Deep Equilibrium Models and Physics-Informed Neural Networks for Modeling and Control of Electric Submersible Pumps

Author
Aurora Sletnes Bjørlo

Supervisors
Lars Imsland
Eduardo Camponogara

January 9, 2023

Department of Engineering Cybernetics

NTNU

Acknowledgements

I would like to thank my supervisor Lars Struen Imsland for his feedback and support. I also would like to thank my supervisor at the Federal University of Santa Catarina, Eduardo Camponogara, for providing me with an exciting project and some direction in the relevant literature. Finally, I want to thank Ola Solli Grønningsæter for our collaboration in developing the electric submersible pump simulator presented in this thesis.

Aurora Sletnes Bjørlo

Hamar, January 9, 2023

Abstract

Many complex and nonlinear systems can be difficult to model and simulate, yet obtaining accurate models for such systems is greatly advantageous in many applications, including control purposes. In a nonlinear system such as the Electric Submersible Pump (ESP), where parameters may change over time, it is difficult to find the model of the system using analytical methods. In the absence of precise models, a data-driven solution may be used. The newly introduced Physics-Informed Neural Network (PINN) combines a priori knowledge of the physics of the system with data to either solve or discover the differential equations governing the system. Another recent advancement in deep learning is the proposal of Deep Equilibrium Models (DEQ). These deep learning models are based on viewing neural networks of infinite depth as a single-layer network defined implicitly, giving advantages such as less memory requirement and more explainability. This project aims at exploring developments in the research related to the two newly proposed deep learning architectures, PINN and DEQ. Furthermore, a simulator of an ESP is developed, and the use of PINN and DEQ for modeling and control of this system is explored theoretically. While some theoretical developments still remain, the results found in this work suggest that combining PINN and DEQ for modeling and control purposes for an ESP can be a promising future research field.

Keywords: Physics-Informed Neural Network. Deep Equilibrium Model. Electric Submersible Pump.

Contents

Acknowledgements **i**

Abstract **ii**

List of Tables **v**

List of Figures **vi**

Acronyms **vii**

1 Introduction **1**
1.1 Problem formulation and contributions 2
1.2 Report outline 2

2 Background **3**
2.1 Neural Networks 3
2.2 Deep Feedforward Neural Networks 4

3 Physics-Informed Neural Networks **6**
3.1 PINN formulation 7
3.2 Review of PINN for control 8
3.2.1 Physics-Informed Neural Nets-based Control 8
3.2.2 Controlling Chaos in Van Der Pol Dynamics 10
3.2.3 Optimal control of PDEs using physics-informed neural networks 11

4 Deep Equilibrium Models **13**
4.1 Theoretical background 13
4.1.1 Forward Pass 15
4.1.2 Backpropagation 15
4.1.3 Advantages and limitations 15
4.2 Review of DEQ research 16

4.2.1 Model and architecture 16
4.2.2 Convergence 17
4.2.3 Learning techniques 17
4.2.4 Application Areas 18
4.2.5 Physics-Informed Deep Equilibrium Models 19

5 Electric Submersible Pump **21**
5.1 Modeling of wells with ESP 22
5.1.1 Model equations and parameters 22
5.2 Modeling and control of ESP with PINN and DEQ 25

6 Electric Submersible Pump Simulator **27**
6.1 Implementation 27
6.2 Verification 27
6.2.1 Scenario 1: f = 60Hz and z = 100% 27
6.2.2 Scenario 2: f = 60Hz and z steps from 60% to 100% and back 28
6.2.3 Scenario 3: z = 60% and f steps from 60Hz to 70Hz and back 29
6.2.4 Scenario 4: f = 60Hz and z steps randomly between 50% to 100% 30
6.2.5 Results of verification 30

7 Discussion **32**
7.1 Physics-Informed Neural Networks 32
7.2 Deep Equilibrium Models 33
7.3 Electric Submersible Pump and the combination of PINN and DEQ for modeling and control 34

8 Conclusion and future work **36**
8.0.1 Future work 37

List of Tables

5.1 ESP model variables and characteristics 24
5.2 Well dimensions, ESP characteristics, and other constants 24
5.3 Parameters from fluid analysis and well tests, and parameters assumed to be constants 25
5.4 Polynomial coefficients 25

List of Figures

2.1 An illustration of the forward pass and backward pass of a neural network. 4
2.2 An illustration of a feedforward neural network. 5

3.1 An illustration of the general schema of a PINN model 7

4.1 An illustration of an infinite feedforward sequence model. 14
4.2 An illustration of an infinite weight-tied feedforward sequence model. 14
4.3 An illustration of a deep equilibrium model. 15

5.1 Model of an ESP lifted well, recreated based on [2] and [3]. 22
5.2 An illustration of a combined PINC and PIDEQ model in a control loop for an ESP model. 26

6.1 The simulation output with constant input of f = 60 Hz and z = 100%. 28
6.2 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. 29
6.3 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. 30
6.4 The simulation output when the valve opening is randomly changing within the limits of 60% and 100%. 31

Acronyms

CNN Convolutional Neural Network.

DAE Differential-Algebraic Equation.

DEQ Deep Equilibrium Model.

DNN Deep Neural Network.

ESP Electric Submersible Pump.

IVP Initial Value Problem.

MDP Markov Decision Process.

MLP Multilayer Perceptron.

ODE Ordinary Differential Equation.

PDE Partial Differential Equation.

PIDEQ Physics-Informed Deep Equilibrium Model.

PIDOC Physics-Informed Deep Operator Control.

PINC Physics-Informed Neural nets-based Control.

PINN Physics-Informed Neural Network.

RNN Recurrent Neural Network.

Chapter 1

Introduction

The deep learning approach to machine learning has existed since the 1940s [6]. It originally gained inspiration from nature. More specifically, from biological brains that are built up with billions of neurons working together to process information. This inspiration is reflected in the term neural network. Deep neural networks consist of many layers of neurons. While such networks are extremely powerful and have shown great success on multiple classes of problems, they also come with challenges.

One of the biggest issues deep neural networks are facing today is their memory usage [12]. The nature of these networks is based on a large number of neurons, as well as many layers. To be able to train such networks, it is vital to store the weights of the network, as well as the computed values of the activation functions in the forward pass. As the number of layers increases in a deep neural network, the required memory grows fast.

A novel approach to deep learning, that does not suffer from this issue is the Deep Equilibrium Model (DEQ), which was proposed by Bai et al. in [12]. The model is based upon a certain class of deep neural network models called weight-tied deep sequence models. If one were to use such a model with an infinite amount of layers, it would eventually converge to an equilibrium. The idea behind the deep equilibrium model is to find this equilibrium point directly, rather than going through infinitely many layers. This can be viewed as the model having just one layer, which gives the benefit of requiring only constant memory.

Another problem that often arises when creating machine learning models, is the lack of data. This is especially challenging when working with complex systems, whether complex engineering systems or physiological and biological systems [13]. For such systems, it may be costly, or time-consuming to gather large amounts of data. However, training deep networks with an insufficient amount of data will lead to uncertain models that lack robustness. Luckily, for many such systems, there exist physical rules that they must obey, and that can be used during training.

This is the idea behind Physics-Informed Neural Network (PINN). These networks use the prior information of the physical laws governing the system to constrain the space of possible solutions. Although this does not generate more data, it can be seen as amplifying the information contained in the existing data set. This is of great benefit in a variety of applications. Recently, the use of such networks for control purposes has gained attention.

Both deep equilibrium models and physics-informed neural networks show great promise for a number of applications. Recent developments in these areas will be explored and detailed further in this work. In addition, another interesting field of research is that of combining these two techniques, as done in the proposed Physics-Informed Deep Equilibrium Model (PIDEQ) from [34]. The testing of PIDEQ on a complex physical system like the Electric Submersible Pump (ESP) remains. In such a case, the PIDEQ could potentially be combined with the recent developments of PINN for control to create a model of the ESP suitable for control purposes.

1.1 Problem formulation and contributions

This work aims at exploring and reviewing the recent advances within the two fields of Deep Equilibrium Models and Physics-Informed Neural Networks, as well as to introduce a simulated model of an Electric Submersible Pump. The main objective of this is to create a theoretical background for future research on the usage of DEQs combined with PINN for the modeling and control of physical systems such as the ESP.

1.2 Report outline

The paper will be divided into three main parts:

- A review of recent research on the topic of Physics-Informed Neural Networks, focusing on developments related to PINN for control purposes.
- A review of current research related to Deep Equilibrium Models, with a focus on the proposed models and architectures, convergence, learning techniques, and applications. Special attention will be given to the recently proposed combination of PINN and DEQ
- Implementation of a simulator of an Electronic Submersible Pump, based on a set of nonlinear differential equations. Verification of the simulator. A proposed theoretical model for using PINN and DEQ in combination for the modeling and control of an ESP system.

Chapter 2

Background

2.1 Neural Networks

Neural networks are a class of machine learning models that have been able to achieve state-of-the-art performance in a large variety of applications, including, but not limited to, speech and language processing [11], medical diagnosis [4], and finance [10]. Much research has been conducted on developing high-performing models of this class. However, as the nature of neural networks is complex, much is yet to be learned about how the parameters of the model should be chosen in order to provide a model with the desired properties.

