Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Aurora Sletnes Bjørlo

Physics-Informed Neural Networks for Modeling of Electric Submersible Pumps in Oil Wells

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Struen Imsland
Co-supervisor: Eduardo Camponogara
August 2023

NTNU
Norwegian University of
Science and Technology

UFSC

Aurora Sletnes Bjørlo

Physics-Informed Neural Networks for Modeling of Electric Submersible Pumps in Oil Wells

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Struen Imsland
Co-supervisor: Eduardo Camponogara
August 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Physics-Informed Neural Networks (PINNs) are Neural Networks (NNs) with the ability to incorporate known governing physical equations into their training process, enabling them to efficiently model physical systems, even with limited access to data. The numerous benefits of PINNs render them an attractive choice for control applications. The Physics-Informed Neural Net for Control (PINC) is a modification of the original PINN that can be used for this. This thesis attempts to leverage the inherent ability of physics-informed machine learning to integrate known physical principles, to model the behavior of an Electric Submersible Pump (ESP). This is done by creating a PINN model and a PINC model for the system. The experimental results of the PINN model demonstrate comparability with solutions derived from RK4. This is further affirming its capacity to efficiently solve complex problems while adhering to the underlying physics of the system, in line with observations from previous research. However, the implemented PINC model exhibits more varied outcomes and a general increase in the deviation between the model output and the solution from RK4 over the course of simulation time. Although the PINC model demonstrates potential, its performance requires further investigation. The challenges encountered in effectively training the PINC and ensuring reliability when applied to the ESP system, suggest the necessity of continued investigation and optimization efforts before the practical application of such models.

Keywords: Physics-Informed Neural Networks; Physics-Informed Neural Net for Control; Electric Submersible Pump; Modeling Complex Systems

Sammendrag

Physics-Informed Neural Networks (PINN) er nevrale nettverk som inkluderer kjente fysiske sammenhenger inn i treningsprosessen sin, slik at de kan modellere fysiske systemer effektivt, selv med begrenset tilgang til systemdata. PINNs har mange fordeler som gjør dem egnet til bruk i reguleringssystemer. Physics-Informed Neural Nets for Control (PINC) er en modifikasjon av den opprinnelige PINN formuleringen som muliggjør denne typen bruk. Denne oppgaven forsøker å utnytte den naturlige evnen fysikkbasert maskinlæring har til å integrere fysiske sammenhenger, for å modellere oppførselen til en Electric Submersible Pump (ESP). Dette gjøres ved å lage en PINN-modell og en PINC-modell for systemet. De eksperimentelle resultatene fra forsøkene gjort med PINN-modellen viser at de oppnådde resultatene i stor grad kan sammenlignes med løsningene funnet ved bruk av RK4. Dette kan ses som en bekreftelse på evnen PINN har til å effektivt løse komplekse problemer, samtidig som de underliggende fysiske lovene respekteres, i tråd med obserasjoner gjort i tidligere forskning. Imidlertid viser den implementerte PINC-modellen mer varierte resultater, og det kan observeres en generell økning i avviket mellom modellutgangen og løsningen fra RK4 gjennom simuleringstiden. Derfor, selv om PINC-modellen viser potensiale, er det nødvendig med videre undersøkelser angående kapasiteten til modellen. Utfordringene som oppstår i forsøket på å trene og sikre påliteligheten til PINC-modellen for ESP-systemet, understreker nødvendigheten av videre undersøkelser og forsøk på optimalisering før slike modeller kan anvendes i praksis.

Nøkkelord: Physics-Informed Neural Networks; Physics-Informed Neural Net for Control; Electric Submersible Pump; Modellering av komplekse systemer

Acknowledgements

Firstly I would like to thank my supervisor Eduardo Camponogara from Universidade Federal de Santa Catarina for not only providing me with an interesting research topic but also for his guidance and feedback, and for his support settling in when I arrived in Brazil. I also want to thank my supervisor from the Norwegian University of Science and Technology, Lars Imsland, for the advice and feedback he has provided me. I would also like to thank Eric Antonelo from UFSC for helpful discussions. I also want to give thanks to the people at the lab, for always helping if needed, and especially Bruno for his valuable insights on implementing PINNs and PINCs.

A special thanks to the friends I made in Brazil, both for the wonderful times we had together and for showing me some of the Brazilian culture. Especially I want to thank Laura and Felipe, who became dear friends that I hope to meet again soon. I also want to thank Ícaro, my surf coach, who I also consider a friend. Not only did he treat me as part of his family, but he also inspired me with his passion and belief that we all have the possibility to change the lives of others for the better.

I also need to thank Maren and Ola. I am forever grateful for our time exploring and enjoying Florianópolis together, and for having had someone to navigate the new city and country with.

Finally, I would like to thank my family and friends back in Norway for all their support during this final semester of my studies.

Aurora Sletnes Bjørlo

Hamar, August 16, 2023

Contents

List of Tables

List of Figures

Acronyms

AI Artificial Intelligence.

DAE Differential-Algebraic Equation.

ESP Electric Submersible Pump.

IVP Initial Value Problem.

L-BFGS Limited-Memory Broyden-Fletcher-Goldfarb-Shanno.

ML Machine Learning.

MPC Model Predictive Control.

PINC Physics-Informed Neural nets-based Control.

PINN Physics-Informed Neural Network.

ReLU Rectified Linear Unit.

RK4 Runge-Kutta 4.

SGD Stochastic Gradient Descent.

tanh hyperbolic tangent.

Chapter 1

Introduction

Artificial Intelligence (AI) has a rich history, despite the field only being formally established in 1956 [16]. The idea of being able to create intelligence in machines has intrigued many scientists throughout the years, due to the enormous potential such technology brings along [17]. Machine Learning (ML) is the subfield of AI that focuses on enabling computers to learn and make predictions or decisions without being explicitly programmed. Major changes are occurring in a variety of sectors including healthcare, banking, manufacturing, and entertainment as a result of AI-powered applications that streamline processes and improve decision-making [36]. This manifests in a variety of applications. Financial institutions may use ML algorithms for risk management [23] and fraud detection [42], while healthcare providers use AI to help with disease diagnosis [19], prognosis [15], and treatment planning [21].

In our daily lives, we surround ourselves with systems rooted in AI-based technologies as well [28]. Applications such as smart devices, virtual assistants, and recommendation systems showcase the impact AI already has in today's society. Yet, this promising field within technology has a transformational potential that is still far from being realized. As AI evolves, its ability to change a wide range of fields such as healthcare and transportation holds the possibility of tackling several major global concerns, as well as altering numerous current businesses.

However, the remarkable growth of AI and machine learning is accompanied by a serious challenge: the availability and quality of data [31]. The current developments are driven by algorithms and models, but their effectiveness depends on access to grand volumes of data for training and tuning. A lack of data, or data of poor quality, might cause models to miss intricate patterns or to reinforce biases. The issue of limited data is prominent within a span of sectors. In the healthcare industry, access to thorough patient records may be prohibited due to data privacy [35], while other sectors such as emerging markets or specialized industries might face problems due to not having the huge datasets required for effective AI integration [40]. The lack of data creates a bottleneck that prevents AI and ML from reaching their full potential.

Industrial systems might often be prone to a shortage of data, due to a number of factors [29]. Limited sensor coverage, outdated equipment without the ability to generate data, worries about data security and confidential information, and difficulties combining and integrating data from a variety of different sources can all be contributors to a lack of data for such systems. One consequence of the lack of complete and current data is that it can cause difficulties in establishing and implementing data-driven models of the system at hand. This can lead to a reduction in the effectiveness of decision-making, preventive maintenance, and overall operations. It also causes difficulties in the event that a controller is to be implemented for the system. However, recent advances within ML might propose a solution to this problem.

The Physics-Informed Neural Network (PINN), after being proposed in 2017 by Raissi et al. [24], has gained significant attention due to its inherent capability of integrating domain-specific knowledge and data-driven modeling [44]. PINNs leverage neural networks to learn the underlying physical laws governing a system. By incorporating physics-based constraints into the loss function used during the training process, PINNs can accurately predict the behavior of complex systems, even in the presence of limited or noisy data.

Due to their capacity for precise and rapid prediction of intricate system dynamics, PINNs show great promise for utilization in control applications. However, the original PINN formulation is unsuitable for such usage, as the input does not include the initial condition of the system and the control input. The Physics-Informed Neural nets-based Control (PINC) architecture, introduced by Antonelo et al. in [43], presents a novel PINN approach, specifically designed to tackle control problems with the added advantage of simulating longer-range time horizons that are not predetermined.

The Electric Submersible Pump (ESP) has a foothold in various industries, with applications ranging from oil and gas production [8], to wastewater treatment [10], and the dewatering of mines [4]. Efficient and reliable control of ESPs is crucial to ensure that they have optimal performance and smooth operation. However, these systems tend to have complex dynamics, and may also be subject to uncertainties. Further complicating the matter, access to process data for modeling tends to be limited. This may result in less-than-optimal performance using traditional control methods, as they may fail to capture the nuances of the system, especially in the case of limited data. However, in order to effectively control such systems, a model that properly describes the dynamics of the system is needed.

This thesis aims to explore the development, training, and application of PINNs and PINCs for the modeling of ESPs, leveraging the inherent physics knowledge and the capacity of neural networks to learn complex relationships from data. By integrating physics-based constraints into the neural network architecture, PINNs, and PINCs can provide accurate and robust models even when data is limited, enabling advanced control strategies such as Model Predictive Control (MPC) to enhance the performance and efficiency of ESP systems.

1.1 Research Objectives and Contributions

This thesis aims to study how a PINN can be used to predict the behavior of an ESP, and furthermore how it can be extended into a PINC to be used in a control setting.

Through the development of these models, this thesis intends to advance the understanding of the PINN and PINC model, their benefits and limitations. Furthermore, these models are to be trained and applied to an ESP system, thus contributing to the development of more efficient and reliable systems in a wide range of industrial domains.

Another objective of this thesis is to apply the PINC model together with an MPC for the ESP system. However, this is not achieved due to the developed PINC model not achieving sufficient accuracy for such applications.

1.2 Thesis Outline

This thesis is divided into 8 chapters:

- Chapter 2 introduces relevant theoretical concepts for this thesis. It includes a brief introduction to differential equations and numerical solvers, as well as background on neural networks. It further elaborates on the concepts of PINN and PINC.
- Chapter 3 gives a brief theoretical background on the ESP. It also describes the ESP simulator development, and verification. This chapter is taken from the pre-work for this thesis [51] and is presented here as well for completeness.
- Chapter 4 outlines the PINN structure and the development, training, and verification of this model for the ESP.
- Chapter 5 describes the development of a PINC for the ESP, as well as the training and verification of the model.
- Chapter 6 presents and evaluates the results acquired using the models developed in this work.
- Chapter 7 discusses the obtained results, as well as their implications for practical applications.
- Chapter 8 summarizes and concludes the thesis. Here, a brief outlook on future research directions is also presented.

Chapter 2

Theoretical Background

This chapter presents the theoretical foundation of this thesis. The focus of this thesis is the application of a PINN and a PINC to an ESP, which is a system with physics that can be described by a set of differential equations. A brief introduction to differential equations and the initial value problem is therefore given, with emphasis on the numerical methods that exist for solving such problems, as these will be used as the ground truth against which the developed models will be compared. Furthermore, an introduction to neural networks is given, before the theoretical background of PINNs and PINCs is presented. While some parts of the sections regarding neural networks, PINN and PINC are based upon the work developed in the prework for this thesis [51], they are in large part rewritten and reworked for this thesis and its objective.

2.1 Differential Equations

Differential equations play a crucial role in many engineering applications that involve describing the evolution of processes over time. In order to define such applications accurately, two distinct pieces of information are required: the knowledge of the process itself specified by differential equations, and the initial conditions, which indicate the starting point or the conditions at a specific time or location. A problem where both of these are defined is referred to as an Initial Value Problem (IVP), and such problems provide a powerful framework for predicting the behavior of dynamic systems.

2.1.1 The Initial Value Problem

Within differential equations and mathematical modeling, IVPs have a special significance. The term IVP is used to describe a problem defined by differential equations as well as one or more initial conditions of the unknown function. This emphasis on defining both an initial condition and the derivative of the unknown function sets this type of problem apart from other differential equations. This dual requirement of IVPs allows them to be used to represent dynamic processes, evolution, and change across time. From the original condition, the system's behavior both backward and forward in time can be predicted. IVPs are an essential tool in a variety of fields, including physics, because of their predictive capability, which is particularly important in situations when knowing a system's past state or future behavior is critical.

To formally define an IVP, a differential equation and an initial condition are needed. First, let the differential equation be defined as:

$$\frac{d}{dt}y(t) = \mathcal{N}(t, y(t)) \tag{2.1}$$

A solution to the initial value problem is a function that solves the differential equation, while also satisfying the initial condition. That is, it satisfies:

$$y(t_0) = y_0 \tag{2.2}$$

While some such problems have analytical solutions readily available, many of the systems described by IVPs are complex with solutions that are difficult to obtain. In such cases, numerical approximations are useful.

2.1.2 Numerical Methods

Solving initial value problems using numerical methods is a common approach in cases where obtaining the analytical solution is challenging or impractical [3]. Numerical methods find an approximation of the solution of the differential equation by discretizing the domain and approximating the derivatives [11]. Several different approaches exist, some of the more commonly known are Euler's method and the Runge-Kutta methods.

2.1.2.1 Euler's Method

Euler's method involves dividing the domain of the equation into small intervals and approximating the function's values at each interval using the derivative and the initial condition. The formal formulation of this is given in Eq. 2.3.

$$y_{n+1} = y_n + h * \mathcal{N}(t_n, y_n) \tag{2.3}$$

Here y_n and t_n represent the function value and the corresponding time at time-step n, h is the step-length used for calculation, $\mathcal{N}$ represents the differential equation, and finally y_{n+1} is the function output at timestep $n+1$. Euler's method provides a straightforward way to compute an approximate solution of a differential equation. However, due to its simplicity, it is prone to introducing some errors.

2.1.2.2 Runge-Kutta Methods

The Runge-Kutta methods are a family of methods for numerical approximation that include Euler's method. However, it also consists of methods utilizing higher-order approximations to improve accuracy. These methods iteratively compute the values of the function at successive points in time, updating the approximation based on the derivative at each step.

Runge-Kutta 4 (RK4) is a fourth-order model within the Runge-Kutta family. It is widely used in applications such as fluid mechanics [9] and heat transfer problems [2], and it is defined by the following equations:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.4a}$$

$$t_{n+1} = t_n + h \tag{2.4b}$$

where k_1, k_2, k_3, and k_4 are defined as:

$$k_1 = \mathcal{N}(t_n, y_n) \tag{2.5a}$$

$$k_2 = \mathcal{N}(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}) \tag{2.5b}$$

$$k_3 = \mathcal{N}(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}) \tag{2.5c}$$

$$k_4 = \mathcal{N}(t_n + h, y_n + hk_3) \tag{2.5d}$$

Here, h is the step size used, t_n and y_n are the time and function value at time step n, and t_{n+1} and y_{n+1} are the time and function value at time step $n + 1$.

By carefully selecting the method and appropriate step size, numerical methods allow one to obtain reasonably accurate approximations for a wide range of initial value problems, making them valuable tools in practical applications and numerical simulations. This enables them to act as a trustworthy solution for a variety of differential equations, thus creating a baseline against which other approximations can be evaluated and verified.

2.2 Neural Networks

Neural networks form the base of deep learning, a subfield of machine learning where such multi-layered networks are used to learn intricate patterns and features from data. The algorithms used in neural networks are inspired by the workings of the human brain. Neural networks are composed of a series of connected layers, where each layer contains a collection of neurons. These networks take data as input and are trained, using a training algorithm, to recognize patterns within the data that will be used for future predictions. Figure 2.1 shows the structure of a simple neural network.

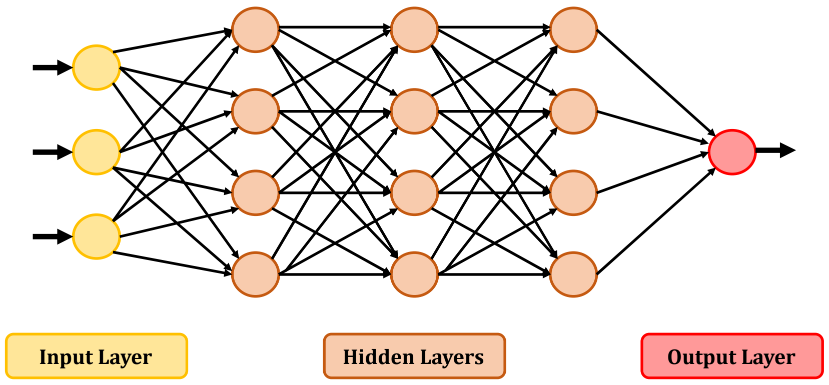


Figure 2.1: This figure is taken from [51]. It shows an illustration of the structure of a neural network. The network has an input layer, colored yellow, that takes three inputs. It then has three hidden layers, each comprising four neurons. The network output layer, shown in red, gives one single output.

There is a wide range of applications for neural networks. This includes using neural networks for solving differential equations. In fact, in recent years several different methods have been proposed. According to Blechschmidt and Ernst in [32] three common ways of doing this include methods based on the Feynman-Kac formula, methods based on the solution of backward stochastic differential equations, and physics-informed neural networks. Neural networks are particularly well-suited for solving differential equations, as they have a large capability to learn and represent intricate relationships from data, and thereby have the capacity to capture complex dynamics and behaviors in a variety of different systems.

2.2.1 Universal approximators

An important characteristic inherent to neural networks is their ability to approximate any measurable function with a desired level of precision, as established by the universal approximation theorem [5]. This even applies to a feedforward neural network with just one single hidden layer. This capability implies that neural networks have the potential to solve complex problems such as IVPs, in a similar manner as conventional numerical solvers are capable of. Thus, neural networks can be readily applied for the modeling of dynamical systems.

