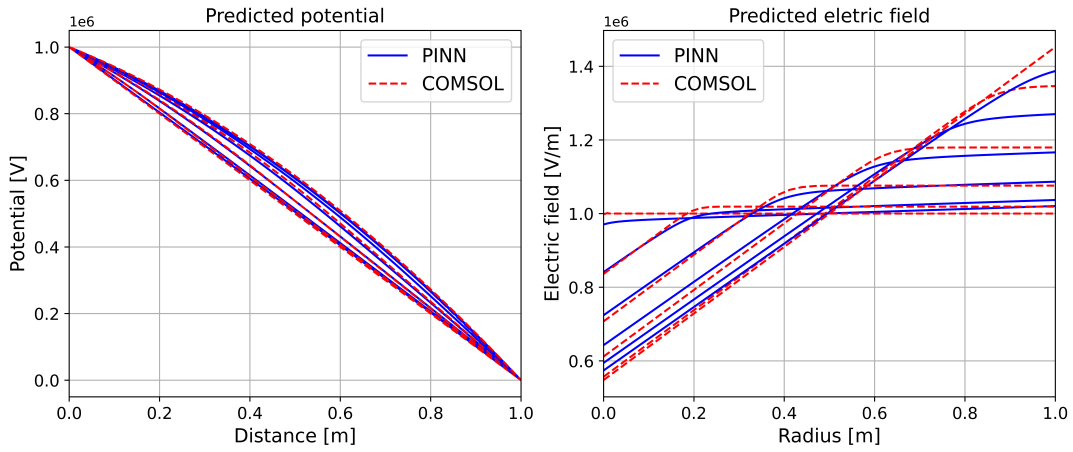

Figure 4.16: Comparison between COMSOL and PINN predictions of the potential and electric field for sequential case 3. Predictions are for timesteps $0, 0.001, \ldots, 0.006$.

tions, the U-model hit NaN values and failed to converge. This proved preventable by scaling down all loss terms with a factor 10^{-3} before computing the MSE, but this deteriorated performance across all runs somewhat indicating a fine line between performance and stability. Secondly, for the runs that did converge, there was significant variance. For the seven runs, the mean error for the potential was 0.033 with a standard deviation of 0.026 while for the charge density, the mean error was 0.226 with a standard deviation of 0.300. The runs are illustrated in Figures A.2

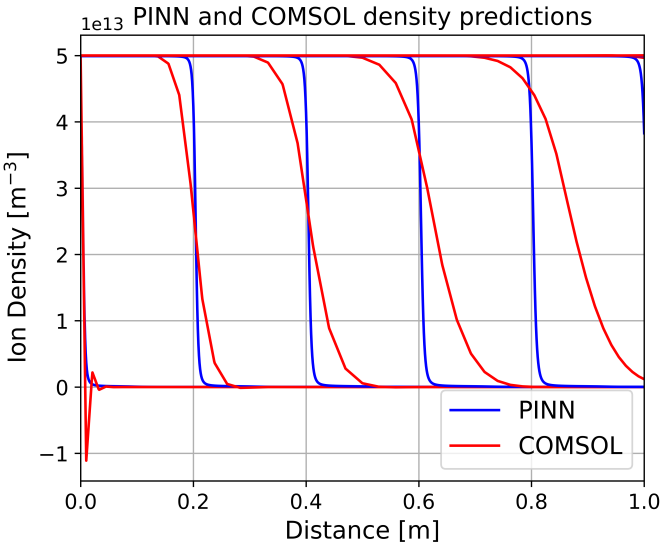

Figure 4.17: Comparison between COMSOL and PINN predictions of charge density for sequential case 3. Predictions are for timesteps $0, 0.001, \ldots, 0.006$.

and A.3 in Appendix A and show that while some initializations converge with great accuracy, some show significantly worse performance resulting in high variance.

While the cause of this high variance, which was not present in the other cases, is unknown several potential opportunities to increase stability exist. Low-hanging fruit includes conducting a full-scale hyperparameter search for network parameters as well as performance-enhancing technique parameters, where only default values were used in this thesis. Moreover, exploring alternative sampling strategies, such as RAR proposed in [41], and additional optimization algorithms, such as L-BFGS which is popularly combined with Adam within the field, would be worth exploring.