Two important concepts related to neural networks and the training of such models are the forward pass and backward pass. An illustration of these concepts is given in Figure 2.1. The forward pass is the process of calculating the output of a given input. This is done by finding the values of the different neurons. During the training process, the loss can then be calculated from the output and the known true value. The aim is to minimize this loss, by adjusting the different weights of the network. In order to do so, the concept of backward pass or backpropagation is important. The goal of this step is to go through the different layers backward, starting from the output, and to update the weights at each layer in a way so that the loss will decrease. This can be achieved using the gradient descent algorithm.

The majority of deep neural networks consist of multiple layers, where the number of layers, L, is said to be the depth of the network. Each layer contains a set of neurons, where the width of a layer is the number of neurons it contains. While both of these can be chosen by the model designer, research indicates that there are trade-offs when going with a deeper or wider network.

Research on how different properties of such networks affect the models beyond their performance was conducted in [16]. Here, the authors found that although deep and wide models may give similar accuracy, they have distinct error patterns

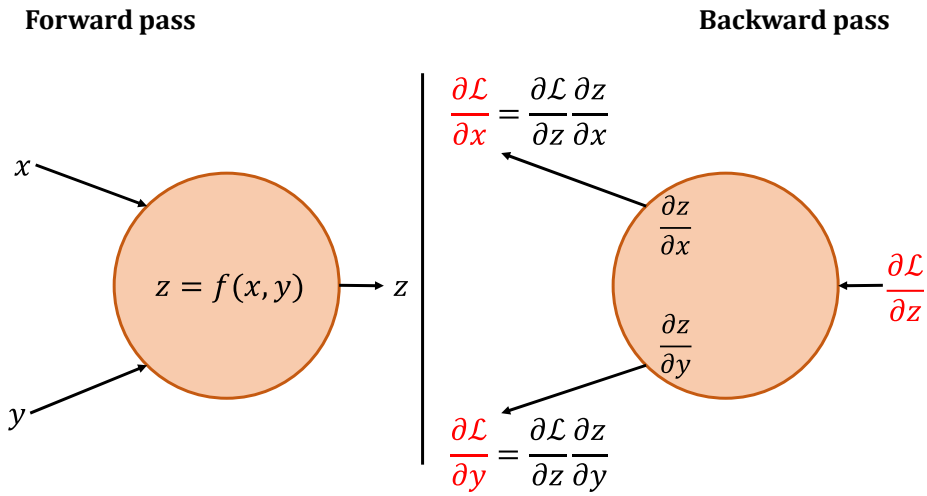

Figure 2.1: An illustration of the forward pass and backward pass of a neural network.

and variations across classes.

Efforts to determine the effect of depth on the upper bound of the expected error showed that increasing the depth of a neural network leads to contradicting effects [7]. Thus, increasing the depth arbitrarily might not necessarily yield a better performance of the network. In [8] they conclude that narrow networks with a size that is larger than the polynomial bound by a constant can approximate wide networks with high accuracy and that this might indicate that depth is more expressive than width for ReLU networks at least.

2.2 Deep Feedforward Neural Networks

Deep feedforward neural networks, or deep feedforward networks, feedforward neural networks, or Multilayer Perceptron (MLP) have a long history. An illustration of such a network is given in Figure 2.2. These models date back to 1965, according to [5]. Today, they form the basis of many well-known deep-learning model classes, like Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

A deep feedforward network is a network where information only can move in one direction, forward, meaning the information goes from the input neurons through each layer sequentially (if there is more than one) and then to the output neurons. There are no cycles within the network, and the information from the output is not fed back to the model.

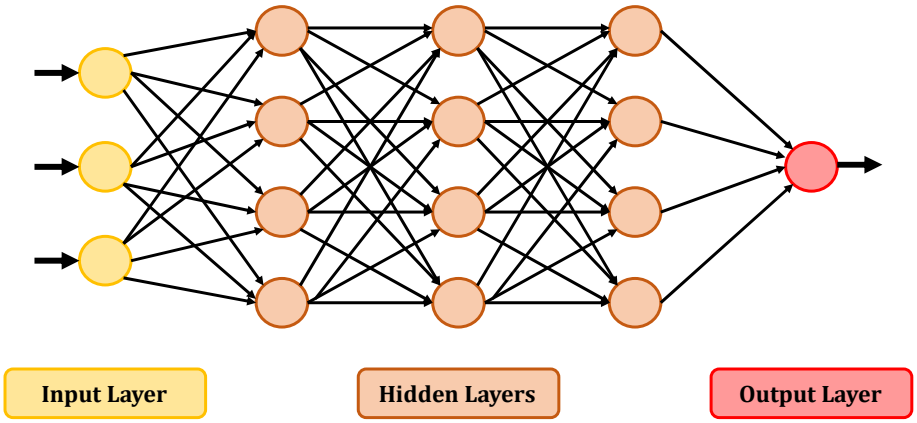

Figure 2.2: An illustration of a feedforward neural network.

The goal of such a network is to approximate a function [6]. This is done by defining a mapping from input to output that depends on the input value, as well as a set of parameters. By adjusting these parameters through the learning process, the network achieves the best possible estimate of the approximate function that it can with the given data. In fact, as shown in [1] a feedforward network that contains just one single hidden layer can be used to approximate any measurable function to the desired degree of accuracy. Thus, deep feedforward neural networks are called universal approximators, meaning they can approximate any function universally.

Chapter 3

Physics-Informed Neural Networks

With the growth and improvements of deep learning models, their potential in more areas is being explored. This includes complex application areas, such as physical, biological, and engineering systems. However, a key issue when designing and implementing such systems is obtaining enough data to ensure the reliability of the models in question. In many situations, it can be costly, time-consuming, or practically impossible to obtain large amounts of data.

However, many of these systems are bound by the laws of physics. Thus, by taking these into account, the span of possible models is constrained to only contain those satisfying these laws. An illustration of the architecture of a typical PINN is provided in Figure 3.1. These networks are well-suited to solve forward problems, predicting data given a system, as well as inverse problems, determining the underlying system given its data.

There are several advantages to physics-informed neural networks, as highlighted in [20]. PINNs are more effective for ill-posed and inverse problems than other deep learning approaches. In addition, these networks can be used even in cases where both the model and the data are imperfect, and there exist formulations of these models suitable to quantify related uncertainties. Furthermore, they can also be used for a greater understanding of deep learning mechanisms.

Although physics-informed neural networks show great promise, they are still facing a number of challenges. As pointed out in [31], several new PINN architectures have been proposed to overcome the limitations of the original PINN. However, these novel architectures have introduced new sets of constraints that ought to be addressed. In [27] unresolved issues of the original PINN formulation, both related to theoretical considerations and implementation issues, are highlighted. In addition, new computational frameworks and algorithms may have to be developed in order to solve complex PINN models [20].

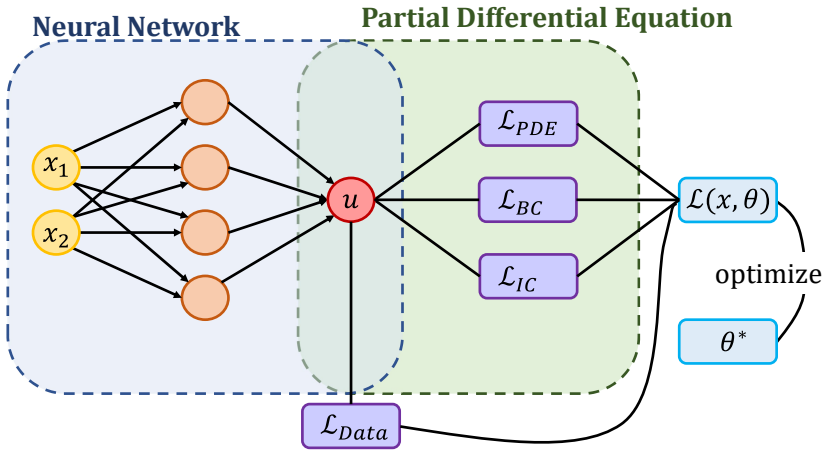

Figure 3.1: An illustration of the general schema of a PINN model

3.1 PINN formulation

The idea behind the original PINN formulation, is to use a fully connected Deep Neural Network (DNN) architecture to find the solution of a dynamical process, y. The output of this network is on the form:

$$\mathbf{y}(x,t) = f_{\mathbf{w}}(x,t) \tag{3.1}$$

where $x \in \Omega$ is a spatial input, and $t \in [0,T]$ is a temporal input. The function $f_{\mathbf{w}}$ represents the mapping obtained by a deep neural network, where $\mathbf{w}$ are the network parameters or weights. These are optimized using the loss function during the training of the PINN model. Consider the following set of equations where the first represents a standard partial differential equation, the second the boundary conditions of the equation, and the last corresponds to the initial conditions:

$$\delta_t y(x,t) + \mathcal{N}_x[y(x,t)] = 0 \qquad ,x \in \Omega, \quad t \in [0,T] \tag{3.2a}$$

$$\mathcal{B}[y(x,t)] = g(x,t) \qquad ,x \in \delta\Omega, \quad t \in [0,T] \tag{3.2b}$$

$$y(x,0) = y_0(x) \qquad ,x \in \Omega \tag{3.2c}$$

Here, $\mathcal{N}_x[\cdot]$ is a general differential operator, $\mathcal{B}[\cdot]$ is the boundary operator, and y_o is the initial value of y. Furthermore, define the left-hand side of Equation 3.2a as:

$$\mathcal{F}(y) = \delta_t y + \mathcal{N}_x[y] \tag{3.3}$$

A general loss function for this system is on the form:

$$\mathcal{L} = \mathcal{L}_{Data} + \mathcal{L}_{PDE} + \mathcal{L}_{BC} + \mathcal{L}_{IC} \tag{3.4}$$