2.2.2 Neural network architecture

Layers are the building blocks of the networks, and they shape the capacity of the network to process and learn from data. There are three main types of layers in a neural network: input layer, hidden layers, and output layer. The input layer is the first layer, and it is the one that receives the initial data inputted to the network. This layer transmits this data to the subsequent layers for processing. The hidden layers are not accessible outside the neural network. These layers are the ones enabling the networks to recognize patterns, extract features from the data, and capture the relationships within the data. The final layer, or the output layer, is the layer that produces the prediction or classification result of the neural network. This is done based on the information processed through all of the preceding layers.

Each layer in a neural network is built up of neurons. Figure 2.2 shows the inner workings of a neuron. Every neuron has one or more inputs from the previous layer of the network, or in the case of the input layer, the inputs are the inputs of the network. Each of the inputs to the neuron is multiplied by their own unique weight. The weights are determined through the training process of the neural network, where they are updated for each training step. The weights signify the contribution, or importance, of the associated variable. The weighted inputs are then summarized, together with the bias. The bias, is there to shift the output of the activation function, similar to a constant term in a linear function. The result of the summation between the weighted inputs and the bias is fed into the activation function, which determines the output of the neuron.

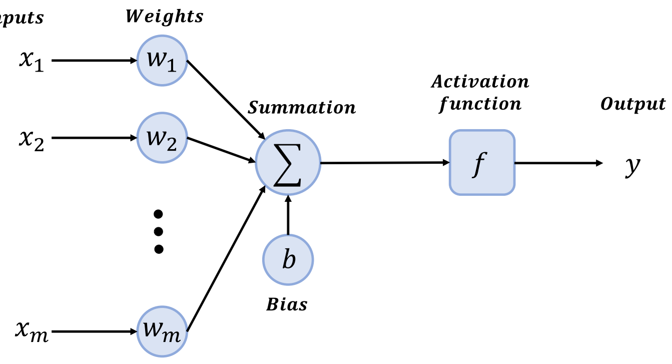


Figure 2.2: This figure shows the structure of a single neuron in a neural network. For a given neuron, there is a set of inputs, where each has an associated weight. The weighted inputs are summed together, and the bias is added. Then, the sum is fed as input to the activation function. The output of the activation function is the output of the neuron.

The activation function determines whether or not a given neuron will be activated. There are numerous possible activation functions to choose from. Figure 2.3 shows three common choices, the sigmoid function, the hyperbolic tangent (tanh), and the Rectified Linear Unit (ReLU). Which activation function to choose depends on a variety of factors, including the problem at hand, the data, and the desired behavior of the neural network. However, within the hidden layers, it is crucial that the activation function used is non-linear. This way, the activation function of the network introduces non-linearity, thus ensuring that the neural network's output isn't a mere linear combination of its inputs. This non-linear transformation enhances the network's capacity to learn complex relationships within the data.

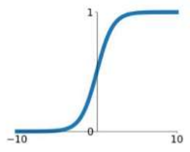
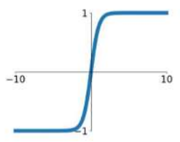
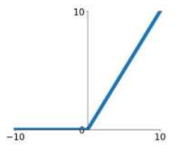
Sigmoid	***Tanh***	***ReLU***
$g(x) = \frac{1}{1+e^{-x}}$	$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$g(x) = max(0, x)$
		

Figure 2.3: This figure illustrates three commonly used activation functions, that is sigmoid, tanh, and ReLU.

2.2.3 Training Neural Networks

The training of a neural network is a process that consists of iterative optimizations aimed at minimizing a predefined loss function. This loss function serves an important role in the training process of the network, as it is what is used to quantify the deviation between the predictions of the network and the actual target values. This error measure is fundamental in ensuring the network's performance improves because, during the training process, the weights of the network are adjusted to minimize this loss. For each iteration of the training process, an optimization algorithm is used to find a direction in which the parameters can be refined to reduce the value of the loss function. As the optimization process continues, the network's predictions tend to align better and better with the target outputs. This ultimately leads the network to become capable of making accurate predictions or classifications, even for new and previously unseen data. Figure 2.4 illustrates the training process of a neural network.

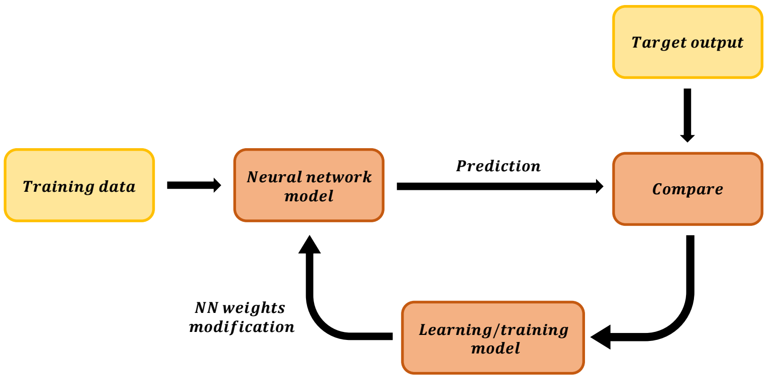


Figure 2.4: This schema shows the training process of a neural network. The training data is inputted into the neural network to get out a prediction. The prediction is then compared to the target output, and the result of this is used for the training algorithm to determine how the weights of the network will be updated.

As the updates applied at each training step are determined by the optimizer of the network, the choice of optimizer directly impacts the accuracy of a neural network [25]. Several different optimizers exist and are commonly used for neural networks, including Stochastic Gradient Descent (SGD) [1], ADAM [12] and Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [6]. Which optimizer is best suited depends on several factors, including the task at hand and the available resources. This is a complex topic with many nuances that falls beyond the scope of this thesis, however, further details can be found in papers such as [7], and [26], as well as in the papers presenting the optimizers ([1], [12], [6]).

2.3 PINN

PINN, as introduced by Raissi et al. in [24], is a novel technique for combining physics-based modeling and machine learning. It is referred to by various names: Scientific Machine Learning [22], universal differential equations [41], physics-informed machine learning [37], theory-guided data science [18], and physics-based deep learning [48]. PINNs present an important change in the approach to complex systems, with their ability to utilize both the intrinsic knowledge from the physics of a system and machine learning's data-driven capabilities.

These neural networks can be used to find the solution to a wide range of dynamic systems, utilizing the underlying physical principles of these systems. This knowledge is incorporated directly into the training process, by integrating the governing equations or constraints that define the behavior of the system into the loss function of the neural network. This ensures that the trained PINN is adapted to the physics of the system, which in turn guarantees that its predictions not only align with the given data but also with the established laws that govern the physical process. This not only enables the PINN to make informed predictions even when the data is sparse or noisy, it also helps ensure the network is capable of generalizing well beyond the training data.

Similar to conventional neural networks, PINNs are universal approximators capable of accurately modeling a wide array of complex problems. As shown already in the original publication [24], the architecture of these networks makes them capable of solving both problems involving forward approximation of functions and problems related to inverse discovery. When used for forward approximation, the PINN accurately predicts outputs given a system, making it a useful predictive tool. In the context of inverse problems, the PINN is able to quickly deduce the input parameters or conditions that correspond to a desired output. This enables the usage of these networks for parameter estimation and system identification as well. This demonstrates the PINN's versatility and utility for a variety of problems.

In many problems involving forward approximation, the functions one wants to approximate are extremely complex. As a result of this, a large number of measurements are frequently required for a neural network to properly be able to reconstruct the function. However, in practice, and particularly when dealing with real-world industrial systems, the access to data is often limited. Furthermore, the data available may consist of incomplete or noisy measurements. The PINN incorporates the governing physical equations of the system into the loss, and as a result, the network is able to train well and deliver accurate predictions for complex systems, even in circumstances with sparse and noisy data.

2.3.1 Mathematical formulation

Define the dynamical process the PINN is intended to solve as $y(t)$. Then the output of the network will be on the following form:

$$y(t) = f_\theta(t) \tag{2.6}$$

where $t \in [0, T]$ is a temporal input. The function f_θ is the mapping obtained by the neural network, where θ denotes the weights of the network.

Consider a nonlinear ODE on the form, where $\mathcal{N}[\cdot]$ represents a general differential operator:

$$\delta_t y(t) + \mathcal{N}[y(t)] = 0, \quad t \in [0, T] \tag{2.7}$$

Then the residual of the differential equation, $\mathcal{F}$, can be defined as:

$$\mathcal{F}(y) := \delta_t y + \mathcal{N}[y] = 0 \tag{2.8}$$

Furthermore, define the boundary conditions and the initial conditions, respectively, as:

$$\mathcal{B}[y(t)] = g(t), \quad t \in [0, T] \tag{2.9a}$$
$$y(0) = y_0 \tag{2.9b}$$

Here, $\mathcal{B}[\cdot]$ is the boundary operator, g represents the boundary of the system, and y_0 represents the initial value of y.

A generic formulation of the loss function for a PINN is given below in Eq. 2.10. The loss function of a PINN is in fact a combination of several loss functions. In general, it consists of at least two terms, $\mathcal{L}_{Data}$ representing the data of the system, and $\mathcal{L}_{Physics}$ which is the loss derived from the known physical dynamics of the system. Figure 2.5 illustrates the workings of a PINN. The loss function in Eq. 2.10 also includes two weighting factors, λ_{Data} and $\lambda_{Physics}$ that allow for adjusting the relative importance, or contribution, to the overall result, from the loss originating from the data, and the loss related to the physical dynamics respectfully.

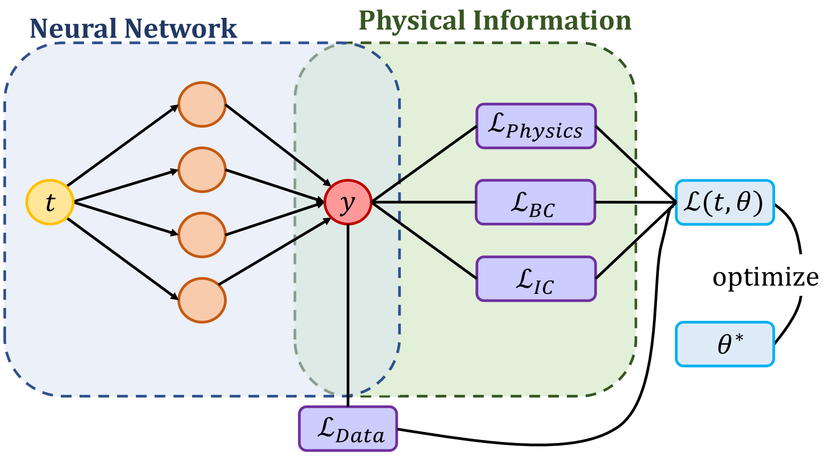


Figure 2.5: This figure illustrates the workings of a simple physics-informed neural network, with t as input, and y as output. It is a modified version of a figure from the prework for this thesis, [51].

$$\mathcal{L} = \lambda_{Data} * \mathcal{L}_{Data} + \lambda_{Physics} * \mathcal{L}_{Physics} \tag{2.10}$$

The losses, $\mathcal{L}_{Data}$ and $\mathcal{L}_{Physics}$ are defined as below in Eq. 2.11.

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left| y(t_d^j) - \hat{y}^j \right|^2 \tag{2.11a}$$

$$\mathcal{L}_{Physics} = \frac{1}{N_F} \sum_{k=1}^{N_F} \left| \mathcal{F}(y(t_f^k)) \right|^2 \tag{2.11b}$$

Here, N_D represents the number of data points, where the data value of the system is known. These points are denoted as (t_d^j), with $y(t_d^j)$ being the output of the neural network at such a given data point and $\hat{y}^j$ is the corresponding known data value. N_F is the number of collocation points, where the residual will be evaluated.

This basic formulation can be extended to include other known physical conditions, like the boundary condition and initial condition. The associated losses can be defined as follows:

$$\mathcal{L}_{BC} = \frac{1}{N_B} \sum_{l=1}^{N_B} \left| \mathcal{B}[y(t_b^l)] - g(t_b^l) \right|^2 \tag{2.12a}$$

$$\mathcal{L}_{IC} = \frac{1}{N_I} \sum_{m=1}^{N_I} \left| y(0) - y_0^i \right|^2 \tag{2.12b}$$

where $\mathcal{L}_{BC}$ represents the loss associated with the boundary condition, and $\mathcal{L}_{IC}$ that of the initial condition.

The complete loss function might then look like given below in Eq. 2.13, where the terms λ_{BC} and λ_{IC} represent the weighting factor of the boundary condition and the initial condition respectfully.

$$\mathcal{L} = \lambda_{Data} * \mathcal{L}_{Data} + \lambda_{PDE} * \mathcal{L}_{PDE} + \lambda_{BC} * \mathcal{L}_{BC} + \lambda_{IC} * \mathcal{L}_{IC} \tag{2.13}$$

2.3.2 Training PINNs

In practice, the training of a PINN can be seen as working in a number of passes, one pass for each term of the loss function. The minimum formulation of the PINN consists of two terms, the data term and the physics term. For this network, the training works in two passes. The first pass is related to the data available for the system. Assume there is access to measurements from the system at N_D distinct spatial locations in the input domain. These coordinates are then passed as input to the neural network, with the goal of minimizing the error between the obtained predictions and the data available at the points. This is similar to the training process of a regular neural network, which is explained in further detail in Section 2.2.3.

The second part of the training process for the PINN, is the pass for the physics term of the loss function. For this, a point cloud is generated for the N_F points at which the physics of the system will be evaluated. Typically, N_F will be significantly larger than N_D, that is $N_F >> N_D$. For each of these N_F points, the gradients are calculated using automatic differentiation, and the loss is calculated based on how well the results aligns with the governing equations. From here, the gradients are backpropagated, similar to what happens for the data loss.

The originally proposed PINN was trained using the quasi-Newton method L-BFGS, which is a Limited-Memory BFGS optimization algorithm. While the L-BFGS works well in the sense that it provides high accuracy, it is also prone to potentially getting stuck in local minima [39]. Several studies have been conducted in order to determine the best approach for training physics-informed neural networks. The ADAM optimizer has become a popular alternative, either used alone as in [33] or, perhaps more commonly, in combination with L-BFGS as done in [39].

2.3.3 Proposed improvements to the original PINN

The original architecture of the PINN works well in simple issues, but it fails in more complicated circumstances where the PDEs are more complex [52]. However, these networks come with great potential, thus several propositions have been made to remediate some of the limitations of the original architecture.

One way to improve the performance of the PINN, is to change the weighting of the terms in the loss function. In the standard PINN formulation, the terms are all weighted equally, which does not account for the possible differences in the gradient contribution. One way to handle this would be to assign fixed weights to each term beforehand. However, this provides little flexibility and also requires tuning of these hyperparameters for every new problem. Another possible solution is the use of dynamic weighting, where the weights of the loss terms are adapted in each epoch such as done in [50].

Efforts have also been made to improve the activation function used in the network. In the original architecture, the hyperbolic tangent is used. While this activation function, in general, produces good results, it has been observed that when used to solve differential equations, it might fail to pick up higher frequencies in the solution, or these higher frequency components appear late in the training [49]. Some of the possible proposed solutions to this include an adaptive activation function using a hyperparameter with a hyperbolic tangent or ReLU [27] and using a siren activation [53].

Another way to improve the overall performance of the PINN, is to change the way the choice of collocation points are generated. In the original formulation of the PINN, these points were chosen randomly or uniformly. This might cause the network to train inefficiently, as some parts of the function might be less complex and thus require less density of collocation points than others. To improve on this, residual-based adaptive refinement has been proposed in [38]. This technique evaluates the distribution of the physics-based loss and increases the density of collocation points in the regions where this loss is high.

2.4 PINC

PINC were introduced by Antonelo et al., in [43]. These physics-informed neural networks are an extension of the original PINN formulation that leverages the power of neural networks to not only predict system behavior but also to be able to seamlessly integrate with real-time controllers for dynamic processes. While the original PINN formulation is very versatile, it has two key issues that need to be addressed for it to be usable in a control application. The first problem is that the network needs to be adapted to include the control action and the initial condition of the modeled system. Secondly, it has to be possible to use the network for long-range simulations.

In order to be able to model the behavior of a dynamical system in a variety of states, it is crucial for the PINC to include the initial condition of the system and the control action as parts of the input. By doing so, the network can be trained for a variety of starting states, and thus predict the behavior of the system for a broad range of control settings. A simple schema illustrating the PINC with extended input is shown in Figure 2.6. It should be noted that the control action needs to remain constant during the time horizon simulated by the network.

The PINC suffers from an increased deviation between the predicted and actual solutions as more time elapses from the initial starting time. However, as the PINC can be trained for a variety of different initial conditions, this problem can be mitigated by chaining a PINC trained for a shorter time horizon. The concept of long-range simulations for the PINC is further detailed in Section 2.4.3

2.4.1 Mathematical formulation

The mathematical formulation of the PINC is similar to that of the original PINN, however, as previously mentioned, the input is extended to include the initial condition of the system and the applied control input, in order for the network to be capable of being used in correlation with a controller. This leads to the output of the network being in the following form:

$$y(t) = f_\theta(t, y(0), u), \quad t \in [0, T] \tag{2.14}$$

where $t \in [0, T]$ is a temporal input, $y(0)$ is the initial condition of the system, u is the control input, and f_θ represents the mapping obtained by the neural network, with θ denoting the weights of this mapping.

The loss function remains the same as for the PINN, given by Eq. 2.10. However, the losses need to be adjusted to take into account the added inputs of the network. The expressions for the terms of the loss functions, therefore, have to be updated, and $\mathcal{L}_{Data}$ and $\mathcal{L}_{Physics}$ are given as the following.

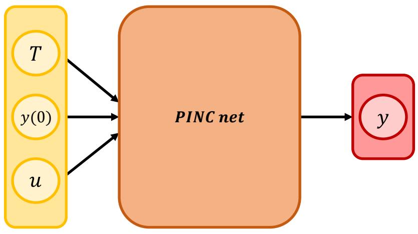


Figure 2.6: An illustration, taken from [51], of a PINC net, showing the input consisting of time, initial condition and control input.