4.4.1 Non-sequential and Inverse Models

Similar to the results of Konradsson [18], single (i.e., non-sequential) PINN models for the coupled PDEs in Case 3 failed to learn for injections of ions strong enough to significantly affect the potential. In particular, for injections exceeding 9.5×10^9 m^{-3} predictions of the charge density started showing significant numerical diffusion. No attempt at the inverse coupled case using such a model was used, as the solution is contingent on being able to solve the forward case.

Moreover, the coupled equivalence of inverse Case 2, i.e., discovering the unknown ion mobility μ using observations of the potential U, was deemed infeasible for a sequential model. The drift-diffusion equation simply describes the propagation of charge carriers in an electric field and therefore it is clearly solvable for an arbitrary ion mobility μ as it applies to everything from electrons to large ions, such as the sulfate ion SO_4^{2-}, with very different ion mobilities. Additionally, it is solvable for an arbitrary electric field $\mathbf{E}$ even with our initial and boundary conditions. Consequently, the global minimum of the n-models loss is independent of μ and the parameter will not be learned during training. Needlessly to say, an incorrect value of μ will drive loss in the U-model as the observed potential will misalign with the

potential minimizing the residual loss (that is dependent on the charge density n). However, as none of the loss terms in the U-model is directly dependent on μ, this parameter cannot be changed to minimize this loss. This predicament is summarized in Figure 4.18.

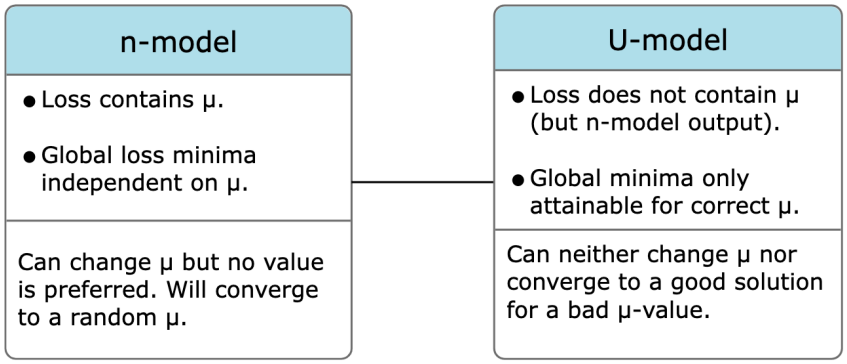

Figure 4.18: Inferring the ion mobility μ from observations of the potential is infeasible.

Of course, discovering an unknown μ using observations of the charge density n instead of the potential U avoids this issue and works well empirically. However, this is conceptually identical to Case 2, as it is solvable without accounting for the coupling, and therefore omitted.

4.5 Summary of Best Practices

This section summarizes key results into a list of best practices for using PINNs to solve Poisson's equation and the drift-diffusion equation within discharge physics. We hypothesize that they are applicable more generally within the domain as well.

- Random Fourier features significantly improved PINN performance on Poisson's and Laplace's equations and enabled solving for much smaller radii. Random weight factorization (RWF) further improved performance when paired with random Fourier features.
- The GELU activation was crucial for solving Laplace's and Poisson's equations.
- Constraining the output from the PINN by using a bounded activation function in the network's output layer increased the robustness of the model and eliminated predictions outside the desired range. This was applied to the drift-diffusion equation with the use of the sigmoid activation function.
- Gradient normalization was crucial for solving inverse cases with multiple loss terms as the observation loss tended to be of a different magnitude to, for instance, the residual loss.
- In accordance with [23], employing a network width of 256-512 neurons per layer resulted in PINN architectures sufficiently large for solving the equations.
- The strongly coupled case of Poisson's equation and the drift-diffusion equation with large injections could be solved using a sequential learning approach when the coupled model failed to converge.

4.6 Future Work

Physics-informed neural networks are still in their infancy and there is much room for future research. In particular, for the domain of discharge physics, this thesis has merely scratched the surface and there are multiple natural continuations within the physics and the machine learning alike.