The first term of this equation represents the loss related to the data, if it is available. This term is typically present when PINNs are used to solve inverse problems, as these involve known data. The second term is related to the loss of the PDE equation, the third is the loss from the boundary condition, and the fourth the loss from the initial condition. These can be defined as follows:

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left| y(t_d^j, x_d^j) - \hat{y}^j \right|^2 \tag{3.5a}$$

$$\mathcal{L}_{PDE} = \frac{1}{N_F} \sum_{k=1}^{N_F} \left| \mathcal{F}(y(t_f^k, x_f^k)) \right|^2 \tag{3.5b}$$

$$\mathcal{L}_{BC} = \frac{1}{N_B} \sum_{l=1}^{N_B} \left| \mathcal{B}[y(t_b^l, x_b^l)] - g(t_b^l, x_b^l) \right|^2 \tag{3.5c}$$

$$\mathcal{L}_{IC} = \frac{1}{N_I} \sum_{m=1}^{N_I} \left| y(0, x_i^m) - y_0^i \right|^2 \tag{3.5d}$$

Here, N_D and N_F represent the number of training data samples and collocation points respectively. The data points are denoted as $\{t_d^j, x_d^j\}_{j=1}^{N_D}$, while $\{t_f^k, x_f^k\}_{k=1}^{N_F}$ represents the collocation points. The number of boundary points is N_B, and $\{t_b^l, x_b^l\}_{l=1}^{N_B}$ denotes these points. The initial condition points are expressed as $\{0, x_i^m\}_{m=1}^{N_I}$, where N_I is the number of such points.

3.2 Review of PINN for control

A recent line of research related to PINN, in line with the recommendation from [27] is the use of PINN for control systems. Several architectures have been proposed and tested on different systems. Some of them, like the work proposed in [22] and [28], are based upon modeling the transition function as a Markov Decision Process (MDP). This yields a discretized loss function. In the following, only control schemes where the physical laws can be included in the continuous form are considered. This section aims to highlight the underlying structure of these systems, as well as their advantages and disadvantages.

3.2.1 Physics-Informed Neural Nets-based Control

With the increasing popularity of the versatile PINN models, the extension of these models to control problems is a natural development. However, the original architecture of PINN is not suited for control purposes directly. This is mainly due to two factors, the first being that the original PINN does not allow the control inputs.

The second is the fact that they are unable to give accurate long-range predictions. The latter problem has been addressed to some extent in recent literature like [15]. However, both of these issues have to be remediated if the PINN is to be used in control systems.

As a solution, [23] proposes the Physics-Informed Neural nets-based Control (PINC), which is a PINN-based architecture that has incorporated solutions to both of the aforementioned issues. Thus, it is applicable to control problems and able to predict long-range time horizons. The control parameters are added by modifying the input of the neural network to include the initial state and the control input. The output of the PINC network is therefore on the form:

$$\mathbf{y}(t) = f_{\mathbf{w}}(t, \mathbf{y}(0), \mathbf{u}) \tag{3.6}$$

where $t \in [0, T]$ is a temporal input, $\mathbf{w}$ is the network parameters, $\mathbf{y}(0)$ is the initial condition, and $\mathbf{u}$ is the control input for $t \in [0, T]$.

The structure of the PINC framework is as follows. It consists of an augmented network that is able to take the initial state value as well as the control action as inputs. To solve the control problem, domain decomposition is used. For each interval of the domain, the solution to the ODE can be computed using fixed values for the state input and the control action. The output of the network is the sequential combination of all intervals, where the terminal state of a network is used as the initial state for the subsequent. To achieve this, the output at a given domain k can be defined as a function of the previous domain, $k - 1$:

$$\mathbf{y}[k] = f_{\mathbf{w}}(t_k, \mathbf{y}[k-1], \mathbf{u}[k]) \tag{3.7}$$

In [23] nonlinear ODEs are considered on the following form:

$$\mathcal{F}(y) = \delta_t \mathbf{y} + \mathcal{N}[\mathbf{y}] = 0, \qquad t \in [0, T] \tag{3.8}$$

The loss function of the network is defined as:

$$\mathcal{L} = \mathcal{L}_{Data} + \mathcal{L}_{ODE} \tag{3.9}$$

Letting N_D and N_F represent the number of training data samples and collocation points respectively, $\{y_d^j, y(0)_d^j, u_d^j\}_{j=1}^{N_D}$ the training data points, and $\{y_f^k, y(0)_f^k, u_f^k\}_{k=1}^{N_F}$ the collocation points, the $\mathcal{L}_{Data}$ and $\mathcal{L}_{ODE}$ can be expressed as:

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left| y(t_d^j, y(0)_d^j, u_d^j) - \hat{y}^j \right|^2 \tag{3.10a}$$

$$\mathcal{L}_{ODE} = \frac{1}{N_F} \sum_{k=1}^{N_F} \left| \mathcal{F}(y(t_f^k, y(0)_f^k, u_f^k)) \right|^2 \tag{3.10b}$$

The application of this method to complex control systems is already a topic of research, and in [33] it is tested on a tracking problem for a complex mechanical system. It has also been used in the context of an unstable gas-lifted oil well in [30].

The PINC mainly has two advantages when used in control systems. The first is that it is able to identify a system using not only the collected data but also the governing physical equations. The second is its ability to speed up the simulation of differential equations compared to numerical solution methods, which is a great advantage in real-time control applications. While the PINC shows promise, it does not come without limitations. The training time for such a network is long. In addition, it comes without guarantees, as it does not in its original form take into account the possibility of discontinuities or the possibility of invalid arguments. The current framework is also only developed for Ordinary Differential Equation (ODE), thus it is uncertain how it extends to other equation types like Partial Differential Equation (PDE) and Differential-Algebraic Equation (DAE).

3.2.2 Controlling Chaos in Van Der Pol Dynamics

In [38] the Physics-Informed Deep Operator Control (PIDOC) was proposed as a framework for the control of nonlinear dynamics using PINN. The PIDOC architecture encodes the control signal, as well as the initial position of the system into the loss function of the PINN. This way, the network output is forced to follow the desired control trajectory. The resulting loss function of the network can be defined as:

$$\mathcal{L} = \mathcal{L}_{Data} + \mathcal{L}_{IC} + \lambda \mathcal{L}_{Control} \tag{3.11}$$

where λ controls the weight of the control signal.

Defining $\tilde{y}$ as the desired trajectory from the controller, these loss functions can be defined as:

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left| y(t_d^j, x_d^j) - \hat{y}^j \right|^2 \tag{3.12a}$$

$$\mathcal{L}_{IC} = \frac{1}{N_I} \sum_{k=1}^{N_I} \left| y(0, x_i^k) - \tilde{y}_0^k \right|^2 \tag{3.12b}$$

$$\mathcal{L}_{Control} = \frac{1}{N_C} \sum_{l=1}^{N_C} \left| \left(\frac{d^2 \tilde{y}^l}{dt^2} - \frac{d^2 y(t_c^l, x_c^l)}{dt^2} \right) + \left(\tilde{y}^l - y(t_c^l, x_c^l) \right) \right|^2 \tag{3.12c}$$

where N_D, N_I, and N_C represent the number of training data samples, initial position samples, and control signal samples respectively. The corresponding points are denoted as $\{t_d^j, x_d^j\}_{j=1}^{N_D}$, $\{0, x_i^k\}_{k=1}^{N_I}$, and $\{t_c^l, x_c^l\}_{l=1}^{N_C}$ respectively.

Initial experiments with the PIDOC show promising results. It has been tested on benchmark problems, such as the Van Der Pol dynamics, and has successfully been able to control the dynamics of these systems. Furthermore, results indicate that changes in the depth and width of the underlying neural network only have a limited effect on the convergence of these systems. This opens up the possibility of using a wide range of architectures, optimizing for considerations such as training efficiency and memory usage.

A limitation of the originally proposed PIDOC is its performance for systems that are highly nonlinear. In these cases, the model is not able to produce results with satisfying accuracy. Furthermore, the performance of the architecture has only been tested on a limited set of systems. Extending the application of this network to different types of systems is a future area of research, as is also comparing this to deterministic control methods.

3.2.3 Optimal control of PDEs using physics-informed neural networks

The idea of extending the PINN framework for control purposes is also explored in [40]. There, the PINN framework is extended to include PDE-constrained optimal control problems. The framework is accompanied by a set of guidelines, both for the selection and creation of a suitable PINN model and for the tuning of the weight parameters incorporated in the loss function.