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left| y(t_d^j, y(0)_d^j, u_d^j) - \hat{y}^j \right|^2 \tag{2.15a}$$

$$\mathcal{L}_{Physics} = \frac{1}{N_F} \sum_{k=1}^{N_F} \left| \mathcal{F}(y(t_f^k, y(0)_f^k, u_f^k)) \right|^2 \tag{2.15b}$$

where $(t_d^j, y(0)_d^j, u_d^j)$ are the data points, N_D is the number of such points, and $\hat{y}^j$ represents the known data values. The number of collocation points is N_F, with these points being denoted as $(t_f^k, y(0)_f^k, u_f^k)$.

2.4.2 Training PINCs

The training of the PINC is in many ways similar to that of the PINN, the main difference lies in the provided training data. In order to ensure good overall performance for the PINC in a variety of conditions, it is necessary for the network to train for a large set of possible combinations of initial conditions and control inputs within the feasible region of the system. This region is the set of all possible values of initial conditions and control inputs that the system can take without becoming unstable. An important step in the training process of the PINC is to determine the feasible region for which the network shall be trained.

As the PINC is trained for a number of possible initial conditions and control inputs, whereas these are embedded directly in the inner equations of the corresponding PINC, it follows that the process of training the PINC can be more challenging. In fact, while the PINN learns to predict a single function, the PINC on the other hand learns an array of such functions. This leads to a need for more training to achieve a PINC performing to the same level as a PINN.

2.4.3 Long-range simulations

The performance of the PINC, much like the PINN, degrades rapidly as the time horizon of the network increases [43]. Yet, as the network can be trained to be used for a variety of different initial conditions and control inputs, and the initial condition is in fact part of the inputs to the network, it can still be used to obtain simulations over longer time intervals. This can be achieved by having a chain of networks, each predicting the solution for one single time horizon.

Due to the nature of the PINC and its ability to learn a variety of conditions, the networks of the chain can in fact all be the same model. Thus, this can be described as using the network in self-loop mode, where the output of the model at time step k, $y[k]$, will be used as the initial condition for the model at time step $k+1$:

$$y[k+1] = f_\theta(t_k, y[k], u[k]) \tag{2.16}$$

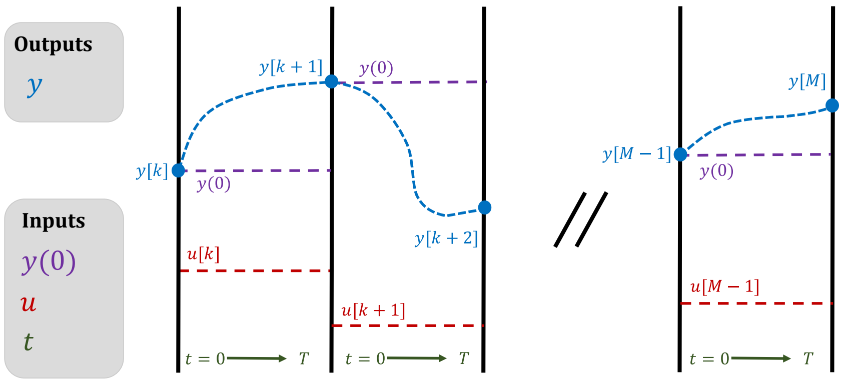


Figure 2.7: This figure illustrates the concept of using the PINC in long-range simulations. For each time interval, a control action is decided. The initial condition is the final state of the previous time interval.

Figure 2.7. illustrates the development of the inputs and outputs of a PINC in self-loop mode over time. As the same network is used in a self-loop mode, for each iteration the time is reset to zero. The control input changes at each time interval, and it is determined by a controller outside of the network, for instance using an MPC controller. The initial condition of the network at each time step will be the end state from the output of the previous network.

Chapter 3

Electric Submersible Pump

An Electric Submersible Pump, or ESP for short, is a type of pump that can be installed in oil wells. The purpose of the pump is to enable or boost production in wells installed in reservoirs where, for some reason, the oil cannot be lifted up by the reservoir pressure alone. This pump system was first proposed in the 1910s by Armais Arutunoff [20]. With this long history, it has found many application areas, such as waterflood operations and offshore production. In general, the system could be applied in any case where large volumes need to be lifted and there is electricity available.

As [20] states, the conventional installation of an ESP system depicted in Figure 3.1, remains a common setup to this day. It is characterized by a few key features. The first is that there's only liquid entering the centrifugal pump of the system. The second is that the viscosity of the produced liquid is of low enough viscosity. And finally, the third is that the motor of the ESP is supplied with an AC current of constant frequency.

The electric submersible pump remains a popular choice party due to its many advantages. The pump system is able to produce high liquid volumes, with relatively high efficiency and low maintenance. Furthermore, it is well suited for use in various locations, ranging from urban environments to offshore installations. It is also suited for use in deviated wells.

However, the ESP also has a set of disadvantages to take into consideration. For instance, the installation requires a reliable power source to operate. When using an ESP one must also take into consideration the amount of gas and materials like sand that are present. The first may require action to prevent it from reducing efficiency, while the latter may wear on the equipment. Some of the other factors are related to the flexibility of the system, the cost of repairs, and the operating conditions.

Due to their extensive use, it is essential to be able to operate ESPs in a manner that limits the failures of the system, while maintaining optimal production. Thus, the manual operation of these wells for optimal production is a challenging task. Therefore, automation solutions for the control of ESPs are essential, and several have already been proposed and tested. These include conventional safety systems, PID controllers for control of the intake pressure or current, and more recently, an MPC approach for more advanced automatic control [13]. For the development of such control solutions, a dynamic model of the system is needed. The following section will introduce the model of an ESP in more detail.

3.1 Modeling of wells with ESP

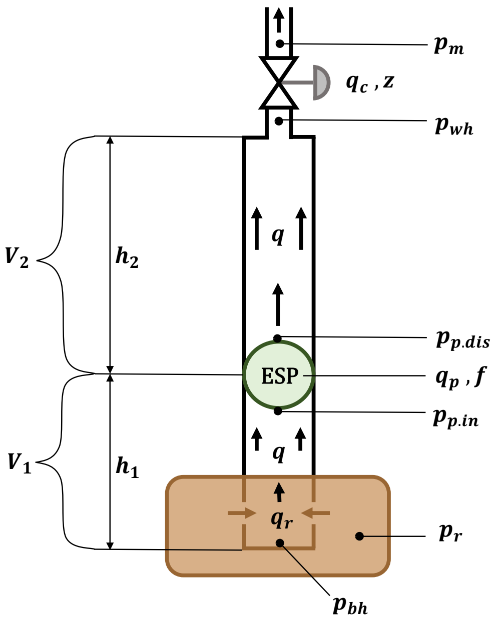


Figure 3.1: Model of an ESP lifted well, recreated based on [13] and [14].

3.1.1 Model equations and parameters

This model is based on the model of an ESP-lifted well presented in [13], and details of the model derivation and limitations can be found there. The equations and parameters of the model will be presented.

The system can be described by the following ordinary differential equations (ODEs):

$$\dot{p}_{bh} = \frac{\beta}{V_1}(q_r - q) \tag{3.1a}$$
$$\dot{p}_{wh} = \frac{\beta}{V_2}(q - q_c) \tag{3.1b}$$
$$\dot{q} = \frac{1}{M}(p_{bh} - p_{wh} - \rho g h_w - \Delta p_f + \Delta p_p) \tag{3.1c}$$

The parameters of these ODEs are defined by the following algebraic equations:

Flow:

$$q_r = PI(p_r - p_{bh}) \tag{3.2a}$$
$$q_c = C_c\sqrt{p_{wh} - p_m}z \tag{3.2b}$$

Friction:

$$\Delta p_f = F_1 + F_2 \tag{3.3a}$$
$$F_i = 0.158 \cdot \frac{\rho L_i q^2}{DA^2} \cdot \left(\frac{\mu}{\rho D q}\right)^{\frac{1}{4}} \tag{3.3b}$$

ESP:

$$\Delta p_p = \rho g H \tag{3.4a}$$
$$H = C_H(\mu)H_0(q_0)\left(\frac{f}{f_0}\right)^2 \tag{3.4b}$$
$$q_0 = \frac{q}{C_q(\mu)}\left(\frac{f_0}{f}\right) \tag{3.4c}$$

Table 3.1 gives an overview of the different model variables used in the model.

Table 3.1: ESP model variables and characteristics

Control inputs	
f	ESP frequency
z	Choke valve opening

Pressures	
p_m	Manifold pressure
p_{hh}	Wellhead pressure
p_{bh}	Bottomhole pressure
$p_{p,in}$	ESP intake pressure
$p_{p,dis}$	ESP discharge pressure
p_r	Reservoir pressure

Flow rates	
q	Average well flow rate
q_r	Reservoir-to-well flow rate
q_c	Flow rate of production choke

ESP characteristics	
H_0	ESP head characteristics
q_0	Theoretical flow rate at reference frequency
C_H	VCF for head
C_q	VCF for ESP flow rate

The parameters of the model are defined as given in Table 3.3, while different constants of the model are given in Table 3.2. The values used here are based upon those given in [14]. The variables H_0, C_H and C_q are polynomial, with coefficients defined as given in Table 3.4.

Table 3.2: Well dimensions, ESP characteristics, and other constants

Symbol	Meaning	Value	Unit
g	Gravitational acceleration constant	9.81	m/s^2
C_c	Choke valve constant	$2 \cdot 10^{-5}$	*
A	Cross-section area of pipe	0.008107	m^2
D	Pipe diameter	0.1016	m
h_1	Height from reservoir to ESP	200	m
h_w	Total vertical distance in well	1000	m
L_1	Length from reservoir to ESP	500	m
L_2	Length from ESP to choke	1200	m
V_1	Pipe volume below ESP	4.054	m^3
V_2	Pipe volume above ESP	9.729	m^3
f_0	ESP characteristics reference freq.	60	Hz

* Appropriate SI unit

Table 3.3: Parameters from fluid analysis and well tests, and parameters assumed to be constants

Symbol	Meaning	Value	Unit
β	Bulk modulus	$1.5 \cdot 10^9$	Pa
M	Fluid inertia parameter	$1.992 \cdot 10^8$	kg/m^4
ρ	Density of produced fluid	950	kg/m^3
P_r	Reservoir pressure	$1.26 \cdot 10^7$	Pa
PI	Well productivity index	$2.32 \cdot 10^{-9}$	$m^3/s/Pa$
μ	Viscosity of produced fluid	0.025	$Pa \cdot s$
P_m	Manifold pressure	20	Pa

Table 3.4: Polynomial coefficients

	c_0	c_1	c_2	c_3	c_4
H_0	9.5970e2	7.4959e3	-1.2454e6	0	0
C_H	1	-0.03	0	0	0
C_q	1	-2.6266	6.0032	-6.8104	2.7944

3.2 Implementation

The implementation of the simulator was done using Python version 3.8 and CasADI (Computer algebra system for Automatic Differentiation) version 3.5.5. CasADI was used as a tool to solve the Differential-Algebraic Equation (DAE). The integrator provided in CasADI was used for this purpose. Plots of the simulation results were created using matplotlib. Complete code for the simulator can be found at `https://github.com/auroraslb/Prosjektoppgave`.

3.3 Verification

The verification of the simulator is necessary before it can be used as an approximation for a real ESP. In order to verify the simulator results, the output of the simulator for different scenarios is compared with the system equations and the expected behavior. The simulator can be considered verified if the resulting output matches these. Four scenarios are considered to see how the simulator responds to constant input, a square pulse in the valve opening, a square pulse in the frequency input, and random changes in the valve opening respectively.

3.3.1 Scenario 1: f = 60Hz and z = 100%

In the first scenario tested, both the frequency and valve opening is held constant throughout the simulation. According to the model outlined in the previous section, such input should lead to the stabilization of all three response signals (q, p_{wh}, and p_{bh}). And indeed, as can be seen from Figure 3.2 this is what is happening with the output from the simulator.

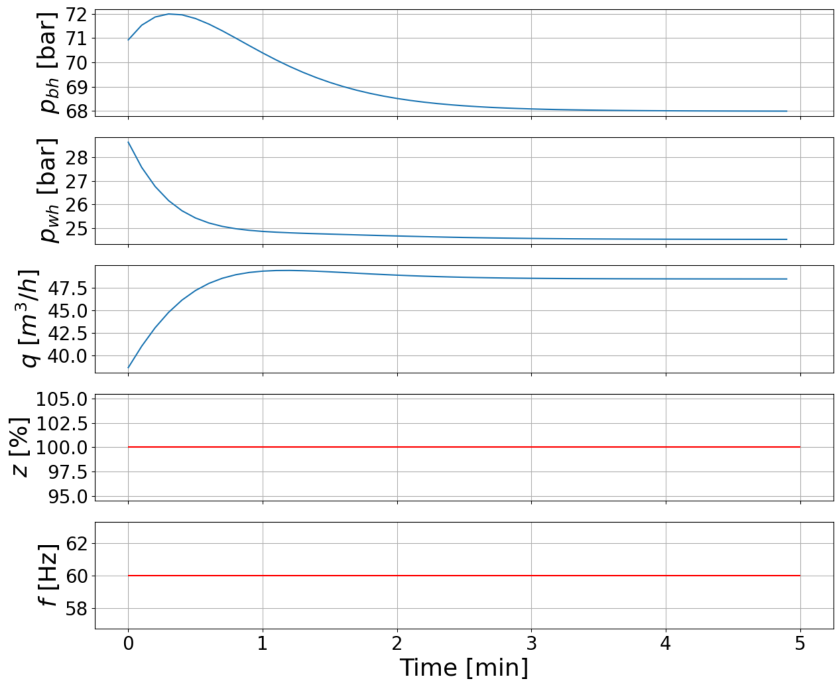


Figure 3.2: The simulation output with constant input of f = 60 Hz and z = 100%.

3.3.2 Scenario 2: f = 60Hz and z steps from 60% to 100% and back

In Figure 3.3, one can see that the outputs initially are stabilizing themselves. Then, as the valve opening z is increased, so is the liquid flow q. The pressures both decrease. This matches the intuitive behavior of the system well, as a larger valve opening gives opens up for a larger flow, while an increased flow leads to lower pressure. It also fits well with the theoretical model. From equation 3.2b there is a negative relationship between z and $\dot{p_{wh}}$, so an increase in z leads to a decrease in p_{wh}. Furthermore, from 3.1c it follows that as p_{wh} decreases, $\dot{q}$ and q increases. Finally, from 3.1a, as q increases, $\dot{p_{bh}}$ and p_{bh} decreases. When the valve opening is decreased again, the opposite applies.

3.3.3 Scenario 3: z = 60% and f steps from 60Hz to 70Hz and back

In this scenario, the response of the system when the valve opening is held constant, but the frequency is increased and then decreased is simulated. Figure 3.4 shows the response of the system in this scenario. As can be seen from the figure, increasing

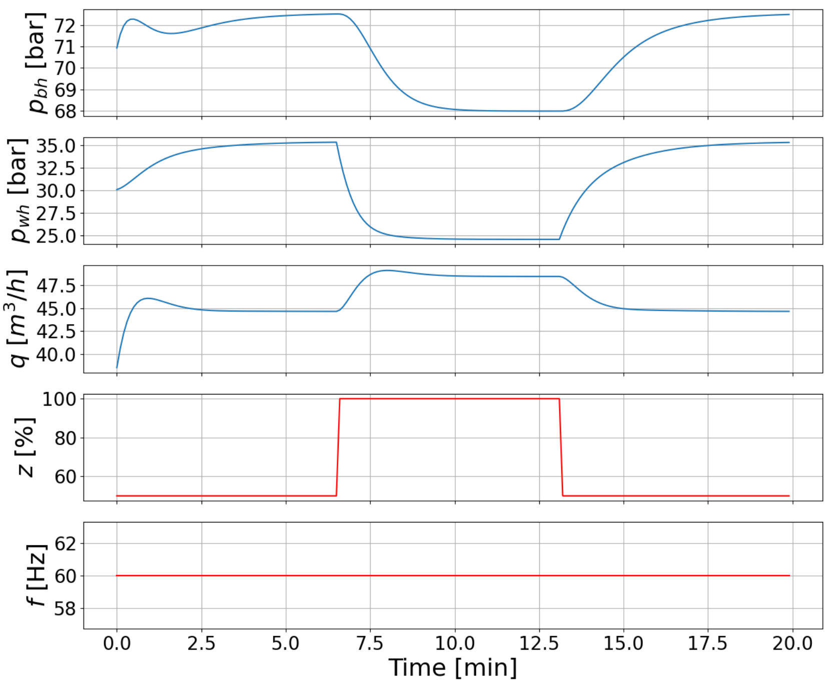


Figure 3.3: The simulation output when the valve opening is switched from 60% to 100% after one-third of the time, and back again after two-thirds.

the frequency leads to a decrease in the bottomhole pressure, and an increase in the wellhead pressure and the liquid flow. This matches the expected behavior of the system, as an increase in the frequency increases the speed of the ESP. Thus liquid is lifted faster upwards, resulting in less pressure on the bottom, but an increased flow and increased pressure on the top. It also matches the theoretical model well. Equations 3.4 give a positive relationship between f and $\dot{q}$, and therefore also q. Furthermore, from equation 3.1a we see that an increase in q leads to a decrease in p_{bh}, and from equation 3.1b it leads to an increase in p_{wh}. When the frequency is lowered again, the opposite applies.