On the physics side, a natural continuation would be to reduce the simplifications used in this work. Firstly, the models could be extended to more complex geometries, for instance, three spatial dimensions. Secondly, more complex physical reactions, such as ionization and recombination effects, could be added to the model. Thirdly, the observations used for the inverse models in this thesis are of the potential and charge density in the air at specific coordinates. However, measuring these entities in the air is very tedious in practice. Consequently, a future study could investigate whether the same inference of hidden physics could be achieved using observations that could be obtained by measuring the currents at the electrodes, that is an indirect measure of both ionic charge transport and so called displacement current that is a measure of the time derivative of the electrical field. Finally, there are plenty of opportunities to develop new inverse cases and to push the boundaries for what can be predicted. For instance, exploring opportunities to discover multiple unknown parameters simultaneously is an interesting challenge.

On the machine-learning side, natural continuations include solving the coupled Case 3 without using sequential training and increasing the stability and accuracy of the existing models and cases. Following the argument in Section 4.4.1, removing the dependency on sequential training is necessary to enable inverse problems where the parameters related to one PDE are inferred from observations of the other and therefore crucial for tackling more complex problems. A first improvement opportunity for all models is to conduct a full-scale hyperparameter search for the performance-enhancing techniques. Each of them has its hyperparameters that need tuning and in this thesis, only a few values were tried empirically. Thus, the reason why RWF and enforcing time causality worked poorly for many of the cases might just be that they work misconfigured for the tasks.

Secondly, it would be interesting to pair the proposed performance-enhancing techniques with optimization using first the Adam algorithm and then the L-BFGS algorithm, as opposed to just Adam as was the case in this thesis. By combining Adam and L-BFGS, accuracy and convergence properties have been shown to improve significantly in, for instance, [18].

Finally, exploring the impact of more advanced sampling strategies for collocation points, such as residual-based adaptive refinement (RAR) [41], would be interesting. This has the potential to drastically improve performance for problems where the challenging part of the PDE or the important information for inference is restricted to a small portion of the domain (e.g., Case 1).

5
Conclusion

This study set out to investigate how the PINNs proposed in [18] can be improved for forward problems within discharge physics and whether they can be extended to solve related inverse problems. In particular, for the forward problem, the aim was to be able to model strongly non-uniform concentration profiles for charge carriers and potential fields as well as to solve the coupled equations for the potential and charge distributions. For the inverse problem, the aim was to investigate the ability to model hidden parameters and distributions and explore their sensitivity to the number and quality of observations.

For forward problems, the findings show that employing random Fourier features and switching to a GELU activation function, enables accurate modeling of significantly steeper gradients for the potential in Laplace's equation. Specifically, for Case 1, the inner electrode was successfully decreased from 5mm to 0.1mm, increasing the electrical field by a factor of 40. Moreover, for the drift-diffusion equation, the results show that by applying the sigmoid activation to the output, the steep charge distribution profile could be modeled with extreme steepness while eliminating over- and undershoot completely. Thus, it highlights the ability of PINNs to address the tendency of numerical diffusion commonly observed in conventional methods such as FEM. Finally, show how the sequential training algorithm proposed by [42] can be used to successfully solve the coupled case for injections large enough to significantly affect the external electric field.

Moreover, for the inverse case, this study demonstrates that by employing gradient normalization PINNs can successfully discover both hidden physical parameters, e.g., geometries, ion mobilities, and ion densities, and hidden distributions, e.g., charge density profiles. Additionally, the models exhibit high accuracy even with few and noisy observations when the observations are directly tied to the hidden physical parameter.

As such, PINNs show promise as an emergent technique for solving PDEs and discovering hidden physics within the domain of discharge physics. Nevertheless, conventional numerical methods, such as FEM, and PINNs are still far from constituting a substitute for these in practice, highlighting the need for continued research. For example, PINNs lack the error estimations and convergence guarantees offered by FEM.

For the application of PINNs within discharge physics, several areas remain to be explored. Physically, the extension of this work to higher dimensions and more complex settings, such as drift-diffusion with high diffusion, should be considered. Additionally, extending the inverse cases to observations easier to measure experimentally would be a significant improvement for real-world applications. From

a machine-learning perspective, exploring more complex sampling strategies and second-order optimizers, such as L-BFGS, would be fruitful areas for future work.

Bibliography

[1] M. Abdel-Salam and E. Z. Abdel-Aziz, "Corona power loss determination on multi-phase power transmission lines," *Electric Power Systems Research*, vol. 58, no. 2, pp. 123–132, 2001, ISSN: 0378-7796. DOI: https://doi.org/10.1016/S0378-7796(01)00106-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779601001067.