The proposed framework comprises two feedforward neural networks, one to solve the forward problem, and one to approximate the control variable. The partial differential equation with boundary and initial conditions can be expressed as follows:

$$\mathcal{F}[\mathbf{y}(t,x);\mathbf{c}_v(t,x)] = 0 \qquad ,x \in \Omega, \quad t \in [0,T] \tag{3.13a}$$

$$\mathcal{B}[\mathbf{y}(t,x);\mathbf{c}_b(t,x)] = 0 \qquad ,x \in \delta\Omega, \quad t \in [0,T] \tag{3.13b}$$

$$\mathcal{I}[\mathbf{y}(0,x);\mathbf{c}_0(0,x)] = 0 \qquad ,x \in \Omega \tag{3.13c}$$

where $\mathbf{c}_v$, $\mathbf{c}_b$ and $\mathbf{c}_0$ represent the volume, boundary and initial control vectors respectively.

Furthermore, the optimal control $\mathbf{c}^*$ can be defined as:

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \mathcal{C}(\mathbf{y},\mathbf{c}) \tag{3.14}$$

where $\mathbf{c}^*$ where meets the conditions defined by Equation 3.13, and $\mathcal{C}$ denotes a cost function.

The loss function corresponding to the optimal control problem comprises loss functions related to the PDE, the boundary conditions, and the initial conditions. In addition, it also contains a term corresponding to the cost function. Consequently, the total loss can be expressed as:

$$\mathcal{L}(\mathbf{w}_y, \mathbf{w}_c) = \mathcal{L}_{PDE} + \lambda_b \mathcal{L}_{BC} + \lambda_i \mathcal{L}_{IC} + \lambda_{\mathcal{C}} \mathcal{L}_{\mathcal{C}} \tag{3.15}$$

where $\mathbf{w}_y$ represents the weights of the network solving the forward problem, and $\mathbf{w}_c$ the weights for the network finding the optimal control. The multipliers λ_b, λ_0, and $\lambda_{\mathcal{C}}$ can be used to determine the relative importance of the boundary condition, initial condition, and cost function specifically. The loss functions can further be defined as:

$$\mathcal{L}_{PDE} = \frac{1}{N_F} \sum_{j=1}^{N_F} \left| \mathcal{F}[y(t_f^j, x_f^j); c(t_f^j, x_f^j)] \right|^2 \tag{3.16a}$$

$$\mathcal{L}_{IC} = \frac{1}{N_I} \sum_{k=1}^{N_I} \left| \mathcal{I}[y(0, x_i^k)] \right|^2 \tag{3.16b}$$

$$\mathcal{L}_{BC} = \frac{1}{N_B} \sum_{l=1}^{N_B} \left| \mathcal{B}[y(t_b^l, x_b^l)] \right|^2 \tag{3.16c}$$

Here, N_F, N_I, and N_B are the number of collocation points, initial points, and boundary points. Moreover, $\{t_f^j, x_f^j\}_{j=1}^{N_F}$, $\{0, x_i^k\}_{k=1}^{N_I}$, and $\{t_b^l, x_b^l\}_{l=1}^{N_B}$ denote the corresponding points.

The resulting PINN framework is compared to standard approaches for solving PDE-constrained optimal control problems in [40]. For the problems explored, both methods are able to find an optimal solution with comparable cost objective values. This suggests that the PINN framework can be a viable option for solving optimal control problems while providing the benefit of easier implementation and less time consumption than current models.

Chapter 4

Deep Equilibrium Models

Deep learning networks have become widely used in recent years. Yet, in many aspects little is still known about what makes these models perform well. Much research has been devoted to aspects regarding the depth and width of such networks ([16], [7], [8]), aiming at finding how these hyper-parameters may be tuned for optimal network performance. While increasing these hyper-parameters might render better-performing models, it can come at the cost of increased memory consumption and reduced model explainability.

In a recent line of research, implicit models in the context of deep learning have been investigated ([18], [12]). A motivation behind this work is that deep sequence models, if the number of layers goes toward infinity, converge to a fixed point. Thus, the model can be defined implicitly by a fixed-point equation. This provides a network with a simplified notation that has less memory requirement than similarly performing neural networks.

4.1 Theoretical background

The deep equilibrium model, as proposed by Bai et al. in [12], is a class of deep learning models. The development of these models is based on the concept of deep feedforward sequence models, illustrated in Figure 4.1. It can be described by the general equation:

$$z^{[i+1]} = f_\theta^{[i]}(z^{[i]}; x), \quad i = 0, 1, ..., L-1, \quad z^{[0]} = 0, \quad G(x) \equiv z^{[L]} \tag{4.1}$$

For these networks, there is a subclass called weight-tied deep feedforward sequence models. Here, the "weight" of each layer is tied. This is equivalent to saying that each layer applies the same weight, or function, to the input. Thus, in Equation 4.2, the function $f_\theta^{[i]}$ can be replaced by a single function f_θ, equal for every layer. An illustration of such a network is given in Figure 4.2. This can be expressed as:

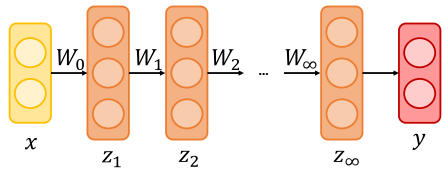

Figure 4.1: An illustration of an infinite feedforward sequence model.

$$z^{[i+1]} = f_\theta(z^{[i]}; x), \quad i = 0, 1, ..., L-1 \tag{4.2}$$

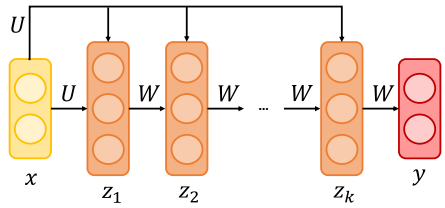

Figure 4.2: An illustration of an infinite weight-tied feedforward sequence model.

Assuming one has the possibility of creating such a model with infinitely many layers, the resulting model could converge to an equilibrium point. Figure 4.3 shows a general deep equilibrium model. The idea behind the deep equilibrium model is to directly find this equilibrium point, rather than iterating through all these layers. This corresponds to solving the equation:

$$\lim_{i\to\infty} z^{[i]} = \lim_{i\to\infty} f_\theta(z^{[i]}; x) \equiv f_\theta(z^*; x) = z^* \tag{4.3}$$

The output of the model is thus z^*.

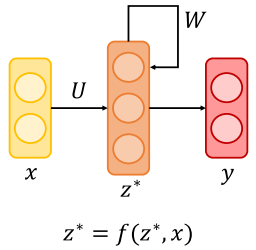

Figure 4.3: An illustration of a deep equilibrium model.

4.1.1 Forward Pass

The output of DEQ is the equilibrium point. Therefore, the forward pass of the network simply consists of solving the equilibrium equation. This can be done using a number of techniques. It can be solved considering the converging infinite sequence. It can also be found by rewriting. In this case, one can apply any root-finding methods, such as Newton's method or quasi-Newton methods.

4.1.2 Backpropagation

The challenge of performing the backward pass is differentiating through the fixed equilibrium point. In deep neural networks, backpropagation can be done explicitly using the operations from the forward pass. However, in DEQ no such operations exist as the equilibrium is solved directly. The implicit function theorem ensures that the differentiation of the steady state of an implicit function, such as the equilibrium equation, can be done directly without going back through an iterative process. This means that a backpropagation scheme based on this theorem does not rely on storing intermediate values, so the memory use is constant. Using the implicit function theorem, the gradient of the loss function can be expressed with respect to the different parameters $(\cdot)$ as follows:

$$\frac{\delta l}{\delta(\cdot)} = -\frac{\delta l}{\delta z^*}(J_{g_\theta}^{-1}|_{z^*})\frac{\delta f_\theta(z^*;x)}{\delta(\cdot)} \tag{4.4}$$

where l is the loss, and $J_{g_\theta}^{-1}|_{z^*}$ is the inverse of the Jacobian of g_θ evaluated at z^*.

4.1.3 Advantages and limitations

The initially proposed deep equilibrium model has a set of advantages. First of all, it allows for an intuitive way to model implicit relations, This modeling technique allows for more expressive models. This modeling framework can be used to describe both traditional deep learning models of any depth, as well as equilibrium

models [12]. Another advantage of these models is that this can all be expressed with a single equilibrium layer (where the width does not grow infinitely). These models are also much more memory efficient than standard feedforward networks, as they do not require the intermediate storage of the values of weights and activation functions.

However, the concepts of implicit deep learning and deep equilibrium models are fairly new, and thus bring with them a number of challenges. One of the main drawbacks of the deep equilibrium models compared to their more traditional deep learning counterparts is their slowness, both for training and inference. Efforts have been made to speed up the models, amongst others by using approximations for the inverse jacobian in the backpropagation step.

4.2 Review of DEQ research

While the original DEQ formulation shows great promise, it also leaves room for improvement in various areas. Recent developments related to these models can be roughly grouped into four categories; model and architecture, convergence and certification, learning techniques and application areas.

4.2.1 Model and architecture

When choosing a deep learning model for solving a problem, requirements tend to differ depending on the situation. This also applies to deep equilibrium models and has inspired the creation of several new models and architectures possessing the desired qualities. Examples of this include proposed models such as the MDEQ, skip DEQ, and monDEQ.

In architectures used for solving visual problems like image classification and segmentation, the layers in traditional deep neural networks enable the representation of images at different spatial scales. As DEQs are composed of only one layer, it is not possible for these networks to gain such representations the same way. The multiscale deep equilibrium model, MDEQ for short, proposed in [14], is a variation of the original DEQ that solves this issue by maintaining multiple spatial resolutions simultaneously. In fact, the exact same MDEQ model can be applied for both classification and segmentation tasks, due to its ability to maintain the different resolutions.