3.3.4 Scenario 4: f = 60Hz and z steps randomly between 50% to 100%

In the final verification scenario, the response of the system to random changes in the choke valve opening is examined. Figure 3.5 shows the input and the corresponding response signals. As in the previous scenario, one can observe that increasing the choke valve opening increases the flow while decreasing the opening also decreases the flow. Furthermore, the previously mentioned inverse relation

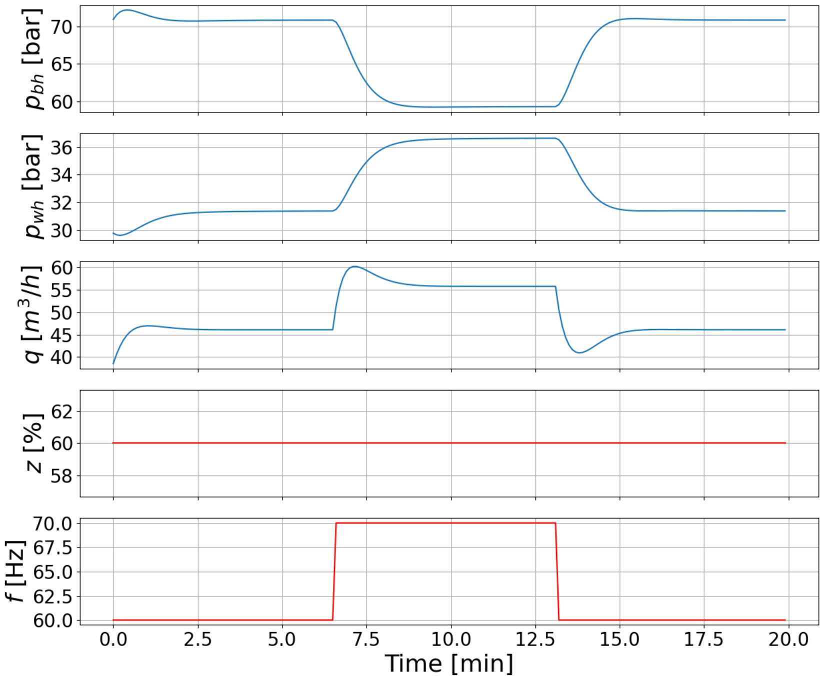


Figure 3.4: The simulation output when the frequency is switched from 60 Hz to 70 Hz after one-third of the time, and back again after two-thirds.

between the flow and the pressures of the system is apparent here as well. The wellhead pressure exhibits a more rapid and pronounced response to changes in the flow rate compared to the bottomhole pressure. This can be due to the fact that the wellhead pressure is directly connected with the choke valve opening.

3.3.5 Results of verification

As the four scenarios treated above all show that the simulation results correspond well with the expected behavior and the theoretical model of the ESP system, thus the simulator can be considered verified.

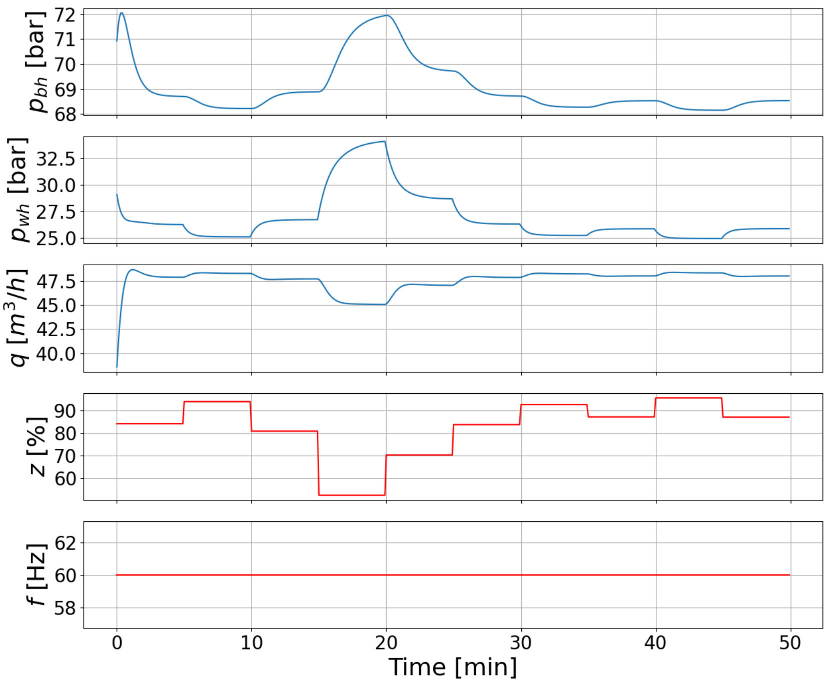


Figure 3.5: The simulation output when the valve opening is randomly changing within the limits of 60% and 100%.

Chapter 4

Physics-Informed Neural Network for the Electric Submersible Pump

This chapter presents the practical aspects of implementing a PINN model for the ESP. This includes detailing the model used and how it was constructed, the training of the model, as well as the process of validating the constructed network.

4.1 Technical Specifications

The PINN model is implemented using Python 3.9 and Pytorch. The training is performed in Google Colaboratory using NVIDIA's T4 GPU and high-RAM setting.

4.2 Construction of the PINN Model

Constructing the PINN for the ESP involves two fundamental components. Firstly, it entails creating the neural network architecture, which serves as the backbone for learning the system's behavior. This involves defining the number and structure of the layers of the network, the activation functions used, and the connectivity patterns between neurons.

For this, a neural network comprising one input layer, one output layer, and eight hidden layers is created. All of the layers are fully-connected, also known as linear, which means that all input neurons are connected to every output neuron. Each hidden layer consists of 50 neurons, and the activation function is the hyperbolic tangent. The inputs of the network are normalized to values between -1 and 1.

The second component of constructing the PINN model, is finding an appropriate loss function that captures the physical dynamics inherent in the system. Such a loss function incorporates the known governing equations of the ESP, presented in

Chapter 3. By minimizing this loss function during the training process, the PINN can effectively learn to model and predict the ESP's behavior while adhering to the underlying physical principles.

The loss function of the PINN for the ESP is based upon the basic formulation given in Eq. 2.10 of the PINN loss function, that is, only two terms are included. One is accounting for the deviation between the solution given by the neural network and the differential equations governing the system, and the other is the known data points. The two terms of the loss function are weighted equally, meaning that λ_{Data} and $\lambda_{Physics}$ in Eq. 2.10 are both set to 1.

4.3 Training the PINN Model

In order to train the neural network, training data has to be gathered. For the PINN for ESP, two sets of training data are necessary. The first is that corresponding to the known data points. The second is the collocation points, where the residual is evaluated to ensure the solution matches the governing physical equations.

The training data corresponding to the data points consists both of the input and the corresponding output value of the system. For the PINN for ESP, the input to the network only consists of time. The output is the bottom hole pressure, wellhead pressure, and the flow through the system. Furthermore, in the case of the ESP, only one data point is known, that is, the initial condition of the system.

The collocation points do not have a corresponding output value, they only consist of the input values. Thus, the collocation points for the PINN for ESP will simply be a set of time values. In order to generate this training data, the number of collocation points was chosen to be $N_f = 10000$, and these were sampled randomly.

The network is trained using 2000 steps, and the ADAM optimizer. Further details on how the training process of the PINN works can be found in Section 2.3.2.

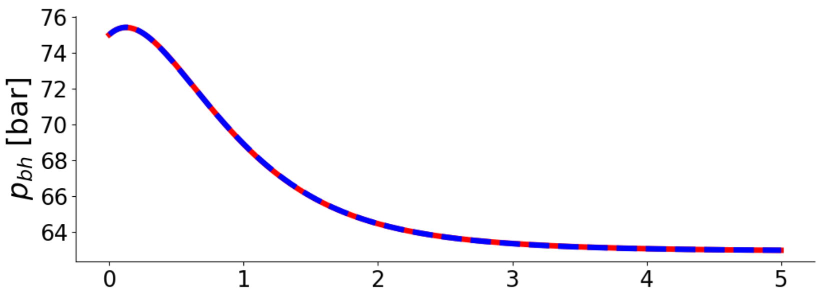
4.4 Validation of the PINN Model

The PINN model of the ESP acts as a solver of the system, finding the solution to the set of differential equations defining it. In order to validate the results of the neural network, the output can be compared to the true solution of the differential equations. For this purpose, the solution obtained using the numerical solver RK4 is considered to be the true solution of the system.

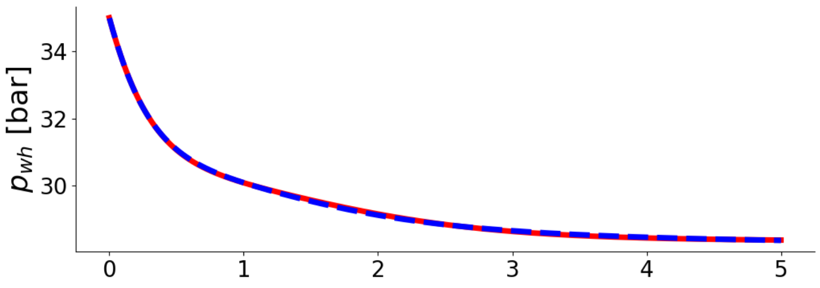
In this section, the neural network is validated using four combinations of control actions. For each scenario, a new network is trained. In all scenarios, the initial condition is the same, $x = [75\text{bar}, 35\text{bar}, 35m^3/h]$, and the time horizon is set to $T = 5\text{min}$. The first scenario is when both control inputs are held high, a high frequency and a large valve opening. The second scenario is when the frequency is held high, but the valve opening is low. The third is when the frequency is low, but the valve opening is large. The final scenario is when both inputs are low. In all scenarios, the solution obtained using PINN is plotted against the RK4 solution.

4.4.1 Scenario 1: f = 65 Hz and z = 80%

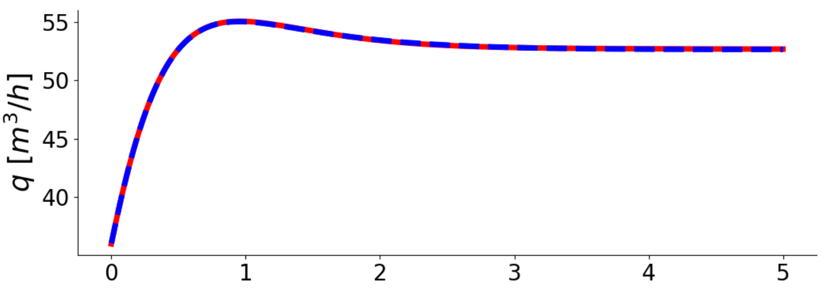
In this scenario, both the control frequency and the valve opening are set to high values. A PINN model was then trained using the procedure described above in Section 4.3. The performance of the resulting model is shown in blue in Figure 4.1, where it is compared to the true solution of the system, obtained using RK4 and plotted in red.



(a)



(b)

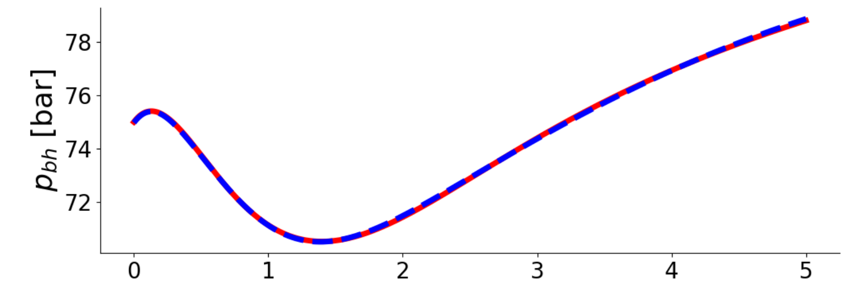


(c)

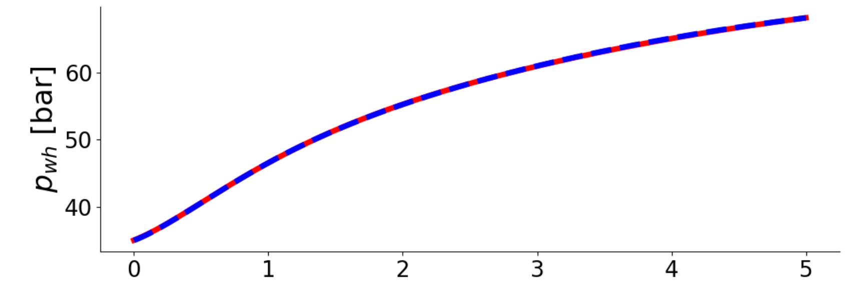
Figure 4.1: This figure shows a PINN (blue) trained with a high value for both the control frequency (65 Hz) and the valve opening (80%). The result is compared to the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

4.4.2 Scenario 2: f = 65 Hz and z = 20%

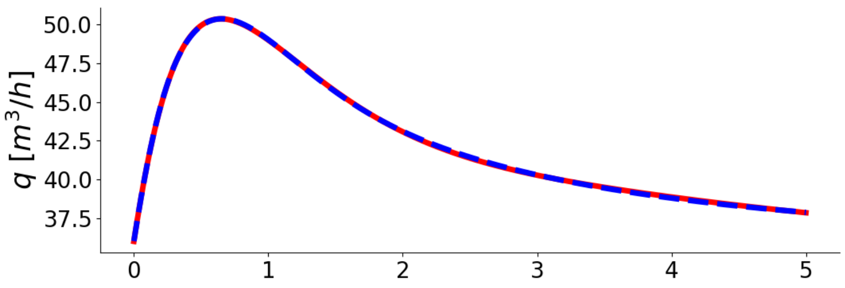
In this scenario, the control frequency is still held high, but the valve opening is set low. Figure 4.2 shows the results of the PINN trained for such a setting (in blue), compared to the solution given by RK4 (in red).



(a)



(b)



(c)

Figure 4.2: This figure shows a PINN (blue) trained with a high value for the control frequency (65 Hz) and a low value for the valve opening (20%). The result is compared to the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

4.4.3 Scenario 3: f = 40 Hz and z = 80%

For the third scenario, the control frequency is set low, while the valve opening is set high. The outcomes of the PINN trained with this setting are depicted in Figure 4.3 in blue, along with the solution obtained from RK4, depicted in red.

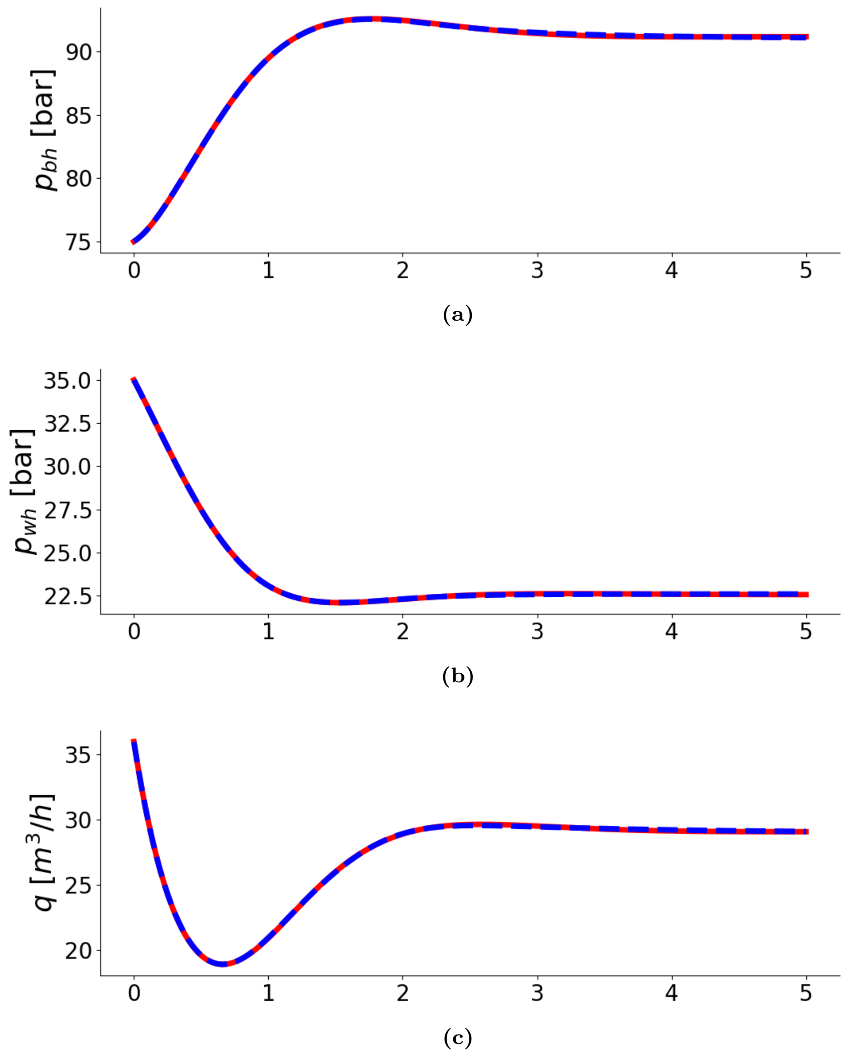


Figure 4.3: This figure shows a PINN (blue) trained with a low value for the control frequency (40 Hz) and a high value for the valve opening (80%). The result is compared to the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

4.4.4 Scenario 4: f = 40 Hz and z = 20%

The final scenario explored is that when both the control frequency and the valve opening are set low. The outputs from a PINC model trained for such conditions are shown in blue in Figure 4.4, and the true solutions of the system obtained using RK4 are shown in red.