[2] S. Singh, *Computational framework for studying charge transport in high-voltage gas-insulated systems*. (Doktorsavhandlingar vid Chalmers tekniska högskola. Ny serie: 4296). Chalmers University of Technology, 2017, ISBN: 9789175976150. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat09075a&AN=clpc.oai.edge.chalmers.folio.ebsco.com.fs00001000.f034e059.2d7a.4e0e.9b88.93c0818b63e3&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[3] K. Georgii, *Simulation and analysis of corona currents in large scale coaxial geometry due to triangular voltages*, 2019. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=ir01625a&AN=cst.20.500.12380.188939&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[4] J. Schiessling, O. Hjortstam, U. Gäfvert, M. Saltzer, and Y. V. Serdyuk, "Measurements and simulations of corona currents due to triangular voltages in large scale coaxial geometry," in *2012 Annual Report Conference on Electrical Insulation and Dielectric Phenomena*, 2012, pp. 357–360. DOI: 10.1109/CEIDP.2012.6378794.

[5] C. H. Taubes, *Modeling Differential Equations in Biology*, 2nd ed. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511811364.

[6] S. L. Heston, "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options," *The Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, Apr. 2015, ISSN: 0893-9454. DOI: 10.1093/rfs/6.2.327. eprint: https://academic.oup.com/rfs/article-pdf/6/2/327/24417457/060327.pdf. [Online]. Available: https://doi.org/10.1093/rfs/6.2.327.

[7] D. Gomes, R. M. Velho, and M. T. Wolfram, "Socio-economic applications of finite state mean field games," *Philos Trans A Math Phys Eng Sci*, vol. 372, no. 2028, Nov. 2014.

[8] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bulletin of the American Mathematical Society*, vol. 49, no. 1, pp. 1–23, 1943.

[9] A. Hrennikoff, "Solution of Problems of Elasticity by the Framework Method," *Journal of Applied Mechanics*, vol. 8, no. 4, A169–A175, Mar. 2021, ISSN: 0021-8936. DOI: 10.1115/1.4009129. eprint: https://asmedigitalcollection.asme.org/appliedmechanics/article-pdf/8/4/A169/6744200/a169_1.pdf. [Online]. Available: https://doi.org/10.1115/1.4009129.

[10] C. Multiphysics, "Introduction to comsol multiphysics®," *COMSOL Multiphysics, Burlington, MA, accessed Feb*, vol. 9, p. 2018, 1998.

[11] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb, *Can physics-informed neural networks beat the finite element method?* 2023. arXiv: 2302.04107 [math.NA].

[12] A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, "Physics-informed neural networks for inverse problems in supersonic flows," *Journal of Computational Physics*, vol. 466, p. 111 402, 2022.

[13] S. Cuomo, V. S. D. Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics-informed neural networks: Where we are and what's next," *CoRR*, vol. abs/2201.05624, 2022. arXiv: 2201.05624. [Online]. Available: https://arxiv.org/abs/2201.05624.

[14] D. Katsikis, A. D. Muradova, and G. E. Stavroulakis, "A gentle introduction to physics-informed neural networks, with applications in static rod and beam problems," *Journal of Advances in Applied & Computational Mathematics*, vol. 9, pp. 103–128, 2022.

[15] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," *Advances in neural information processing systems*, vol. 30, 2017.

[16] Z. Fang, "A high-efficient hybrid physics-informed neural networks based on convolutional neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5514–5526, 2021.

[17] M. Islam, M. S. H. Thakur, S. Mojumder, and M. N. Hasan, "Extraction of material properties through multi-fidelity deep learning from molecular dynamics simulation," *Computational Materials Science*, vol. 188, p. 110 187, 2021.

[18] A. Konradsson, *Physics-informed neural networks for charge dynamics in air*, 2020. [Online]. Available: https://odr.chalmers.se/items/649c0043-7e55-44f4-86b5-fb2a40601f9a.

[19] A. Heilmann and C. Dartora, "The relation between dielectric breakdown and transported power density in high-voltage transmission lines," *Revista Brasileira de Ensino de Física*, vol. 36, Aug. 2014. DOI: 10.1590/S1806-11172014000400012.