In [35] it is pointed out that although DEQs provide great adaptivity, this does not come without a cost. It makes it difficult to predict how computationally intensive the model will be, and several DEQ models do indeed tend to have a greater computational expense than explicit models. The proposed solution is to combine the two techniques, aiming at leveraging the benefits of both. The Skip DEQ trains both an explicit and an implicit correction. This novel architecture renders a model that has a decreased training and prediction time, compared to the original DEQ formulation.

As previously mentioned, the original DEQ formulation does not provide guarantees in terms of whether a solution to the particular deep equilibrium problem exists,

and if it does, whether this solution is unique or not. Thus, a novel framework based upon the original DEQ was proposed in [24] for use in situations where such guarantees are desired or necessary. The "Monotone Operator Equilibrium Networks", or MonDEQs for short, use monotone operators to provide a solution with guaranteed stability. Furthermore, combining monotone operator theory and interval bound propagation, the IBP-MonDEQ has been introduced in [37]. This layer is able to certify the robustness of the DEQ model.

4.2.2 Convergence

Another area of interest related to the theory of deep equilibrium models is whether these models have guarantees for global convergence. For the training process of a DEQ to be stable, a necessary condition is the well-posedness of the implicit mapping. The question of convergence was studied for linear DEQ models in [21]. Here, it was shown that linear convergence to a global minimum could be achieved for these models. This question was investigated also for non-linear models in [32]. Here, over-parameterized deep equilibrium models using the ReLU activation were studied. It was shown that for such models there exists a unique equilibrium point during the training of the model, and convergence to a globally optimal solution is guaranteed.

4.2.3 Learning techniques

The biggest challenge deep equilibrium models are currently facing is their speed. While they have proven to provide similar results as state-of-the-art deep neural networks on benchmark problems using very little memory, they can be 3-4 times slower in training and 1.7-2 times slower in inference ([12], [14]) than their equivalent-performing deep neural network counterparts. This inefficiency is a great weakness and is currently a vital part of ongoing research in the domain. Several techniques have been proposed to increase the efficiency of DEQ, such as regularization, new neural solvers, and pruning amongst others.

Implicit models can, in addition to overfitting the dataset, become unstable. Regularization is a technique that can be used in the training process to prevent these issues. A regularization scheme adapted to DEQs was proposed in [17]. The idea behind this technique is to regularize an approximation of the Jacobian of the layer at the point of convergence. By doing this, the learning process of the DEQ is stabilized. This leads to a great increase in the speed of the DEQ model, rendering approximately the same speed and performance as traditional deep neural networks.

Another option to increase the speed of training is to use better solvers for finding the equilibrium point. The design of DEQ models allows for decoupling the structure of a layer from the method used to compute the fixed-point. Instead of using a general purpose solver, one can leverage this decoupling and create a custom solver for finding fixed-points of this network. This is what was done in [26], where a Neural Deep Equilibrium Solver was introduced. This solver is a variation of the Anderson acceleration that has better initial guess and parameterized

iterations. DEQs using this new solver achieve better inference speed than DEQs using traditional solvers like Anderson.

In classical machine learning pruning is a widely used technique for obtaining smaller and faster networks. The idea behind pruning is removing the neurons of the network that contribute the least. The potential of pruning as a technique to reduce both the training and inference cost of deep equilibrium models, is shown in [36]. Methods for pruning at initialization are used, both for DEQ and monDEQ models.

4.2.4 Application Areas

Most of the research on DEQs has been focused on testing the models on familiar DNN problems such as image segmentation [39]. The DEQs perform well on these issues, producing outcomes comparable to those of cutting-edge networks, however they are but one of many models that are available. On the other hand, using the DEQs to tackle challenging issues where conventional deep learning hasn't yet produced satisfactory results would demonstrate their usefulness. This encourages researchers to test the framework on a wider range of issues in order to comprehend its potential. Three application areas that have been explored in recent literature with promising results are: joint inference and optimization, implicit representations and optical flow estimation.

In deep learning, tasks such as adversarial attacks and latent space optimization often involve optimization over the inputs to the deep network. Such optimization can be computationally expensive, as it requires a forward and backward pass through the network for each gradient step. In [19] it is shown that deep equilibrium models can be well suited to solve this type of problem. Gradient-based optimization can be viewed as a fixed point iteration, so finding the solution to this problem can be viewed as finding the joint equilibrium of the gradient descent procedure and the DEQ model. Thus, using an augmented version of the DEQ these two problems can be solved simultaneously.

Implicit representations is a line of research within deep learning where the goal is for the network to learn a continuous representation of high-frequency data such as images and the geometry of a scene. For example, such models have been used to learn continuous spatial-temporal image representations. Many of the key qualities of DEQs make them good models for implicit representations, and in [29] it is shown that DEQs actually outperform existing implicit representation models. As implicit representations typically train with large batch sizes, they can benefit well from the reduction in memory consumption that the DEQs offer. Furthermore, past fixed points can be reused in the training process as initialization points, giving rise to a more efficient training approach.

Optical flow estimation is a technique in the computer vision field that involves tracking the movement of pixels between frames in a video. It allows the computer to determine the velocity at which each pixel is moving and can be used for a variety of tasks such as motion detection, tracking objects, and predicting future positions of objects in the frame. This information can be used to improve the

performance of various computer vision algorithms, such as object recognition and scene segmentation. Although some classical deep learning models have been able to achieve good results, these models tend to scale poorly when the image size or number of iterations increases. The DEQ model proposed in [25] to solve this problem has been shown to outperform current approaches and achieve a state-of-the-art solution on realistic optical flow datasets. Although the training of this model is slightly more complex than that of the originally proposed method, it improved performance as well as reduced computational and memory footprint, making it advantageous in comparison to current methods.

4.2.5 Physics-Informed Deep Equilibrium Models

Another possible application area of the DEQ is that of using DEQ in combination with the PINN model. This is the topic of research in [34], where PIDEQ, a physics-informed deep equilibrium model, is proposed. The aim was to investigate whether or not the DEQ could be trained using a physics-informed approach for solving the Initial Value Problem (IVP) of an ODE. The PIDEQ is defined by the following equations:

$$z^* = f_\theta(t, z^*) \tag{4.5a}$$
$$f_\theta(t, z) = \tanh(Az + ta + b) \tag{4.5b}$$

where the output of the model is defined as Cz^*, and the parameters of f_θ are a vectorization of the the matrices and vectors so $\theta = (A, C, a, b)$.

The loss function used for training a PIDEQ can be defined as:

$$\mathcal{L} = \mathcal{L}_{PDE} + \mathcal{L}_{BC} + \mathcal{L}_{IC} + \left\| \frac{df_\theta}{dz} \right\|_F \tag{4.6}$$

where $\left\| \frac{df_\theta}{dz} \right\|_F$ is the Frobenius norm of the Jacobian.

The results indicate that the PIDEQ can indeed be used to solve such problems, although it achieved a larger approximation error and had slower training than the PINN when doing so. However, this work only considered the IVP of the Van der Pol oscillator problem. Thus, in order to properly be able to evaluate the performance of the PIDEQ compared to the PINN, more experiments are required. In particular, for complex problems requiring deeper networks, such a comparison would be of interest.

In addition to this, several aspects of the model could be considered to improve its performance, most notably the speed of the training. The PIDEQ is based on the original DEQ formulation, with Jacobian regularization as proposed in [17]. While this regularization has been shown to improve the speed of the DEQ, the PIDEQ sill trains slower than a comparable PINN model. Bypassing this limitation would open up more possible applications for the PIDEQ.

Finally, another area of future research indicated by [34], is the use of such networks for control problems. As highlighted in Section 3.2, the use of PINN for control has gained attention recently, as such models provide better predictions for complex and sparse systems due to their incorporation of the physical information of the system. One of the advantages of the DEQ, and therefore also of PIDEQ, is that it requires fewer parameters than deep neural networks. This provides them with the advantage of being more explainable, which can lead to a more robust controller design.

Chapter 5

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 [9]. 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 [9] states, the conventional installation of an ESP system depicted in Figure 5.1, 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 [2]. 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.

5.1 Modeling of wells with ESP

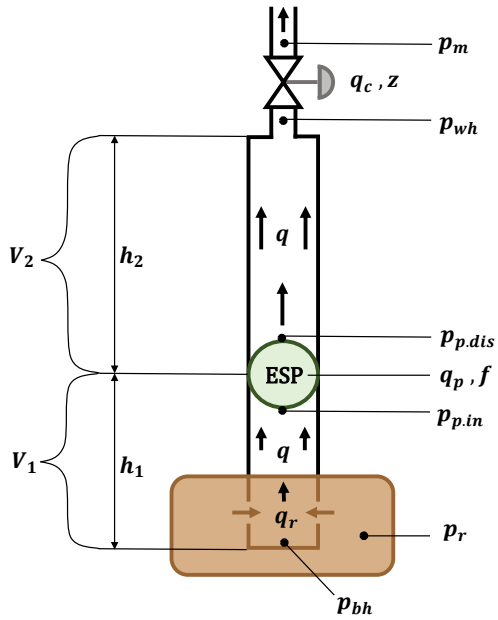

Figure 5.1: Model of an ESP lifted well, recreated based on [2] and [3].

5.1.1 Model equations and parameters

This model is based on the model of an ESP-lifted well presented in [2], 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):

$$\dot{p}_{bh} = \frac{\beta}{V_1}(q_r - q) \tag{5.1a}$$

$$\dot{p}_{wh} = \frac{\beta}{V_2}(q - q_c) \tag{5.1b}$$

$$\dot{q} = \frac{1}{M}(p_{bh} - p_{wh} - \rho g h_w - \Delta p_f + \Delta p_p) \tag{5.1c}$$

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

Flow:

$$q_r = PI(p_r - p_{bh}) \tag{5.2a}$$

$$q_c = C_c\sqrt{p_{wh} - p_m}z \tag{5.2b}$$

Friction:

$$\Delta p_f = F_1 + F_2 \tag{5.3a}$$

$$F_i = 0.158 \cdot \frac{\rho L_i q^2}{DA^2} \cdot \left(\frac{\mu}{\rho D q}\right)^{\frac{1}{4}} \tag{5.3b}$$

ESP:

$$\Delta p_p = \rho g H \tag{5.4a}$$

$$H = C_H(\mu)H_0(q_0)\left(\frac{f}{f_0}\right)^2 \tag{5.4b}$$

$$q_0 = \frac{q}{C_q(\mu)}\left(\frac{f_0}{f}\right) \tag{5.4c}$$

Table 5.1 gives an overview of the different model variables used in the model.

Table 5.1: 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 5.3, while different constants of the model are given in Table 5.2. The values used here are based upon those given in [3]. The variables H_0, C_H and C_q are polynomial, with coefficients defined as given in Table 5.4.

Table 5.2: 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 5.3: 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 5.4: 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

5.2 Modeling and control of ESP with PINN and DEQ

As the ESP system is of a complex and nonlinear nature, obtaining a model for the system is essential for creating a control system. Due to the complexity of the ESP, and individual differences between unique pumps, it can be challenging to obtain enough training data for a deep learning model to make accurate predictions for such a system. Thus, the system might benefit from modeling using PINN. Furthermore, due to the system's complexity, using DEQ in combination with PINN might be beneficial to achieve a good and explainable model of the system. This can be achieved using the PIDEQ proposed by [34].

A goal of obtaining a good model of the ESP system is to be able to use it for control purposes. Thus, the PIDEQ should be modified so that the PINN it is based upon is usable for control purposes. This can be achieved by replacing the PINN of PIDEQ with the PINC proposed in [23]. An illustration of such a control scheme is shown in Figure 5.2.

For such a model, the DEQ needs to be adapted to take the initial condition and the control signal as inputs. A general version of the resulting model can be described by the following equation:

$$z^* = f_\theta(t, y(0), u, z^*) \tag{5.5a}$$
$$\tag{5.5b}$$

The loss function of the network can be expressed as a combination of Equation 3.9 and Equation 4.6, replacing the $\mathcal{L}_{ODE}$ and $\mathcal{L}_{PDE}$ terms with the more general $\mathcal{L}_{DE}$:

$$\mathcal{L} = \mathcal{L}_{Data} + \mathcal{L}_{DE} + \mathcal{L}_{BC} + \mathcal{L}_{IC} + \left\| \frac{df_\theta}{dz} \right\|_F \tag{5.6}$$

Some considerations have to be made when implementing such a model. The differential equation and the corresponding loss function need to be formulated. Furthermore, it is necessary to ensure that the combined system is able to predict for long time horizons.

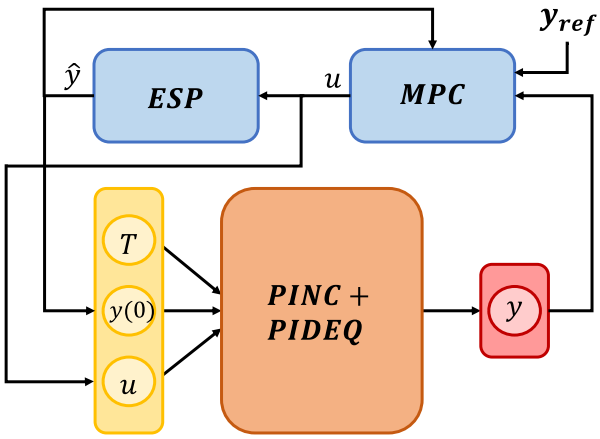

Figure 5.2: An illustration of a combined PINC and PIDEQ model in a control loop for an ESP model.

Chapter 6

Electric Submersible Pump Simulator

This chapter details the implementation and verification of an Electric Submersible Pump (ESP) simulator based upon the model described in Chapter 5.

6.1 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 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`.

6.2 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.

6.2.1 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 6.1 this is what is happening with the output from the simulator.

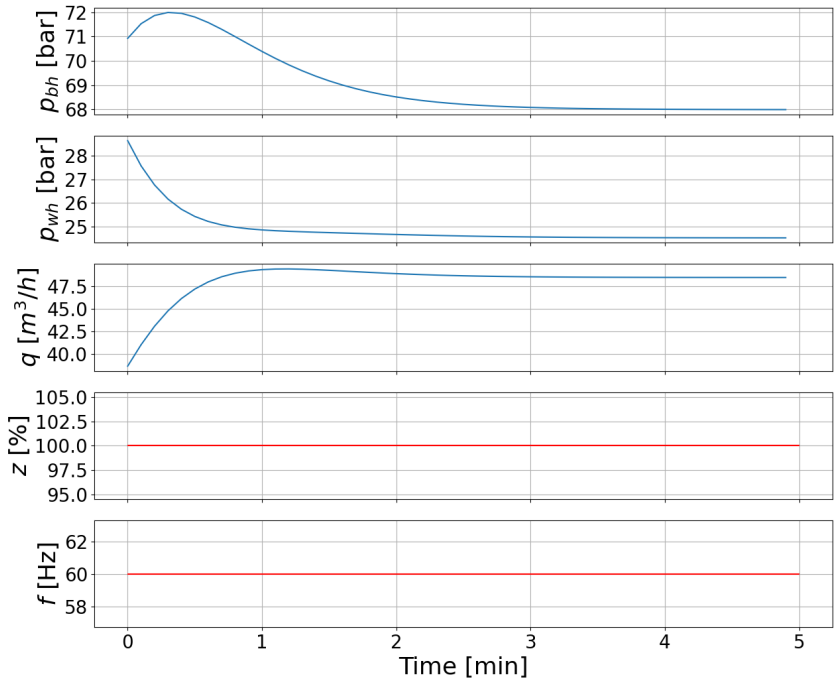

Figure 6.1: The simulation output with constant input of f = 60 Hz and z = 100%.

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

In Figure 6.2, 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 5.2b 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 5.1c it follows that as p_{wh} decreases, $\dot{q}$ and q increases. Finally, from 5.1a, as q increases, $\dot{p_{bh}}$ and p_{bh} decreases. When the valve opening is decreased again, the opposite applies.

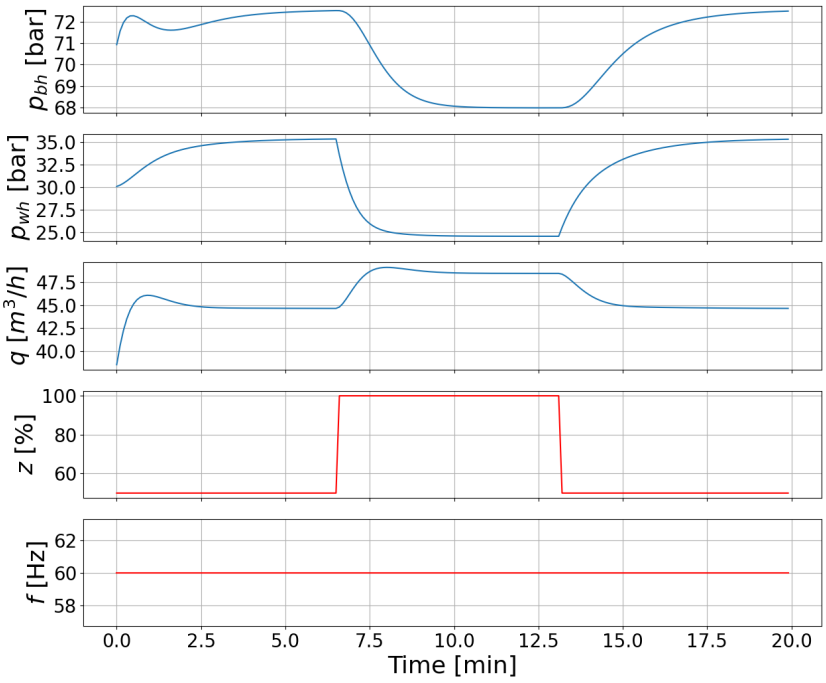

Figure 6.2: The simulation output when the valve opening is switched from 60% to 100% after one-third of the time, and back again after two-thirds.

6.2.3 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 6.3 shows the response of the system in this scenario. As can be seen from the figure, increasing 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 5.4 give a positive relationship between f and $\dot{q}$, and therefore also q. Furthermore, from equation 5.1a we see that an increase in q leads to a decrease in p_{bh}, and from equation 5.1b it leads to an increase in p_{wh}. When the frequency is lowered again, the opposite applies.