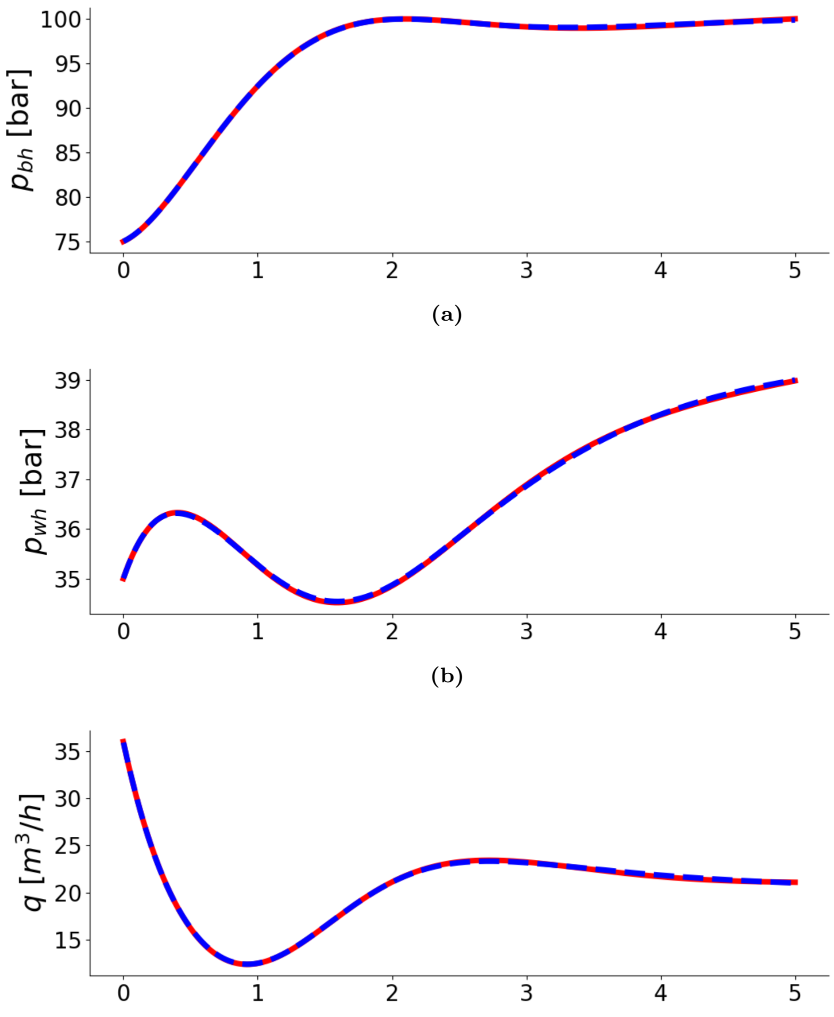


Figure 4.4: This figure shows a PINN (blue) trained with a low value for both the control frequency (40 Hz) and the valve opening (20%). The result is compared to the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

4.4.5 Results of validation

The above scenarios show great overlap between the solution obtained using a PINN and using a traditional numerical solver, the RK4. The scenarios tested different combinations of the control input settings, and the model performed well throughout them all. As the PINN was able to accurately give solutions in the cases of both high and low control frequency, and with high and low valve opening, this implies its capability to provide accurate predictions within the range of tested values as well. Thus, the model can be considered validated.

Chapter 5

Physics-Informed Neural Network for Control of the Electric Submersible Pump

This chapter describes the steps required for implementing the PINC for the ESP, including the practical details on the implemented model, the training of the model, and the method for validating the resulting neural network.

5.1 Technical Specifications

The technical specifications here are the same as for the PINN. The PINC model is implemented using Python 3.9 and Pytorch. The training of the model is performed in Google Colaboratory using NVIDIA's T4 GPU and high-RAM setting.

5.2 Construction of the PINC Model

The PINC model for the ESP is based on the previously developed PINN model for the system. However, in making the transition from a regular PINN into one that is able to work in a control system, some key aspects are changed. The input of the model is extended from just consisting of the time, to also include control inputs and the initial condition of the ESP system. In the case of the ESP, this results in the input comprising the time, the valve opening, and control frequency, as well as the initial condition of the three state variables: the wellhead pressure, the bottomhole pressure, and the flow. Thus, the input of the network is now on the form given below in Eq. 5.1.

$$X = [t, z, f, x_1, x_2, x_3] \tag{5.1}$$

Here, t is the time, z and f are constants representing the control input, and x_1, x_2 and x_3 are the initial conditions of the system.

The network structure comprises one input layer, one output layer, and ten hidden layers. Each layer consists of 50 neurons, and they are all fully-connected. The activation function used is the hyperbolic tangent. The loss function is based upon the basic formulation of the loss function for a PINN given in Eq. 2.10.

5.3 Training the PINC Model

As the PINC needs to be trained for a variety of initial conditions and possible control inputs, training efficiency is of greater importance here than in a given PINN. As with the PINN, the ADAM optimizer is used to train the PINC as well. The number of training steps used to create the different PINC models varies depending on the use case, and is specified for each model.

5.3.1 Training data

In order to properly utilize the PINC in a control setting, it is necessary to train it with a range of feasible initial conditions, and a set of possible control inputs. As with the PINN, two sets of data are necessary to gather to train the PINC. The first corresponds to the known data points, in this case, the initial conditions. The second is the set of collocation points.

In order to sample points for training the PINC model, three parameters need to be defined. That is, firstly, how many points in time the equations will be evaluated. Secondly, the number of possible control inputs, or combinations of z and f, that the network will be trained for. And finally, the different possible initial conditions, consisting of x_1, x_2, and x_3, the network will learn. Let these numbers be defined as N_t, N_c and N_i respectively.

The training data corresponding to the data points will consist of all the possible initial conditions the PINC will be trained with. In this case, the time is held constant, as the initial condition is evaluated at time zero. However, as the input of the neural network consists of both the control input and the initial condition as well, the total number of training points will be the combination of each possible initial condition and each possible control input. This results in a total of $N_D = N_c * N_i$ data points.

The data used for evaluating the physics of the system, the collocation points, need to cover the range of initial conditions as well as the control inputs. This means that for each sampled time point, this has to be combined with each sampled initial condition as well as each sampled control input. This results in a total of $N_F = N_t * N_c * N_i$ collocation points.

5.3.2 Defining a feasible region

For the network to properly train, it is necessary for the points gathered to be within the feasible region of the differential equation. This can be achieved by finding ranges of values for the different states between which the equation produces valid results. Then, this can be used to set minimum and maximum values that the points can be sampled between. In order to define a feasible region for this network, RK4 was used. A variety of different settings for initial condition and control inputs were tested to see whether or not they were valid combinations for the equations.

After conducting these experiments, the feasible region for the initial values was chosen as follows. The bottom-hole pressure, or x_1, was defined to be within 70 to 85 bar. The wellhead pressure, x_2, was set to be in the interval 25 to 40 bar. The flow, x_3 was held between 25 and 50 m^3/h . For the feasible region for the control input, the valve opening was set between 10 and 90%, and the frequency between 40 and 65 Hz.

5.4 Validation of the PINC Model

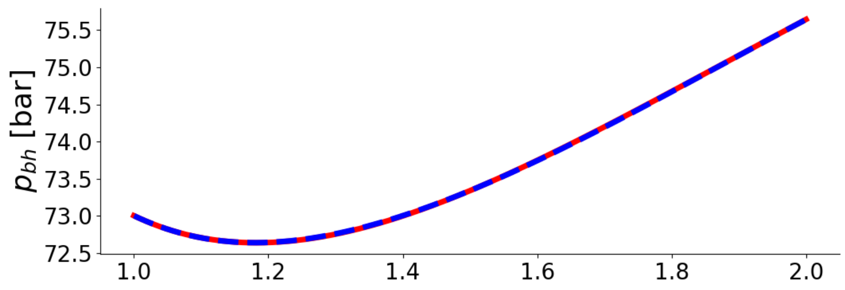
The process of validating the PINC model is somewhat similar to that of the PINN model. That is, in both cases the resulting output is compared to the output of a numerical method, in this case RK4. However, in the case of PINN, there is only one initial condition and one fixed control input to validate. The PINC, on the other hand, will be trained in order to be used in a control model with multiple possible control signals and initial conditions.

Long-range simulations of the PINC are obtained by chaining the network in a self-loop mode. Thus, in order to validate the neural network, one can choose a set of values for the control input as well as a range of initial conditions. Each of these settings can then be validated, in the same way the PINN was validated. If the network is able to accurately predict short-range simulations, it can thus provide accurate long-range simulations.

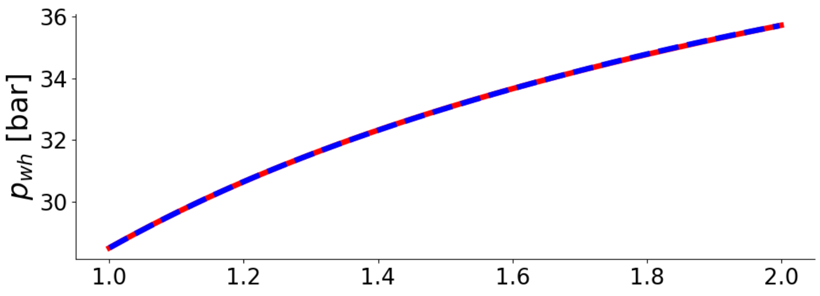
The PINC model is validated in two steps. The first step consists of validating the network's ability to predict the solution of the system given a single initial condition and control input, that is whether it works when used in a similar manner as the PINN. The second step of the validation is training a single network for three sets of initial conditions and control inputs, in order to validate the capability of the PINC to predict the solution of the system given a variety of initial conditions and control inputs.

5.4.1 Scenario 1: Fixed initial condition and control input

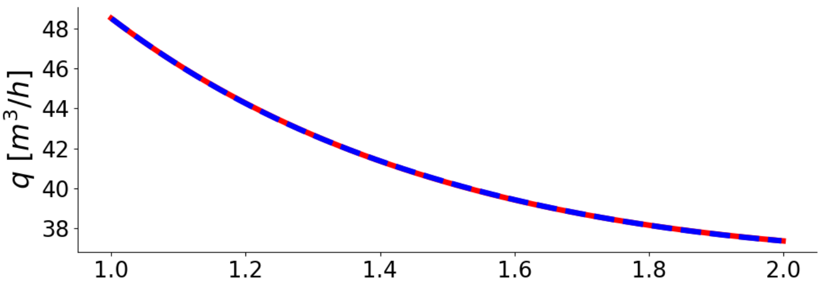
In this scenario, the PINC model was trained using 1000 epochs. The initial condition was fixed and set to $[75\text{bar}, 35\text{bar}, 35m^3/h]$, and the control input was fixed as well, at $[60\text{Hz}, 80\%]$. Figure 5.1 shows the performance of the model. The red line represents the true solution obtained by RK4, while the dotted blue line is the solution obtained using the PINC.



(a)



(b)



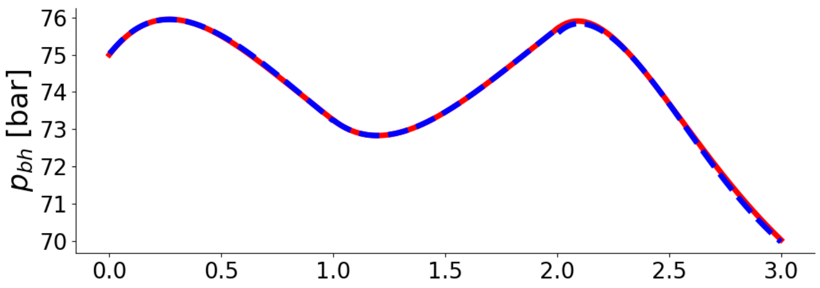
(c)

Figure 5.1: This figure shows a PINC (blue) trained with a fixed initial condition and fixed control input. It is compared against the true solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

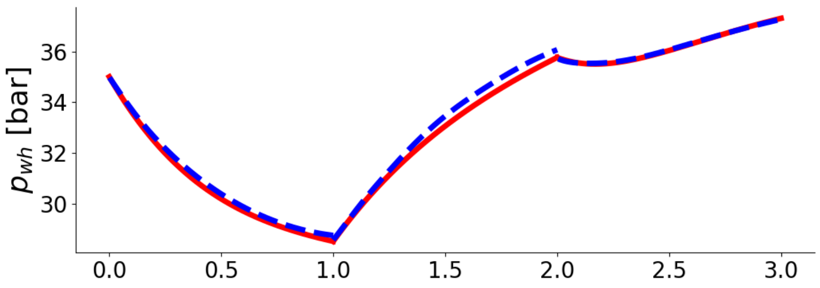
5.4.2 Scenario 2: Long-range simulation with PINC

Here, the network is trained for three sets of initial conditions and control inputs. These are $[75\text{bar}, 35\text{bar}, 35m^3/h]$ as initial condition with $[60\text{Hz}, 80\%]$ as control input, $[73\text{bar}, 28.5\text{bar}, 48.5m^3/h]$ with $[55\text{Hz}, 30\%]$, and $[75.5\text{bar}, 35.7\text{bar}, 37m^3/h]$ with $[65\text{Hz}, 50\%]$.

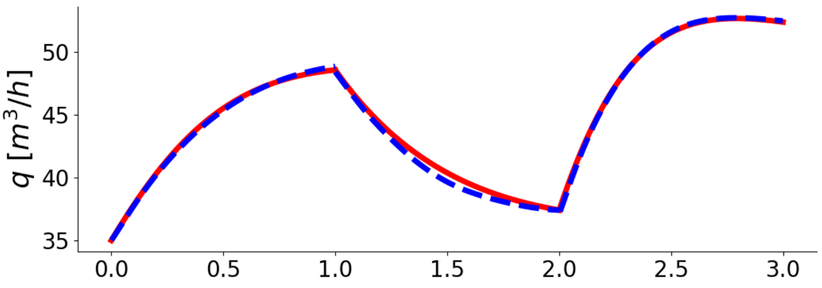
For each condition, there are 15,000 collocation points. The network is trained using batches with similar initial conditions and control inputs. Each condition was trained for a total of 2,000 training steps. Figure 5.2 illustrates the performance of the PINC model (the dotted blue line), compared with the true solution obtained using RK4 (red line).



(a)



(b)



(c)

Figure 5.2: This figure shows a PINC (blue) trained for long-range simulation. It is compared against the true solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

5.4.3 Results of verification

As can be seen from the scenarios tested above, the PINC is able to accurately predict the solution to a problem given a single initial condition and control input. It is also able to predict the solution in the case where the network is trained for multiple initial conditions and control inputs, however with a small discrepancy. The model can be considered verified for use for a single case, however, the results of the verification for multiple inputs indicate the need for further experiments on the topic.

Chapter 6

Results and Findings

When evaluating the performance of the PINN and PINC models, several factors can be taken into account. The first is the overall accuracy of the solution. Is it correct? How well does it compare with the solution gained from numerical solvers such as the RK4? Another important aspect is the time component. What amount of time does it take to train the models, and how long does it take to use them for inference? And how does this compare to the time it takes to use a numerical solver?

6.1 Performance Evaluation of the PINN Model

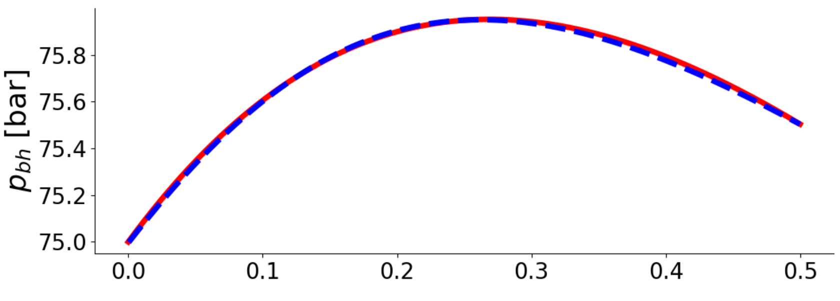
In order to evaluate the performance of the PINN, several scenarios were examined. In the following scenarios, the initial condition was $[x_1, x_2, x_3] = [75\text{bar}, 35\text{bar}, 36m^3/h]$ and the control input was set to $[z, f] = [90\%, 65\text{Hz}]$. Three different time horizons were then examined, 0.1 minute, 1 minute, and 5 minutes. For each time horizon, a new PINN model was trained, and RK4 was used to find a solution as well. Furthermore, the time taken to train the PINN model, use the trained model for obtaining the solution, and the time taken to solve the problem using RK4 was noted. These details are listen in Table 6.1

Table 6.1: This table presents the results of the experiments performed to evaluate the performance of the PINN. It shows three scenarios where the PINN was trained for different simulation times, and with a different number of epochs. The resulting training time, inference time, time to perform RK4 and the training error of the model are listed.

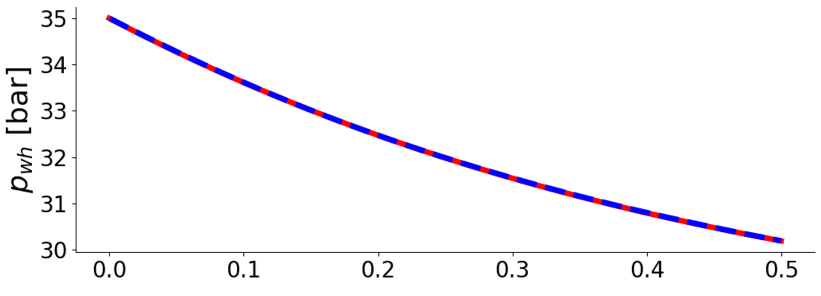
#	Epochs	Sim. time	Training time	Inference time	RK4 time	Training error
1	1000	0.1 min	26.4032 s	0.00265241 s	0.324058 s	2.2461e-06
2	2000	1.0 min	44.7757 s	0.00255489 s	0.268201 s	2.2285e-07
3	5000	5.0 min	108.674 s	0.00189567 s	0.275140 s	6.5441e-08

6.1.1 Scenario 1: 0.5 minute

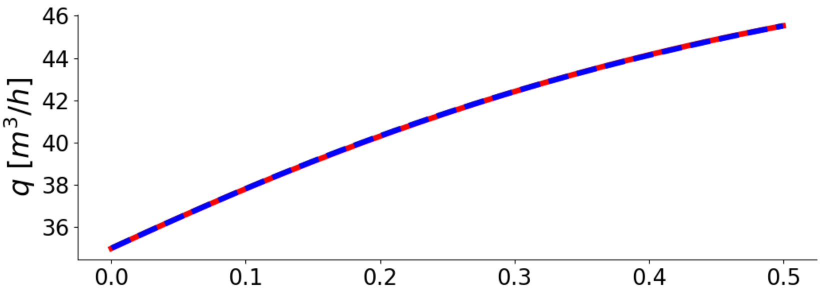
The solutions obtained from PINN and RK4 using a time horizon of 0.5 minute is shown in Figure 6.1. The dashed blue line representing the solution gained from the PINN mostly overlaps the red line representing the solution obtained from RK4.



(a)



(b)

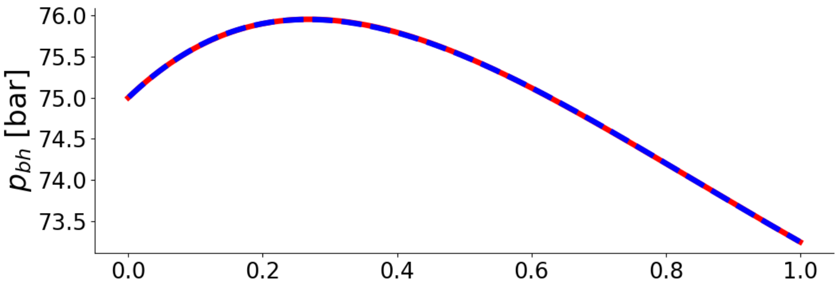


(c)

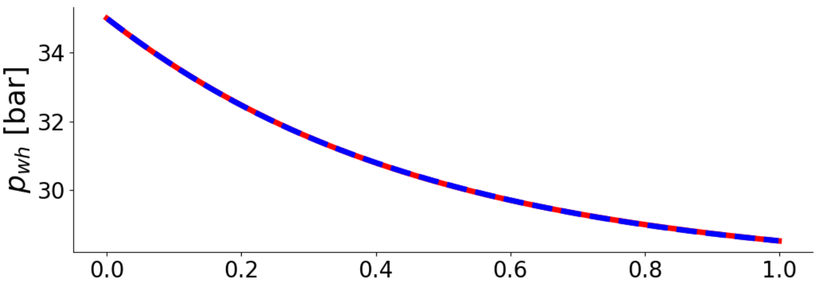
Figure 6.1: This figure shows the PINN (blue) trained for a simulation time of 0.1 minute, compared with the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

6.1.2 Scenario 2: 1 minute

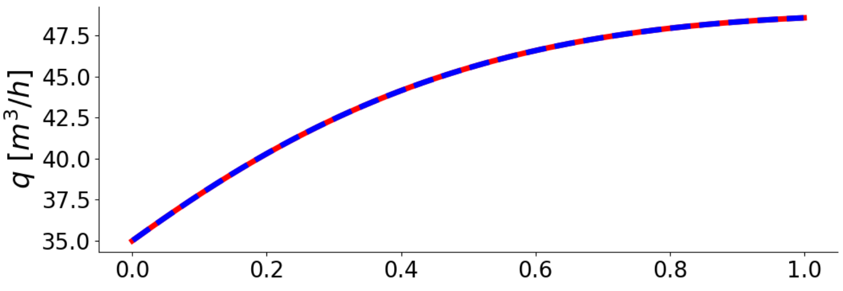
The plot of the different states during one minute is shown in Figure 6.2. As before, the red lines are the state outputs obtained using RK4, while the blue dotted lines are the results of the trained PINN model.



(a)



(b)

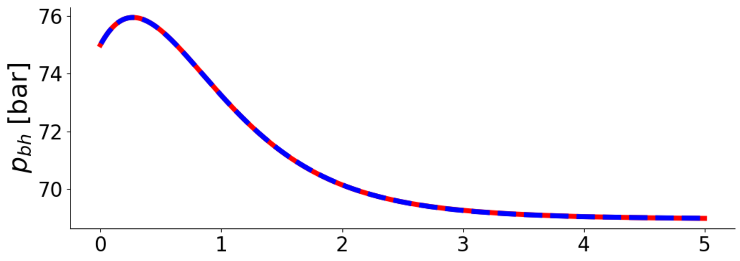


(c)

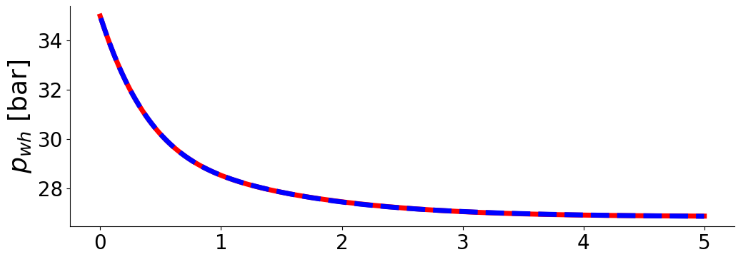
Figure 6.2: This figure shows the PINN (blue) trained for a simulation time of 1 minute, compared with the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

6.1.3 Scenario 3: 5 minutes

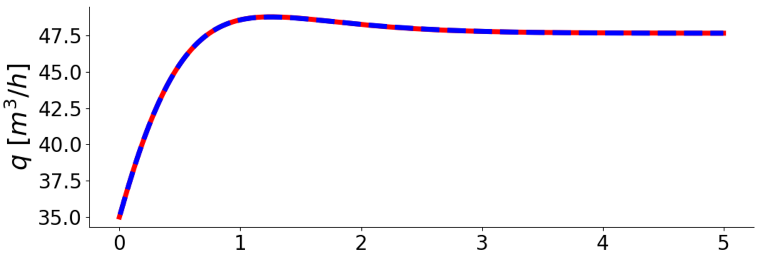
The final figure, Fig 6.3 shows the solutions obtained by PINN, the dotted blue lines, and RK4, the red line, when the PINN model is trained for a time interval of 5 minutes.



(a)



(b)



(c)

Figure 6.3: This figure shows the PINN (blue) trained for a simulation time of 5 minutes, compared with the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

6.2 Performance Evaluation of the PINC Model

In order to evaluate the performance of the PINC model, four scenarios are being tested. The first is that where the PINC is trained for a single initial condition and control input. In the second, the control input is still fixed, while the PINC is trained on an interval of initial conditions. The third and fourth scenarios, respectively, are where one and two of the control inputs are trained on an interval as well. In each of these scenarios, the time horizon used for the PINC is one minute. A summary of the obtained results is listed in Table 6.2.

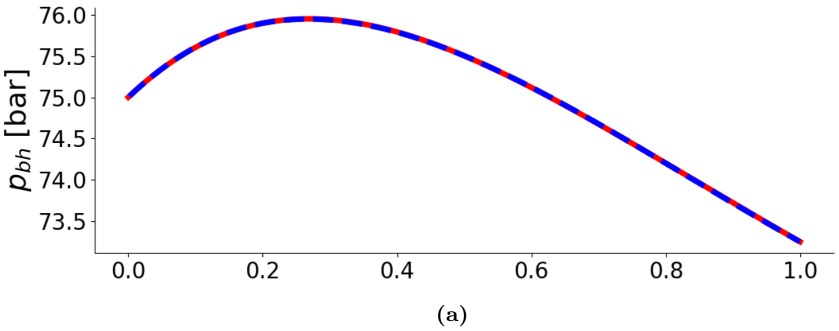
For each of the four scenarios, plots of the three states modeled by the neural network are shown. In addition, the deviation between the solution presented by the neural network and the solution obtained by the numerical solver has been recorded. Plots showing the development of this deviation during the time horizon modeled by the PINC are included as well. The deviation is measured using the Euclidean distance between the two points at each time step. This deviation is shown in two plots. One is showing the deviation for each time step, while the other is showing the average of the deviations up until that time step.

Table 6.2: This table presents the results of the different experiments performed to evaluate the performance of the PINC model. It shows four different scenarios where the PINC was trained varying the intervals of the inputs to the model, and the number of training epochs. In the first scenario, the model is trained for a fixed set of inputs. For the second, the initial conditions vary. In the third scenario, the initial conditions and one control input varies. In the fourth, and final, scenario, all inputs to the model vary. The resulting training time, inference time, time to perform RK4 and the training error of the obtained model are listed.

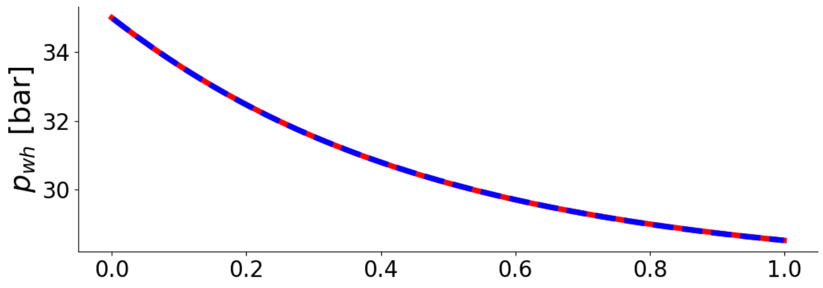
#	Epochs	Training time	Inference time	RK4 time	Error
1	1000	18.37266 s	0.00330234 s	0.227689 s	2.4531e-07
2	3000	781.484 s	0.0114233 s	0.459043 s	7.6598e-06
3	5000	1224.08 s	0.0110455 s	0.517097 s	4.4435e-05
4	10000	2414.29 s	0.00353932 s	0.233735 s	5.5706e-05

6.2.1 Scenario 1: Fixed initial condition and control input

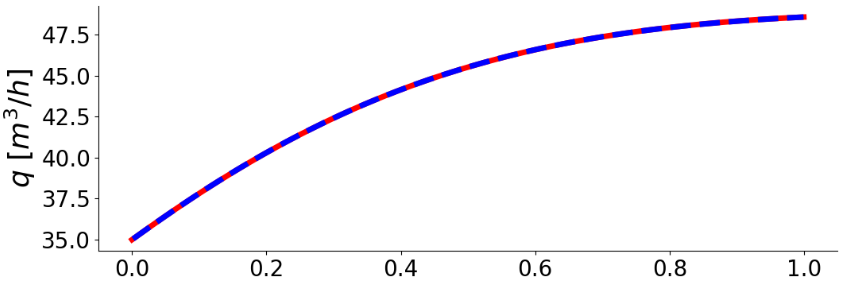
Here the results of training the PINC using a fixed initial condition and a fixed control input are presented. The network is trained with the initial condition being $[x_1, x_2, x_3] = [75\text{bar}, 35\text{bar}, 35m^3/h]$, and control input fixed to $[z, f] = [80\%, 60\text{Hz}]$. The training data of the model consists of 1 data point and 1000 collocation points. The PINC model is trained using 1000 training epochs.



(a)



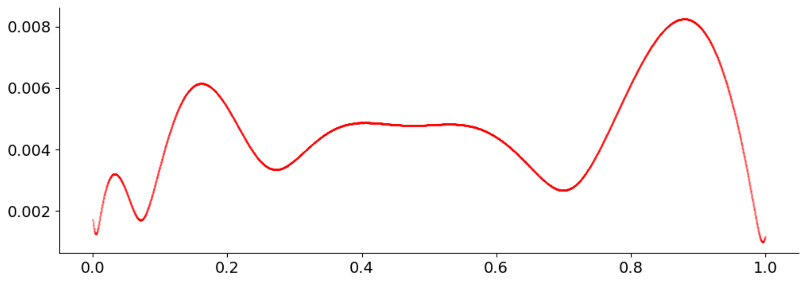
(b)



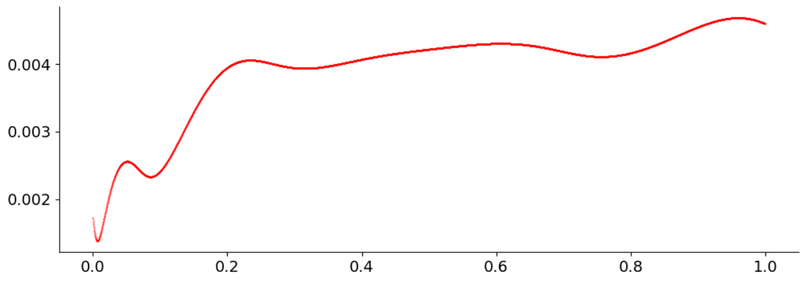
(c)

Figure 6.4: This figure shows the comparison between PINC (blue) trained with only one initial condition and control input, and the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

The first, Figure 6.4, shows the comparison between the results obtained using the PINC model and the true solution obtained using RK4. The second figure, Figure 6.5 shows the deviation between these two solutions, and the development of this deviation during the time interval.



(a)



(b)

Figure 6.5: This figure shows the development of the deviation (measured in Euclidean distance) during the simulated time interval. (a) Shows the deviation at any given time point. (b) Shows the average deviation up until any given time point

6.2.2 Scenario 2: Interval of initial conditions

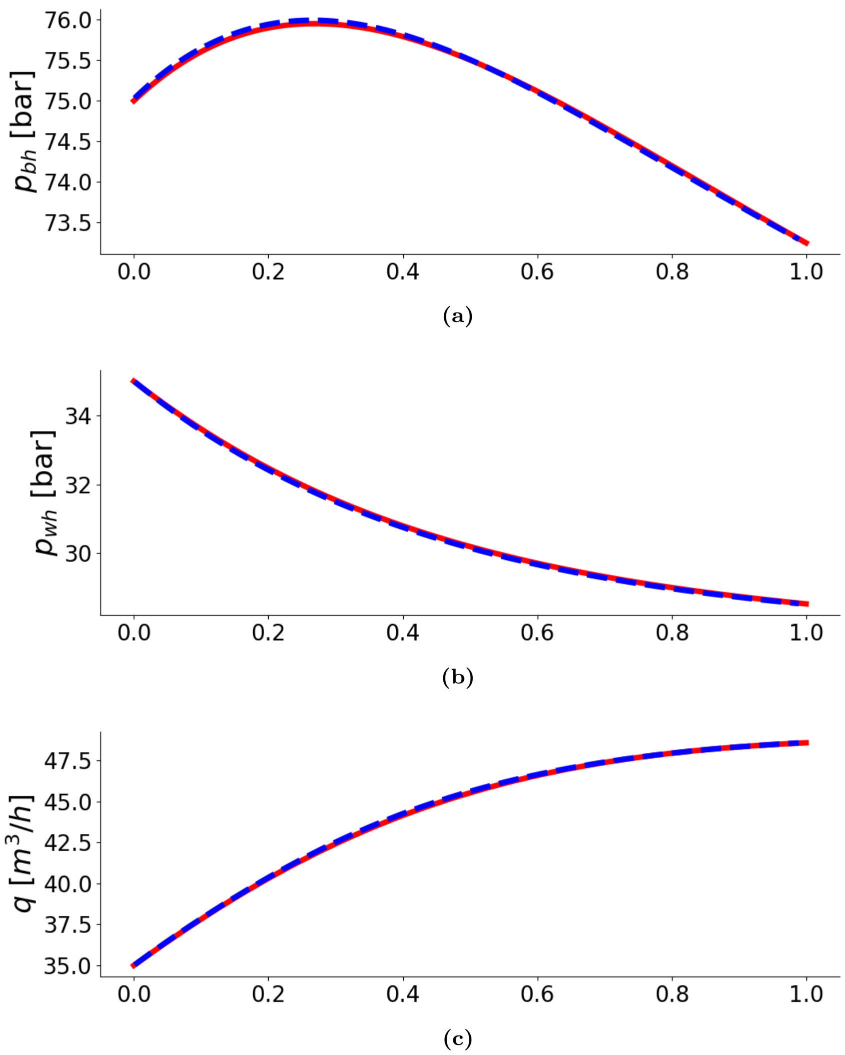


Figure 6.6: This figure shows the comparison between PINC (blue) trained with the initial condition within an interval and a constant control input, and the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

The results presented here stem from keeping the control input fixed, while the initial conditions are sampled from an interval. The control input is still being fixed at $[z, f] = [80\%, 60\text{Hz}]$. The initial conditions of the system are now sampled randomly between $[x_1^{min}, x_2^{min}, x_3^{min}] = [70\text{bar}, 30\text{bar}, 30m^3/h]$ and $[x_1^{max}, x_2^{max}, x_3^{max}] = [80\text{bar}, 40\text{bar}, 40m^3/h]$. A total of 100 data points for the initial condition are sampled, with a corresponding 1000 collocation points. The network is trained using 3000 epochs.

Figure 6.6 compares the output of the PINC model, shown with a dashed blue line, and the solution from RK4, shown with a red line. Figure 6.7 illustrates the difference between the two.

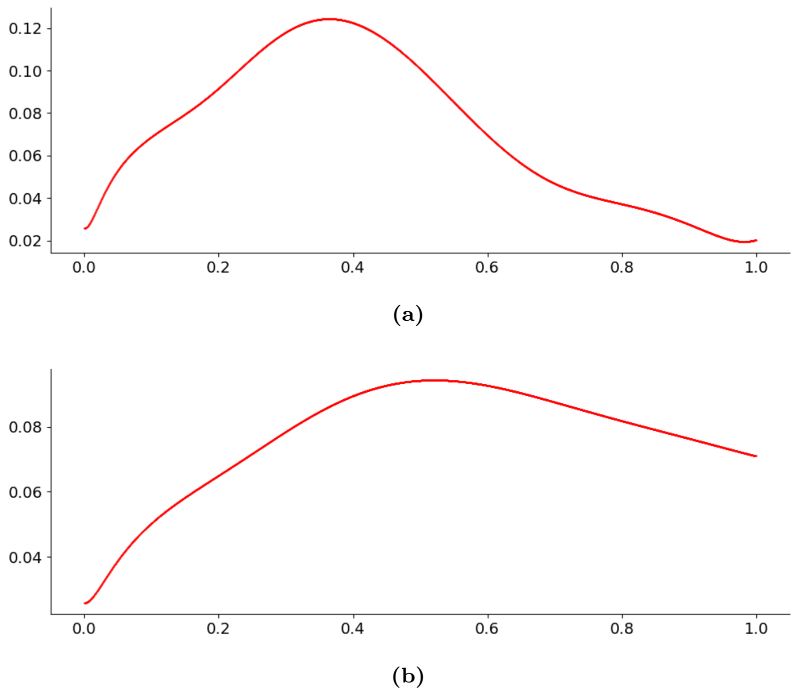


Figure 6.7: This figure shows the development of the deviation (measured in Euclidean distance) during the simulated time interval. (a) Shows the deviation at any given time point. (b) Shows the average deviation up until any given time point

6.2.3 Scenario 3: Interval of initial conditions and one control input

In this scenario, not only the initial condition but also one of the control inputs are sampled at random intervals. The initial condition is still randomly picked from the interval ranging from $[x_1^{min}, x_2^{min}, x_3^{min}] = [70\text{bar}, 30\text{bar}, 30m^3/h]$ and $[x_1^{max}, x_2^{max}, x_3^{max}] = [80\text{bar}, 40\text{bar}, 40m^3/h]$.