[20] Y. V. Serdyuk, M. Macken, O. Hjortstam, B. Källstrand, U. Gefvert, and M. Saltzer, "Charging insulating barrier by corona in air in large coaxial system.," *2015 IEEE Conference on Electrical Insulation & Dielectric Phenomena (CEIDP)*, pp. 705–708, 2015, ISSN: 9781467374965. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=edb&AN=112657223&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[21] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[23] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," *arXiv preprint arXiv:2308.08468*, 2023.

[24] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[25] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019, ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html.

[27] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu, *Towards understanding the spectral bias of deep learning*, 2020. arXiv: 1912.01198 [cs.LG].

[28] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113 938, Oct. 2021. DOI: 10.1016/j.cma.2021.113938. [Online]. Available: https://doi.org/10.1016%2Fj.cma.2021.113938.

[29] M. Tancik *et al.*, *Fourier features let networks learn high frequency functions in low dimensional domains*, 2020. arXiv: 2006.10739 [cs.CV].

[30] S. Wang, H. Wang, J. H. Seidman, and P. Perdikaris, *Random weight factorization improves the training of continuous neural representations*, 2022. arXiv: 2210.01274 [cs.LG].

[31] T. Salimans and D. P. Kingma, *Weight normalization: A simple reparameterization to accelerate training of deep neural networks*, 2016. arXiv: 1602.07868 [cs.LG].

[32] S. Wang, S. Sankaran, and P. Perdikaris, *Respecting causality is all you need for training physics-informed neural networks*, 2022. arXiv: 2203.07404 [cs.LG].

[33] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, *Physics-informed neural networks with hard constraints for inverse design*, 2021. arXiv: 2102.04626 [physics.comp-ph].

[34] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Net-*

works, vol. 9, no. 5, pp. 987–1000, 1998, ISSN: 1045-9227. DOI: 10.1109/72.712178. [Online]. Available: http://dx.doi.org/10.1109/72.712178.

[35] P. L. Lagari, L. H. Tsoukalas, S. Safarkhani, and I. E. Lagaris, "Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions," *International Journal on Artificial Intelligence Tools*, vol. 29, no. 05, p. 2 050 009, 2020. DOI: 10.1142/S0218213020500098. eprint: https://doi.org/10.1142/S0218213020500098. [Online]. Available: https://doi.org/10.1142/S0218213020500098.

[36] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, no. 1, p. 6, 2008.

[37] S. W. Ellingson, *Electromagnetics, Vol. 1*. VT Publishing, 2018. DOI: https://doi.org/10.21061/electromagnetics-vol-1.

[38] J. Bradbury *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: http://github.com/google/jax.

[39] J. Heek *et al.*, *Flax: A neural network library and ecosystem for JAX*, version 0.7.5, 2023. [Online]. Available: http://github.com/google/flax.

[40] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038%2Fs41586-020-2649-2.

[41] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021. DOI: 10.1137/19M1274067.

[42] S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri, "Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113 959, 2021.

A

Additional Plots and Tables

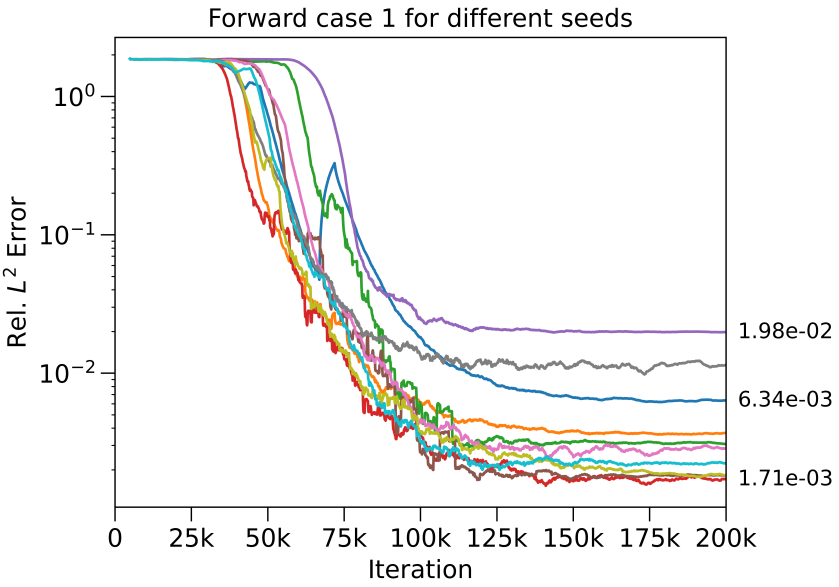

Figure A.1: Convergence results for forward Case 1 using initialization with 10 different seeds. Results are a running average with a window size 50 to reduce batch-related variance.