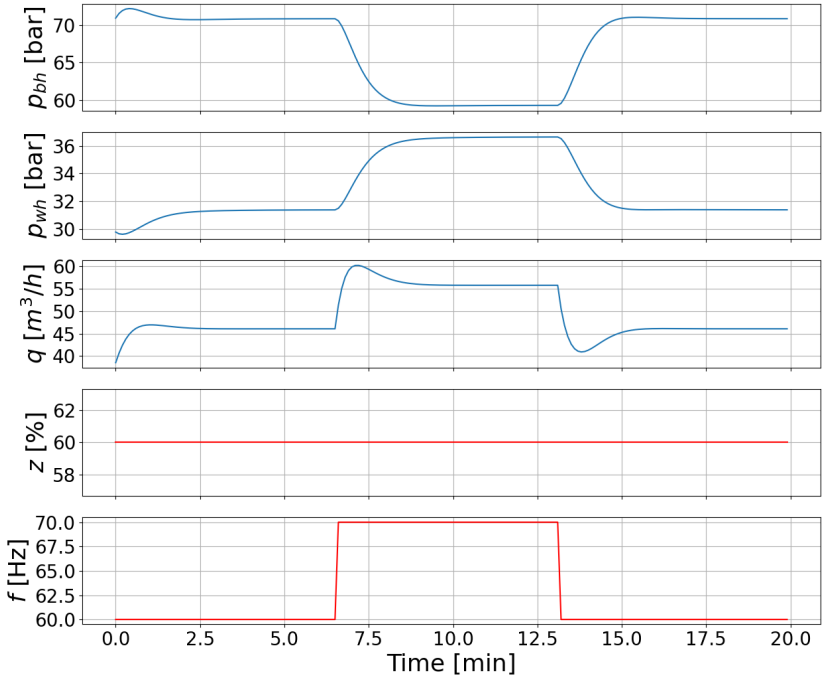

Figure 6.3: The simulation output when the frequency is switched from 60 Hz to 70 Hz after one-third of the time, and back again after two-thirds.

6.2.4 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 6.4 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 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.

6.2.5 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.

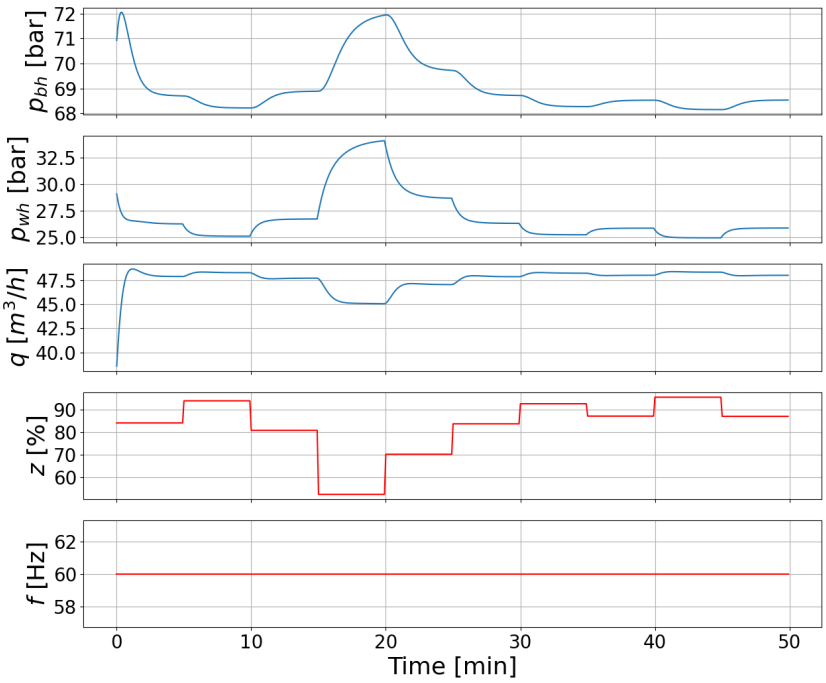

Figure 6.4: The simulation output when the valve opening is randomly changing within the limits of 60% and 100%.

Chapter 7

Discussion

This chapter will discuss the findings of the literature reviews of the deep learning frameworks Physics-Informed Neural Networks (PINNs) and Deep Equilibrium Models (DEQs). The Electric Submersible Pump model will also be discussed, focusing on the use of PINN and DEQ for the modeling and control of such a system.

7.1 Physics-Informed Neural Networks

PINN has become a popular framework since its recent proposal, much due to benefits such as its effectiveness for inverse problems and its ability to obtain good performance with smaller amounts of data. Although these networks show great promise, some issues remain, such as further development of the theoretical considerations and remediating the shortcomings of PINN variations. Yet, novel application areas of PINN keep appearing, with one recent line of research being PINN for control purposes.

Three different control schemes based on physics-informed neural networks have been examined in this work, Physics-Informed Neural Nets-based Control (PINC), Physics-Informed Deep Operator Control (PIDOC), and optimal control using physics-informed neural networks. The first two of these aim to incorporate and enforce control into a physics-informed network. The third scheme explored differs in the sense that it also is concerned with finding the optimal control for the system.

Comparing the PINC and PIDOC, a few distinctions appear. The PINC incorporates control by using the initial PINN formulation with increased network inputs, to include the control action and an initial position as well. On the other hand, PIDOC does not require any input augmentation. The design of this control scheme incorporates the desired trajectory into the loss function instead. In this case, no control action is incorporated. The nature of the PINC formulation allows it to

readily be incorporated into a control scheme. The PIDOC however, since it does not incorporate the control action as an input, can not be coupled directly with a controller.

Similarly to the PINC, the framework for optimal control using PINN also employs input augmentation. The PINC simply extends the PINN network to account for control actions. The optimal control scheme, however, is not only concerned with adjusting the network to incorporate control actions. It also aims at finding the optimal control input for the system. This results in a two-network architecture, the forward problem network similar to that found in PINC, but aslo a control input network. In many ways, due to their similar formulations, the optimal control scheme could be viewed as an extension of the PINC.

The use of PINN for control settings may provide great benefits. They allow for the identification of the underlying system based not only on data but also incorporating a priori information. Current research on such models indicates that they are more data-efficient, and they may provide faster solutions to differential equations than numerical solvers. However, there are still some aspects of these models that should be further examined and improved in order for them to be widely used. The currently developed architectures have only been tested on particular systems. Extensions of these models to different control systems, and verification of their performance are necessary. Furthermore, architectures like the PINC suffers from slow training speed, an issue that should be examined.

7.2 Deep Equilibrium Models

The deep equilibrium model offers a new architecture for deep learning models. This provides several benefits, most notably reduced memory consumption. In addition, it also has advantages such as increased explainability compared to other deep networks, and it is an intuitive way of modeling implicit relations. Some of the challenges DEQs are facing are their slowness, and lack of theoretical guarantees. An additional consideration is identifying relevant domains or contexts in which this type of model may be applicable. In this work, a literature review focusing on four areas within the field of research was conducted. These areas were recently proposed models and architectures, convergence, learning techniques, and applications.

Several attempts have been made to remediate the issue of DEQ slowness. Inspired by their success in improving other machine learning approaches, both regularization and pruning have been adapted to the deep equilibrium framework. In addition, due to the nature of the DEQ formulation, one can leverage custom solvers to achieve an improvement in speed. Regularization has shown great improvements, but as the technique is designed to reduce the complexity of the learned model, it is not able to solve the underlying problem that leads to instability. It has also been shown that pruning can reduce problems related to both training and inference, however, this technique and its possible limitations have not yet been thoroughly examined. The custom solver currently proposed led to an increase in inference speed, with minimal change in the training speed. While providing great

adaptivity, it also has the downside of having to train the solver.

Other recent advances have focused on providing theoretical guarantees for these models. The topic of convergence for DEQ has been examined for different DEQ models. While it has been shown that for linear models convergence follows, for non-linear models it currently has only been shown for over-parameterized models using a ReLU activation. This currently poses limitations on possible models to choose if guaranteed convergence is a requirement.

Recently, some attention has also been given to the application of DEQ to a wider range of problems. Most previous research has focused on problems where deep learning already achieves good results, and while DEQs provide the benefit of reduced memory consumption, their speed may prevent them from competing with established models in these cases. With a focus on a broader range of applications, such as optical flow estimation, implicit representations, and joint inference problems, DEQ could potentially offer solutions to problems where other deep learning models are unable to produce satisfactory results.

Another recent application of DEQ is in the context of physics-informed deep learning. The proposed model, the PIDEQ, has been tested for the Van der Pol oscillator problem. There, the PIDEQ did not show any advantage over a PINN model. Still, it is hypothesized that the PIDEQ can outperform PINN on more complex problems requiring deeper networks. They could also prove useful in control settings, due to the low amount of parameters necessary to implement these models.

7.3 Electric Submersible Pump and the combination of PINN and DEQ for modeling and control

A model for an Electric Submersible Pump (ESP) described by ODEs and algebraic equations was presented in this work. Based on this definition, a simulator was implemented using Python and CasADi. CasADi is a useful tool for nonlinear optimization and algebraic differentiation, yet few examples exist on how to use it to solve DAEs using Python. A link to the repository containing the simulator developed in this work has been included for such reference. This simulator exhibited behavior corresponding well to what was expected of the system and can be used to approximate the behavior of a real-world pump.

The ESP is a complex nonlinear system where control is desirable, yet it is difficult to gather enough data to obtain a reliable model. A theoretical solution to this issue is proposed in this work in the form of a control system built on the ideas of PINN and DEQ, by combining PINC and PIDEQ. Individually, the PINC and PIDEQ have shown their potential. The first has already been tested on several applications, including a gas-lifted oil well. The latter

Combining these two frameworks could potentially lead to a model capitalizing on the individual strengths of its two components, such as the ability of PINC to train on sparser datasets and the explainability and intuitive formulation of the

PIDEQ. However, neither framework is without its limitation. In the development of a combined solution, it might be necessary to make adjustments to ensure the model exhibits desired properties. If, for instance, it is found that the resulting model is too slow, it might be necessary to take advantage of other advances for improving the speed of DEQ.

Chapter 8

Conclusion and future work

In this work, literature reviews on the two recently proposed deep learning architectures PINN and DEQ have been conducted. The review of PINN was focused on the topic of PINN for control problems. The review of DEQ literature focused on recently proposed developments of the model and architecture, convergence, learning techniques, and application areas. The focus was on contributions within these areas that mitigated some of the limitations of the original DEQ formulation. In addition, recent work combining PINN and DEQ to solve the IVP of an ODE was examined.

The review of research on physics-informed neural networks and PINNs for control highlighted several newly proposed architectures for this purpose. Results indicate such network architectures may be of great benefit in control applications, but further research into their limitations and application to different systems is needed.

From the review of deep equilibrium models and their various advancements, several interesting points were discovered. Limitations of the original DEQ formulation such as the training speed have been a topic of research over the past few years, with several proposed methods for mitigating this issue. Furthermore, the application of these models to various problems such as optical flow estimation and implicit representations has shown promising results. The problem of slow training for these networks is however still an issue, and more research is needed to make the models more efficient, as well as to identify new application areas.

In addition to the reviews, a simulator of an ESP was implemented. This simulator is based upon the model given in [2] and [3] for such a system. The details of the model are given, and the resulting simulator is verified using four different scenarios. The scenarios include the operation of the pump with constant input, the response to a square pulse for the frequency and the valve opening, and random fluctuations in the valve opening. The simulator response to all these scenarios matches well with the theoretical model, thus it was concluded that the model is verified.

8.0.1 Future work

As both Deep Equilibrium Models and Physics-Informed Neural Networks are recently proposed advancements within deep learning, several topics of research remain for each. In addition to this, newly proposed applications and extensions provide even more potential focus areas for future work. Further research into the use of physics-informed neural networks in control applications is needed, extending the currently proposed methods to new applications, and comparing them to existing control methods. Also, further studies of the proposed PIDEQ model should be conducted. Specifically, the application of the model to different systems should be examined.

Finally, another open area of research is the combination of these two techniques. Section 5.2 outlines a physics-informed deep equilibrium model for the control of an electric submersible pump. A future area of research is the implementation of such a system. In order to achieve this, the implementation and verification of a PIDEQ for a system containing ODEs and DAEs are necessary. Such a PIDEQ should be tested for an ESP, and the resulting model compared with the performance of PINN. Furthermore, extending the PINC to systems described by ODEs and DAEs is necessary as well. The combination of these two techniques is done by adjusting the PIDEQ to account for control variables, in a similar manner as in the PINC formulation. To test whether the resulting architecture is suitable for control purposes, it can be tested for an oil well operated with an ESP.

Bibliography

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

[2] 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.

[3] 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.

[4] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission", in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1721–1730.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

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

[7] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu, "On the depth of deep neural networks: A theoretical view", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[8] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width", *Advances in neural information processing systems*, vol. 30, 2017.

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

[10] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey", *Heliyon*, vol. 4, no. 11, e00938, 2018.

[11] J. Gu, Z. Wang, J. Kuen, *et al.*, "Recent advances in convolutional neural networks", *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[12] S. Bai, J. Z. Kolter, and V. Koltun, *Deep equilibrium models*, Oct. 28, 2019. DOI: 10.48550/arXiv.1909.01377. arXiv: 1909.01377[cs,stat].

[13] 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.

[14] S. Bai, V. Koltun, and J. Z. Kolter, *Multiscale deep equilibrium models*, Nov. 24, 2020. DOI: 10.48550/arXiv.2006.08656. arXiv: 2006.08656[cs,stat].

[15] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal physics-informed neural network for time-dependent PDEs", *Computer Methods in Applied Mechanics and Engineering*, vol. 370, p. 113 250, Oct. 1, 2020, ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113250.

[16] T. Nguyen, M. Raghu, and S. Kornblith, "Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth", *arXiv preprint arXiv:2010.15327*, 2020.

[17] S. Bai, V. Koltun, and J. Z. Kolter, *Stabilizing equilibrium models by jacobian regularization*, Jun. 27, 2021. DOI: 10.48550/arXiv.2106.14342. arXiv: 2106.14342[cs,stat].

[18] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai, "Implicit deep learning", *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 3, pp. 930–958, Jan. 2021, ISSN: 2577-0187. DOI: 10.1137/20M1358517.

[19] S. Gurumurthy, S. Bai, Z. Manchester, and J. Z. Kolter, *Joint inference and input optimization in equilibrium networks*, Nov. 25, 2021. DOI: 10.48550/arXiv.2111.13236. arXiv: 2111.13236[cs].

[20] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning", *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[21] K. Kawaguchi, *On the theory of implicit deep learning: Global convergence with implicit layers*, Feb. 18, 2021. DOI: 10.48550/arXiv.2102.07346. arXiv: 2102.07346[cs,math,stat].

[22] X.-Y. Liu and J.-X. Wang, "Physics-informed dyna-style model-based deep reinforcement learning for dynamic control", *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 477, no. 2255, p. 20 210 618, Nov. 24, 2021, Publisher: Royal Society. DOI: 10.1098/rspa.2021.0618.

[23] "Physics-informed neural nets-based control", DeepAI. (Apr. 6, 2021).

[24] E. Winston and J. Z. Kolter, *Monotone operator equilibrium networks*, May 3, 2021. DOI: 10.48550/arXiv.2006.08591. arXiv: 2006.08591[cs,stat].

[25] S. Bai, Z. Geng, Y. Savani, and J. Z. Kolter, "Deep equilibrium optical flow estimation", presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 620–630.

[26] S. Bai, V. Koltun, and J. Z. Kolter, "Neural deep equilibrium solvers", presented at the International Conference on Learning Representations, Mar. 14, 2022.

[27] 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.

[28] G. Gokhale, B. Claessens, and C. Develder, "Physics informed neural networks for control oriented thermal modeling of buildings", *Applied Energy*, vol. 314, p. 118 852, May 15, 2022, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2022.118852.

[29] Z. Huang, S. Bai, and J. Z. Kolter, "$(\textrm{implicit})ˆ2$: Implicit layers for implicit representations", presented at the Advances in Neural Information Processing Systems, Jan. 12, 2022.

[30] 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, Jul. 2022.

[31] Z. K. Lawal, H. Yassin, D. T. C. Lai, and A. Che Idris, "Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis", *Big Data and Cognitive Computing*, vol. 6, no. 4, p. 140, Dec. 2022, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2504-2289. DOI: 10.3390/bdcc6040140.

[32] Z. Ling, X. Xie, Q. Wang, Z. Zhang, and Z. Lin, *Global convergence of over-parameterized deep equilibrium models*, May 27, 2022. DOI: 10.48550/arXiv.2205.13814. arXiv: 2205.13814[cs,stat].

[33] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed neural networks-based model predictive control for multi-link manipulators", *IFAC-PapersOnLine*, 10th Vienna International Conference on Mathematical Modelling MATHMOD 2022, vol. 55, no. 20, pp. 331–336, Jan. 1, 2022, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.09.117.

[34] B. M. Pacheco, "Physics-informed deep equilibrium models for solving odes", Bachelor's Thesis, Universidade Federal de Santa Catarina, 2022.

[35] A. Pal, A. Edelman, and C. Rackauckas, *Mixing implicit and explicit deep learning with skip DEQs and infinite time neural ODEs (continuous DEQs)*, Feb. 4, 2022. DOI: 10.48550/arXiv.2201.12240. arXiv: 2201.12240[cs,math].

[36] T.-A. Ta, T.-T. Long, H. Pham, T. Nguyen, and D. Le, "Pruning deep equilibrium models", Jul. 9, 2022.

[37] C. Wei and J. Z. Kolter, "Certified robustness for deep equilibrium models via interval bound propagation", presented at the International Conference on Learning Representations, Mar. 15, 2022.

[38] H. Zhai and T. Sands, "Controlling chaos in van der pol dynamics using signal-encoded deep learning", *Mathematics*, vol. 10, no. 3, p. 453, Jan. 2022, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2227-7390. DOI: 10.3390/math10030453.

[39] S. Zhang, L. Zhu, and Y. Gao, "An efficient deep equilibrium model for medical image segmentation", *Computers in Biology and Medicine*, vol. 148, p. 105 831, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105831.

[40] S. Mowlavi and S. Nabi, "Optimal control of pdes using physics-informed neural networks", *Journal of Computational Physics*, vol. 473, p. 111 731, 2023.