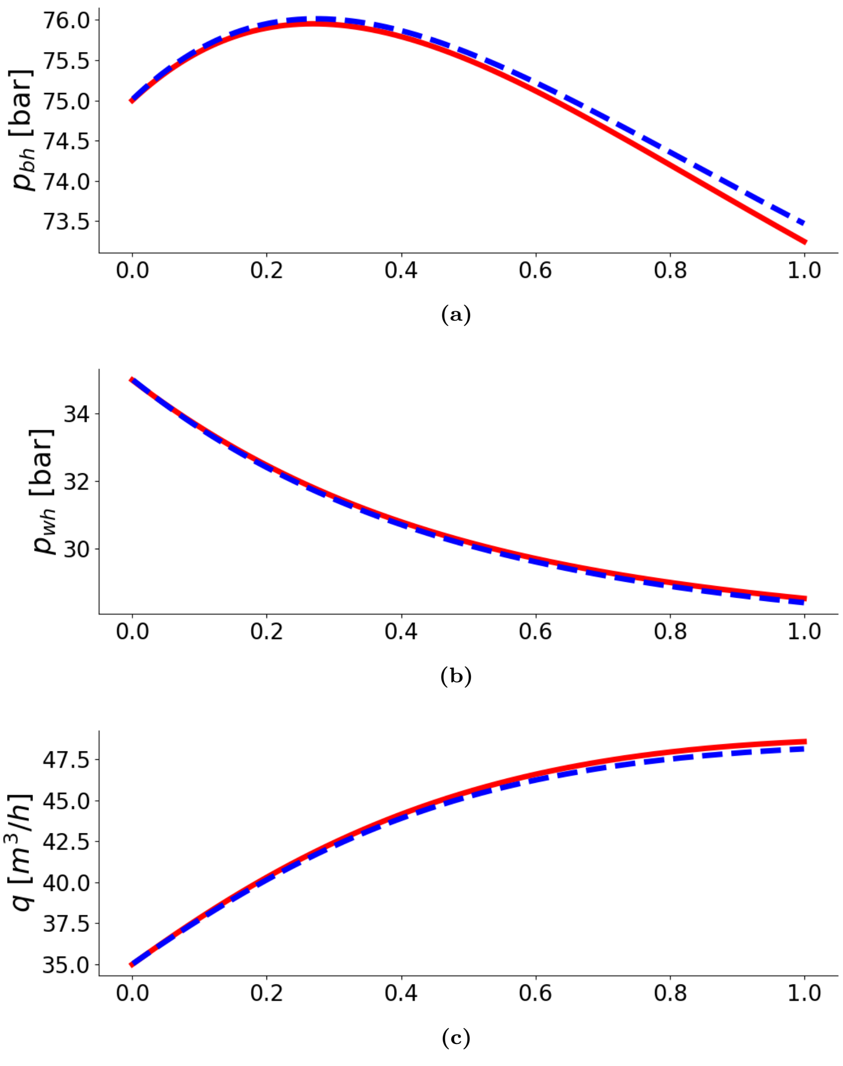
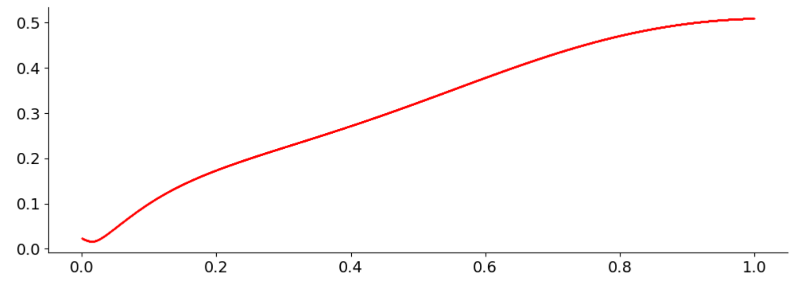


Figure 6.8: This figure shows the comparison between PINC (blue) trained with the initial condition and control frequency within an interval and a constant valve opening, and the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

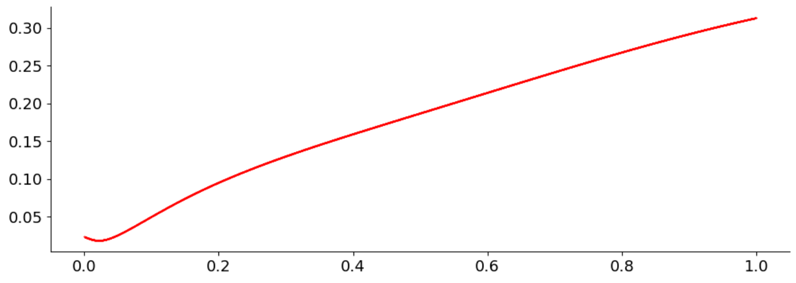
The control frequency is sampled between the values 59.5 Hz and 60.5 Hz. The valve opening is still held constant at 80%.

The network is trained using 100 data points covering different initial conditions and control inputs, and a corresponding 1000 collocation points. The network is trained using 5000 epochs.

The plots of the results obtained are shown in Figure 6.8. The output of the PINC model is shown in dashed blue lines, while the solution of the system is shown in red. The evolution of the deviation between these two is illustrated in Figure 6.2.3.



(a)



(b)

Figure 6.9: This figure shows the development of the deviation (measured in Euclidean distance) during the simulated time interval. (a) Shows the deviation at any given time point. (b) Shows the average deviation up until any given time point

6.2.4 Scenario 4: Interval of initial conditions and two control inputs

In the final scenario, all of the inputs are sampled from intervals, that is, the initial conditions and both control inputs. The initial conditions are still sampled from the interval $[x_1^{min}, x_2^{min}, x_3^{min}] = [70\text{bar}, 30\text{bar}, 30m^3/h]$ and $[x_1^{max}, x_2^{max}, x_3^{max}] = [80\text{bar}, 40\text{bar}, 40m^3/h]$. The control inputs are sampled between $[z^{min}, f^{min}] = [75\%, 59.5\text{Hz}]$ and $[z^{max}, f^{max}] = [80\%, 60.5\text{Hz}]$.

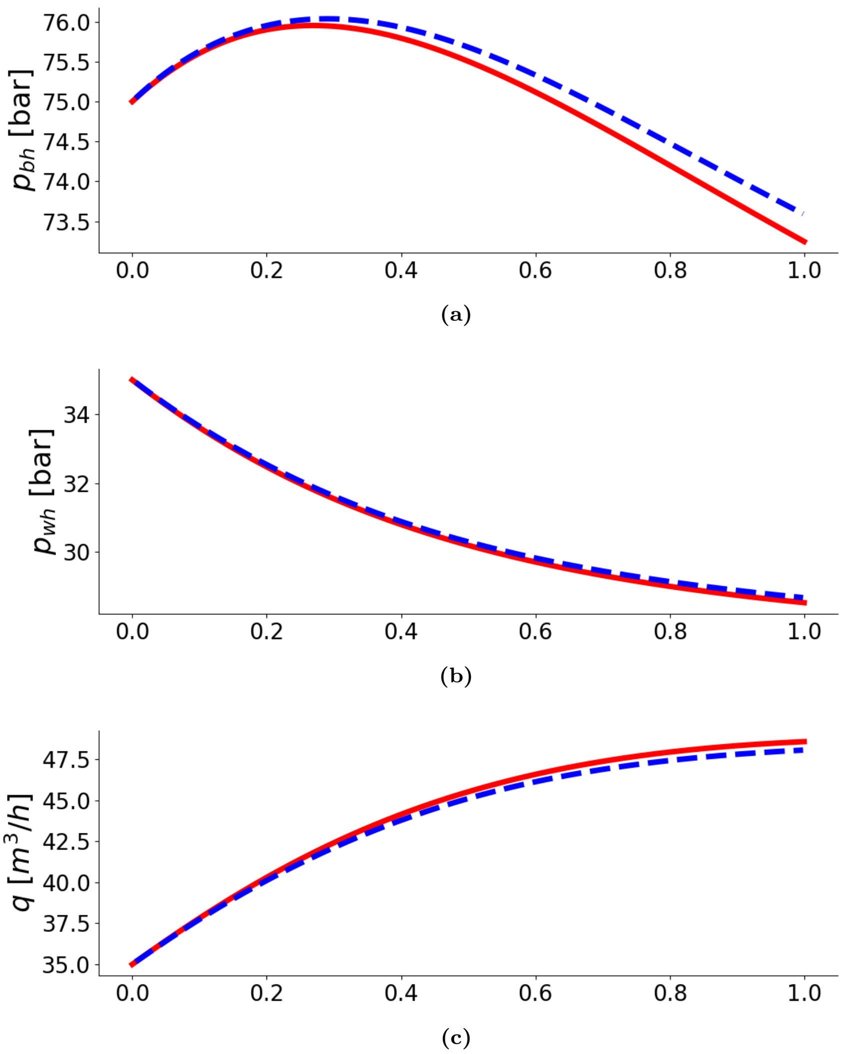
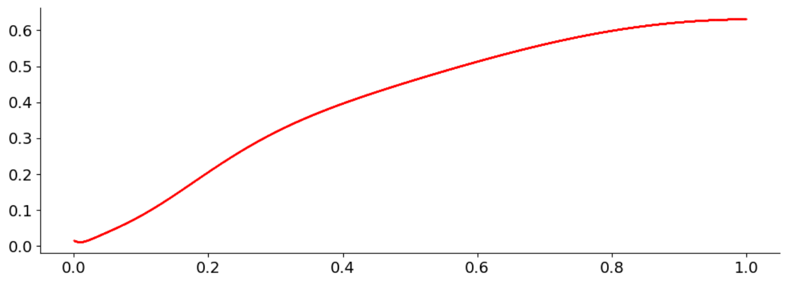


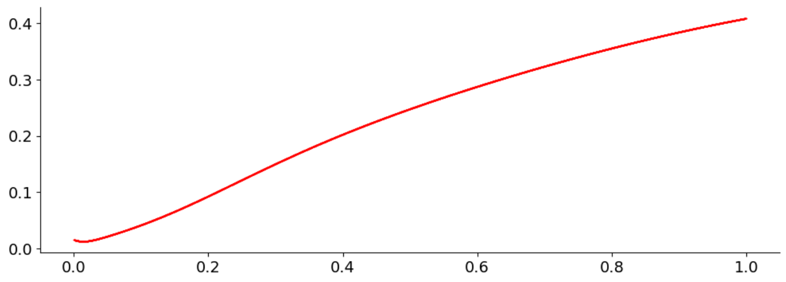
Figure 6.10: This figure shows the comparison between PINC (blue) trained with the initial condition and control inputs trained on intervals, and the solution obtained using RK4 (red). (a) shows the solution for the bottomhole pressure. (b) shows the wellhead pressure. (c) is the flow.

As before, the network is trained using 100 sampled points covering the range of initial conditions and control inputs, with 1000 corresponding collocation points. This time the network is trained using 10.000 epochs.

As with the previous scenarios, the solution obtained using the PINC (blue) is shown in comparison with the true solution of the system (red) in Figure 6.10. To illustrate the performance of the solution obtained using PINC with time, the discrepancy between the two has been calculated, and Figure 6.11 illustrates this evolution.



(a)



(b)

Figure 6.11: This figure shows the development of the deviation (measured in Euclidean distance) during the simulated time interval. (a) Shows the deviation at any given time point. (b) Shows the average deviation up until any given time point

Chapter 7

Discussion

This chapter will elaborate on the interpretations of the results presented in Chapter 6. The limitations and challenges of the PINN and PINC will be treated as well. Finally the practical implications of the results will be detailed.

7.1 Interpretation of Results for PINN

As could be seen in Section 4.4, the PINN model is able to produce results very similar to those obtained using traditional numerical methods such as RK4. Furthermore, in Section 6.1 the performance of different PINN models for different time horizons was examined. When evaluating the performance of the PINN model in these scenarios, there are several metrics to be considered.

The first aspect is the ability of the PINN to provide the correct solution to a given problem. For the purpose of evaluating the obtained model, the solution was calculated using the numerical method RK4 as well. The results obtained in Section 6.1 show that the PINN does indeed provide a similar solution to that obtained using RK4. These results are in line with previous research on PINNs such as [30],[33], and [34], that have shown that these networks are able to accurately obtain solutions to a variety of problems, including power systems, fluid dynamics, and heat transfer problems.

The performance of the PINN model in terms of computation time is an interesting point of comparison to traditional numerical methods like the RK4 solver. The duration of the training process for the PINN will vary based on the complexity of the problem at hand, and in general, it requires significantly more time than the straightforward numerical computations performed by the RK4. This is in line with the results obtained in Section 6.1, where it can be seen from Table 6.1 that the training time of the PINN models is far larger than the time taken for RK4 to compute the solution for the problem.

However, a significant advantage of the PINN approach is revealed once the network is trained. Upon completion of the training phase, the PINN can rapidly deliver the desired solution, as seen from the inference times found in Table 6.1. Even in scenario 3, where the training of the network takes almost 400 times as long as the RK4 solver, the inference time is less than one-hundredth of the time taken by RK4. Thus, while predicting, the PINN is able to outpace the time-consuming nature of RK4 calculations. This trade-off between extended training time and subsequent fast inference time, allowing for a swift generation of the solution, positions PINN as a strong contender in scenarios where repeated evaluations are crucial.

7.2 Interpretation of Results for PINC

As was shown in Section 5.4, when training the PINC for the ESP for just a single initial condition and a single control input, it is able to obtain accurate results just as the PINN. When the model is trained for only a few different initial conditions and/or control inputs it performs with a declined accuracy, with an observable deviation between the true solution of the system and the predictions provided by the PINC model. Thus, it is evident that the model faces some challenges.

Just like the PINN model, the PINC framework encounters temporal constraints. An issue that is still notable, and in fact even more pronounced for the PINC, is its extended training time. In order to obtain the desired accuracy, the PINC requires a far higher training time than the PINN model. This is due to the fact that the PINC has to train for a variety of initial conditions and control inputs. Unlike the quick calculations of the RK4 solver, the PINC's training process is far more time-consuming. This makes the PINC less promptly deployable, especially when dealing with more complex problems, and/or problems with a large feasible region for the initial condition and control inputs.

However, like the PINN, once the PINC is trained it has a great advantage in its efficiency during the inference phase. Again, the solution prediction outpaces the computational demands of the RK4 solver, as seen in Table 6.2. This renders the PINC with a great advantage, especially in the case of real-time applications where fast solutions are essential.

Another interesting point to note from the results presented in Section 6.2, is how the error of the solution increases as the network is trained on intervals, rather than fixed points, for the initial condition and control inputs. In the first scenario, the PINC is effectively working as a PINN, only being trained for one single initial condition and one control input. The result of this is an accurate solution, that matches that obtained using RK4.

However, already in the second scenario, where the PINC is trained on an interval of initial conditions, but with a fixed control input, it can be seen that the solution is becoming less accurate. This happens despite the model being trained using a higher number of epochs. These results are in line with the results presented in [47], which indicated that training a PINN network with varying initial conditions results in high training times without a proportional decrease in the training error.

This becomes yet more evident in scenarios 3 and 4, wherein scenario 3 PINC is trained using an interval of initial conditions, an interval for the frequency of the system, and a constant valve opening. In scenario 4, the valve opening as well is set to be within an interval of possible values. The results of these scenarios show that when increasing the training samples of the PINC to include varying control input as well, the obtained solutions become even less accurate. However, the error of the obtained solution, especially in scenario 3 and 4, can be observed to grow bigger over time. Thus, shorter time horizons are a necessity in order for the models to be reliable, so in order to obtain long-range simulations, the PINC needs to be used in self-loop mode.

For the purpose of this thesis, several attempts were made to improve the performance of the developed PINC model. This included increasing the amount of training data, both for the initial values and the collocation points. Attempts were made using a higher number of training epochs as well. Attempts were also made at changing the order of the training data, and using batches in order to see whether this would affect the overall results. Finally, the prediction horizon of the network was decreased. However, most of these changes proved to have little effect on the performance of the PINC model. Changing the amount of training data or the number of training epochs did provide slightly better models, however, these experiments were time- and resource-consuming.

While the problem of accuracy to a certain extent can be mitigated by small adaptions to the PINC model and its training, the underlying issue still remains. There are several possible solutions that can be examined in order to minimize the problem. One is to look into the optimizer used. While this thesis only implements the PINC network using the ADAM optimizer, in works such as [45] and [43], this optimizer was used in combination with the L-BFGS. Another could be to look at the proposed improvements to the original PINN formulation and incorporate these ideas into the PINC model as well, this includes dynamic weighting strategies, a change of the activation function, and strategic sampling of collocation points for improved training efficiency.

However, other factors such as the complexity of the systems modeled and the size of the feasible region might also factor in. Some systems might require a more elaborate neural network structure in order to accurately capture the dynamics of the system in different control settings. More training could also be a necessity, however due to restrictions on hardware this might not always be a feasible solution.

7.3 Limitations and Challenges

One of the main challenges regarding the PINN, and especially the PINC, is the efficiency in relation to the training. In relation to the PINN model, this mostly applies if the neural network is used to solve a complex system. In such cases, the PINN requires a larger amount of collocation points in order to properly approximate the real solution of the differential equations describing the system

The PINC, compared to the PINN, consists of several more inputs. In the case of the ESP system, the PINN has one single input, which is time, while the PINC has six: the initial conditions of the three states, the two control inputs applied, and the time. In order to fully utilize the capabilities of the PINC, it is necessary to train the model for a variety of different initial conditions and control inputs within the feasible region of the system. That way, the network can be used in combination with controllers such as the MPC. However, this requires extensive training of the network. For complex systems requiring many data points, systems with a large set of inputs, or systems with a large feasible region, the training of the PINC model may be an issue.

Another issue in regards to the PINC model, is the necessity of defining a feasible region for all inputs. As the model may become unstable if it's trained on infeasible inputs, this is an important step. However, obtaining this region may require in-depth knowledge of the system at hand, which may not always be easy to obtain. One solution could be to check the feasibility of different states using a solver such as RK4. However, for large systems containing potentially a very large number of variables, this approach becomes unrealistic.

7.4 Implications for Practical Applications

As was shown in Chapter 6, when used for inference, physics-informed machine learning models, are able to find the solution to the differential equations fast, often outperforming traditional numerical methods like the RK4. This is a great advantage when time is of the essence, such as in real-time decision support. In control applications, if a controller necessities a model of the system dynamics in order to determine the appropriate control action, time is critical as the system continues to run while these calculations are happening. In these applications, the PINC can be used to predict the solution in a swift manner, thus quickly offering insights and recommendations to support agile decision-making.

Furthermore, the flexibility of the PINN makes the model suitable for practical applications that involve adapting to changing conditions and diverse inputs. This is especially true in real-world scenarios that require dynamic responses and effective control strategies. Similarly, the PINC excels in practical situations where the dynamics vary, as it is able to be used for a variety of different conditions and inputs. In fact, both PINN and PINC offer benefits for real-world applications, as they both are able to provide adaptable and efficient solutions to complex challenges.

In practical applications, access to data may be somewhat limited, and the system might also deviate from the theoretical behavior. In such situations, the PINN and PINC models come at a great advantage, as they have an inherent ability to generalize, thus making them able to make informed predictions beyond the data used for training. This can provide great benefits in practical applications where there for instance is access to sparse data. Conventional numerical solvers might have difficulties obtaining accurate outcomes in regions characterized by limited data points. However, PINN models have the ability to combine the available data with other known information of the system, rendering it capable of approximating solutions even using a very limited number of data points. In fact, as demonstrated in Chapter 6, both the PINN and the PINC models are capable of providing accurate results with only a single data point for the ESP system. Thus, PINN and PINC allow for enhanced prediction accuracy for practical applications where there are data irregularities.

Yet another key point of these types of models that make them useful in several practical applications, is that they are able to solve problems that traditional numerical methods may face difficulties. This includes applications such as high-dimensional systems of equations [46], where the computational requirements increase exponentially for numerical methods as the dimension of the system grows. Another example is in the solution of inverse problems [24], where the PINN can learn the mapping between inputs and outputs directly, whereas traditional numerical methods could require iterative optimization techniques.

Despite having numerous positives that can be leveraged in practical applications, there are still challenges to be faced, most notably that of the extensive training time. While both the PINN and the PINC hold great promise, the duration of the training phase for the models may potentially be hindering their application in time-sensitive scenarios. As a considerable amount of time and computational resources are required for training these models, this may impact the speed at which such models can be deployed, especially in fast-paced environments. Therefore, it might be necessary to do an assessment of the trade-off between the extensive training time and the subsequent benefits these models offer when determining their applicability and suitability for practical applications where agile and prompt responses are required.

Chapter 8

Conclusion and future work

This chapter summarizes the main results and findings in this thesis. Furthermore, possible areas for future research are highlighted.

8.1 Summary of Findings

The objective of this thesis has been to create a Physics-Informed Neural Network (PINN) for an Electric Submersible Pump (ESP) and to evaluate how these networks can be used to predict the behavior of the ESP, as well as how this network can be extended to a Physics-Informed Neural Net for Control (PINC) to be used in control applications.

8.1.1 PINN

Since they were first proposed in 2017, PINNs have gained wide attention, due to their grand potential within a large field of application areas. In this thesis, a PINN for the ESP was proposed and verified against the numerical solution obtained using RK4.

In line with other research on the topic, the results obtained indicated that the PINN is indeed capable of predicting a solution to the system that matches well that which can be found using classical numerical solvers. While the training time of the PINN network is significantly larger than the time taken by the RK4 method, the inference, or prediction, time of the trained network is significantly lower than the time taken by RK4.

Another advantage of the PINN model that was observed in this thesis, is its ability to solve the system even for sparse amounts of data. In the experiments conducted in this thesis, only one data point was used, that is the initial condition. Yet, the network was able to find accurate solutions to the system of differential equations.

8.1.2 PINC

Many of the key advantages of the PINN models, such as their flexibility and their short inference time, make these models suitable for use in control applications. However, the PINN in itself can not readily be used in such settings, as the input does not include the control action or initial condition. To remediate this issue, the PINC was proposed. This extended version of the PINN includes the control input and initial condition of the system as input, thus allowing it to be used in control applications.

However, the PINC suffers from a known issue of the PINN, which is that the error of the predicted solution gets larger with time. That means the model can only be used for short time horizons. However, a solution to still obtain long-range simulations has already been proposed, and it consists of chaining the network so that the final condition of the network will be used as the initial condition in the next iteration. This requires the network to be trained for a variety of different initial conditions and control inputs, in order to be able to give accurate responses.

In this thesis, several PINC models were trained. In order to examine the generalization capabilities of the network, it varied how many of the inputs of the network were fixed and how many were chosen from an interval of values. The PINC obtained good results when all inputs were fixed, and also while the initial conditions varied while the control input was fixed. However, the latter already required far more training to obtain similar results. When extending the varying input to include one or both of the control inputs, the error in the solution grew even more, especially with time.

While appearing promising, the issues of training the PINC models should not be ignored. For complex problems, and for problems with larger feasible regions, training a PINC to perform well for a variety of conditions will require many computational resources and a large training time. While examining the exact time of such a process is outside of the scope of this thesis, this has been studied for a regular PINN extended to be used for multiple initial conditions in [47], and the results indicated that the time taken for proper training of such a network is within the scope of days and weeks.

8.2 Future Research Directions

As was detailed in the theoretical background on the PINN, there are several ways of formulating the loss function of the network to incorporate knowledge of the physical dynamics of a system. However, this work, as well as other current research on PINC mainly utilizes the basic formulation of the loss function. An interesting future direction of research would be to incorporate more terms to the loss function of the PINC, to see how this affects its performance.

Furthermore, in this work, only one data point was considered. The same was done in other work on PINC, such as [43] and [45]. However, in the implementation of the regular PINN, the term for data points opens up the ability to combine data gathered from the actual behavior of the system, and training the network using a combination of this and the known dynamics. This is especially useful if the dynamics of the system are not necessarily precisely known, which in most processes is the case. Thus, another potential direction for future research within the field of PINC is to look into the possibility of incorporating real system data into the training process.

Another point to consider is one of the key disadvantages of the PINN and PINC models, which is their long training time. In this thesis, both models were trained using NVIDIA's T4 GPU, yet in order to train the PINC model for a sufficiently large set of initial values and control inputs, significant time is needed. For complex systems, or systems where a wide range of initial conditions and/or control inputs are to be considered, the training may become very inefficient. Thus, in order to truly unlock the potential within physics-informed machine learning, a key direction of future research is how to efficiently train these networks.

Finally, while the PINC is a fairly new proposal, it is based upon the more researched PINN model. As highlighted in Section 2.3.3, much effort has already been made to render the original PINN more effective and capable of tackling a wider array of complex problems. Thus, a possible direction for future research on the topic of PINC could be to explore whether implementing these proposed improvements in the PINC model renders it more efficient. This includes exploring the effect of weighting the terms of the loss function in an intelligent manner, looking into the possible effects of changing the activation function used, and also improving training efficiency by generating the collocation points more effectively.

8.3 Final Remarks

This thesis presented a Physics-Informed Neural Network (PINN) and a Physics-Informed Neural Net for Control (PINC) for the Electric Submersible Pump (ESP). The experiments concluded on the PINN model gave results in line with current research on the topic, proving that this type of network is able to provide an accurate solution to the system for this problem.

The PINC model treated in this thesis also shows great promise, however, the results obtained for this model for the ESP indicate that more research is needed before such models can be applied. While having great potential, the issue of training these networks in an effective and reliable manner remains.

Bibliography

[1] H. Robbins and S. Monro, “A stochastic approximation method”, *The annals of mathematical statistics*, pp. 400–407, 1951.

[2] E. Fehlberg, “Classical fourth-and lower order runge-kutta formulas with stepsize control and their application to heat transfer problems”, *Computing*, vol. 6, pp. 61–71, 1970.

[3] T. Hull, W. Enright, B. Fellen, and A. Sedgwick, “Comparing numerical methods for ordinary differential equations”, *SIAM Journal on Numerical Analysis*, vol. 9, no. 4, pp. 603–637, 1972.

[4] M. L. Nave, “Considerations for application of electrical submersible pumps for underground coal mine dewatering”, in *Conference Record of the 1988 IEEE Industry Applications Society Annual Meeting*, IEEE, 1988, pp. 1263–1266.

[5] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators”, *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[6] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization”, *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[7] T. Ragg, H. Braun, and H. Landsberg, “A comparative study of neural network optimization techniques”, in *Artificial Neural Nets and Genetic Algorithms*, 1997, pp. 341–345.

[8] W. A. Limanowka, M. M. Voytechek, and R. E. Limanowka, “Asphaltene deposition problems in oil industry with focus on electric submersible pump applications”, in *SPE Annual Technical Conference and Exhibition?*, SPE, 1999, SPE–56 662.

[9] M. H. Carpenter, C. Kennedy, H. Bijl, S. Viken, and V. N. Vatsa, “Fourth-order runge-kutta schemes for fluid mechanics applications”, *Journal of Scientific Computing*, vol. 25, pp. 157–194, 2005.

[10] S. Berezin, "Submersible pumps for wastewater applications", *World pumps*, vol. 2006, no. 480, pp. 26–30, 2006.

[11] S. C. Chapra, *Numerical methods for engineers*. Mcgraw-hill, 2010.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[13] A. Pavlov, D. Krishnamoorthy, K. Fjalestad, E. Aske, and M. Fredriksen, "Modelling and model predictive control of oil wells with electric submersible pumps", in *2014 IEEE Conference on Control Applications (CCA)*, ISSN: 1085-1992, Oct. 2014, pp. 586–592. DOI: `10.1109/CCA.2014.6981403`.

[14] B. J. T. Binder, A. Pavlov, and T. A. Johansen, "Estimation of flow rate and viscosity in a well with an electric submersible pump using moving horizon estimation", *IFAC-PapersOnLine*, 2nd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production OOGP 2015, vol. 48, no. 6, pp. 140–146, Jan. 1, 2015, ISSN: 2405-8963. DOI: `10.1016/j.ifacol.2015.08.022`.

[15] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction", *Computational and Structural Biotechnology Journal*, vol. 13, pp. 8–17, Jan. 1, 2015, ISSN: 2001-0370. DOI: `10.1016/j.csbj.2014.11.005`.

[16] M. Mijwil, *History of Artificial Intelligence*. Apr. 1, 2015, vol. 3, 1 p., Pages: 8. DOI: `10.13140/RG.2.2.16418.15046`.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] A. Karpatne, G. Atluri, J. Faghmous, *et al.*, "Theory-guided data science: A new paradigm for scientific discovery", *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, Oct. 1, 2017. DOI: `10.1109/TKDE.2017.2720168`.

[19] N. G. Maity and S. Das, "Machine learning for improved diagnosis and prognosis in healthcare", in *2017 IEEE aerospace conference*, IEEE, 2017, pp. 1–9.

[20] G. Takacs, *Electrical submersible pumps manual: design, operations, and maintenance*. Gulf professional publishing, 2017.

[21] G. Valdes, C. B. Simone II, J. Chen, *et al.*, "Clinical decision support of radiotherapy treatment planning: A data-driven machine learning strategy for patient-specific dosimetric decision making", *Radiotherapy and Oncology*, vol. 125, no. 3, pp. 392–397, 2017.

[22] N. Baker, F. Alexander, T. Bremer, *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence", Feb. 2019. DOI: `10.2172/1478744`.

[23] M. Leo, S. Sharma, and K. Maddulety, "Machine learning in banking risk management: A literature review", *Risks*, vol. 7, no. 1, p. 29, 2019.

[24] M. Raissi, P. Perdikaris, and G. E. 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, Feb. 1, 2019, ISSN: 0021-9991. DOI: `10.1016/j.jcp.2018.10.045`.

[25] C. Desai, “Comparative analysis of optimizers in deep neural networks”, *International Journal of Innovative Science and Research Technology*, vol. 5, no. 10, pp. 959–962, 2020.

[26] N. Fatima, “Enhancing performance of a deep neural network: A comparative analysis of optimization algorithms”, *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 9, no. 2, pp. 79–90, 2020.

[27] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks”, *Journal of Computational Physics*, vol. 404, p. 109 136, 2020, ISSN: 0021-9991. DOI: `https://doi.org/10.1016/j.jcp.2019.109136`.

[28] R. S. Lee, *Artificial intelligence in daily life.* Springer, 2020.

[29] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, “Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions”, *Information and software technology*, vol. 127, p. 106 368, 2020.

[30] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, “Physics-informed neural networks for power systems”, in *2020 IEEE Power Energy Society General Meeting (PESGM)*, 2020, pp. 1–5. DOI: `10.1109/PESGM41954.2020.9282004`.

[31] S. E. Whang and J.-G. Lee, “Data collection and quality challenges for deep learning”, *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3429–3432, 2020.

[32] J. Blechschmidt and O. G. Ernst, “Three ways to solve partial differential equations with neural networks—a review”, *GAMM-Mitteilungen*, vol. 44, no. 2, e202100006, 2021.

[33] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, “Physics-informed neural networks (pinns) for fluid mechanics: A review”, *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.

[34] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks for heat transfer problems”, *Journal of Heat Transfer*, vol. 143, no. 6, p. 060 801, 2021.

[35] I. Y. Chen, E. Pierson, S. Rose, S. Joshi, K. Ferryman, and M. Ghassemi, “Ethical machine learning in healthcare”, *Annual review of biomedical data science*, vol. 4, pp. 123–144, 2021.

[36] Y. K. Dwivedi, L. Hughes, E. Ismagilova, *et al.*, “Artificial intelligence (ai): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy”, *International Journal of Information Management*, vol. 57, p. 101 994, 2021.

[37] G. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning”, pp. 1–19, May 24, 2021. DOI: `10.1038/s42254-021-00314-5`.

[38] 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.

[39] S. Markidis, “The old and the new: Can physics-informed deep-learning replace traditional linear solvers?”, *Frontiers in big Data*, vol. 4, p. 669 097, 2021.

[40] D. Mhlanga, “Artificial intelligence in the industry 4.0, and its impact on poverty, innovation, infrastructure development, and the sustainable development goals: Lessons from emerging economies?”, *Sustainability*, vol. 13, no. 11, p. 5788, 2021.

[41] C. Rackauckas, Y. Ma, J. Martensen, *et al.*, *Universal differential equations for scientific machine learning*, 2021. arXiv: `2001.04385 [cs.LG]`.

[42] A. Ali, S. Abd Razak, S. H. Othman, *et al.*, “Financial fraud detection based on machine learning: A systematic literature review”, *Applied Sciences*, vol. 12, no. 19, p. 9637, 2022.

[43] E. A. Antonelo, E. Camponogara, L. O. Seman, E. R. de Souza, J. P. Jordanou, and J. F. Hubner, *Physics-informed neural nets for control of dynamical systems*, May 31, 2022. DOI: `10.48550/arXiv.2104.02556`. arXiv: `2104.02556[cs]`.

[44] S. Cuomo, V. S. Di 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”, *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, Jul. 26, 2022, ISSN: 1573-7691. DOI: `10.1007/s10915-022-01939-z`.

[45] J. E. Kittelsen, “Physics-informed neural networks for modeling and control of gas-lifted oil wells”, M.S. thesis, Norwegian University of Science and Technology, 2022.

[46] Z. W. Miao and Y. Chen, “Physics-informed neural networks method in high-dimensional integrable systems”, *Modern Physics Letters B*, vol. 36, no. 01, p. 2 150 531, 2022.

[47] V. Schäfer, “Generalization of pinns for various boundary and initial conditions”, M.S. thesis, University of Kaiserslautern, 2022.

[48] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um, *Physics-based deep learning*, Apr. 25, 2022. DOI: `10.48550/arXiv.2109.05237`. arXiv: `2109.05237[physics]`.

[49] S. Wang, X. Yu, and P. Perdikaris, “When and why pinns fail to train: A neural tangent kernel perspective”, *Journal of Computational Physics*, vol. 449, p. 110 768, 2022.

[50] Z. Xiang, W. Peng, X. Liu, and W. Yao, “Self-adaptive loss balanced physics-informed neural networks”, *Neurocomputing*, vol. 496, pp. 11–34, 2022.

[51] A. S. Bjørlo, “Deep equilibrium models and physics-informed neural networks for modeling and control of electric submersible pumps”, Unpublished pre-work, 2023.

[52] S. Monaco and D. Apiletti, “Training physics-informed neural networks: One learning to rule them all?”, *Results in Engineering*, vol. 18, p. 101 023, Jun. 1, 2023, ISSN: 2590-1230. DOI: `10.1016/j.rineng.2023.101023`.

[53] P. Stiller, R. Pausch, A. Debus, M. Bussmann, and N. Hoffmann, “Physics-informed neural networks for 3d laser propagation: High-frequency periodic functions and transfer learning”, *Available at SSRN 4353536*,

NTNU
Norwegian University of
Science and Technology

UFSC