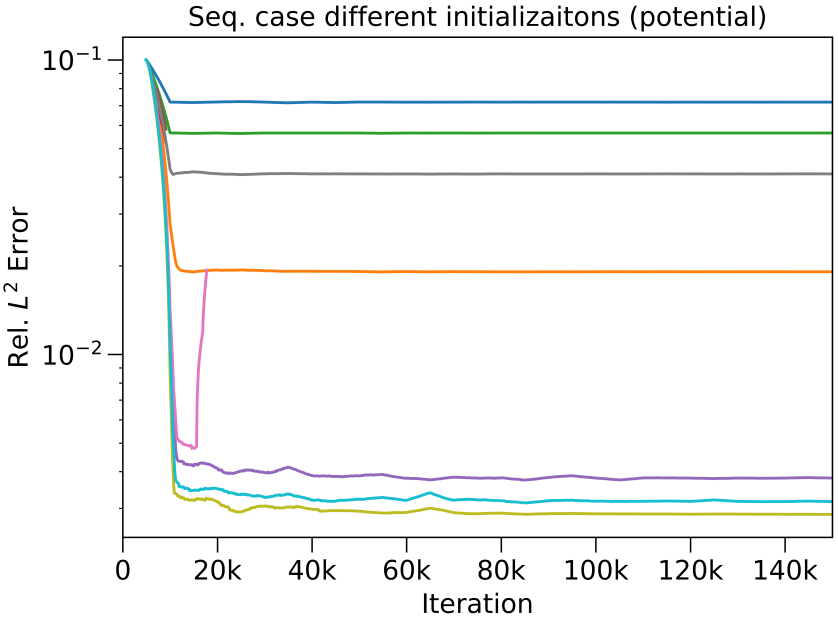

Figure A.2: Sequential model predictions for the potential in Case 3 using 10 different seeds. Shows sensitivity to initialization and three runs terminate with NaN values, although preventable by scaling down losses by 10^{-3}.

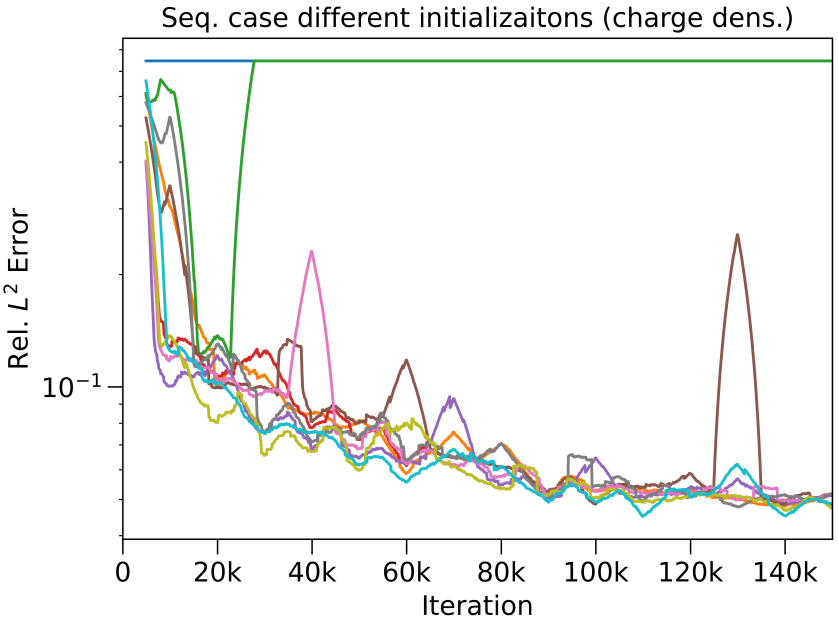

Figure A.3: Sequential model predictions for the charge density profile in Case 3 using 10 different seeds.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY