NTNU
Norwegian University of
Science and Technology

TTK4551: Specialization project (7.5 stp.)

Physics-informed Neural Networks

Sondre Bø Hernes

Submission date: December 17, 2019
Supervisor: Lars Struen Imsland

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem description

The goal of this project is to use machine learning to obtain a mathematical model for a three phase gravity separator.

The objective given by Lars Imsland is stated under:

1. Give a brief overview over neural networks and how to train them, with focus on the training as an optimization problem.
2. Perform a literature study on "physics-informed" (physics-guided, or similar) neural network approaches.
3. Implement one of these approaches in Python/Matlab, using CasADi as an optimization framework.
4. Test using data from a gravity separator case study, with various forms of noise/model errors added.

Abstract

This project aims to identify a three-phase gravity separator using data driven models. With the extensive growth of available data and computational assets, it is possible to learn the unknown dynamics or the basic relations of a dynamical system with help of artificial neural networks. Sometimes the dynamical system is very complex and the lack of available data makes it hard to use a black box approach, an alternative is then a so-called hybrid method, which tries to exploit the knowledge of both of the known physical phenomena and the data available. This paper subject is on neural and physics-guided networks and both approaches will be interpreted with focus on how to train these networks with different methods. Based on the available data from a gravity separator a black box approach using Artifial neural networks and a hybrid model are implemented and compared.

Preface

This specialization project is was performed in the Department of Engineering Cybernetics, at NTNU and took place in the spring semester 2019. The aim of this project is to give a foundation for the master thesis. I would like to thank my supervisor Lars Struen Imsland, for great guidance throughout this project and giving me the opportunity to research this topic.

Contents

List of Figures

Acronyms

ANN Artificial neural networks. 5, 7, 9–14, 17–19, 21, 28

ML Machine learning. 7, 14, 17

NLP Non-linear programming. 18, 21, 28

NN Neural Network . 12

PGNN Physics-guided Neural Network . 12–14, 29

SI System identifaction. 5

Chapter 1

Introduction

1.1 Background and Motivation

In many engineering systems, it is challenging to build mechanistic models due to e.g. lack of understanding of physical mechanisms, and the difficulty in obtaining parameters in the models from input and output data.

An important part in oil and gas production, is to separate the different elements that comes from the well stream. The multi-phase flow from the wells are considered a unknown variable, since it is difficult to measure the different elements that enters the pipe. The dynamic process that separates these elements is called a three-phase gravity separator, this separator have rich and complex dynamics, and ranging from the doctrine of hydrodynamics to thermodynamics and conservation laws[1]. Could a data drive model estimate the multi phase inflow rate to the separation tank?

Building black-box models from data require a large set of data, this might not be available in typical engineering system. A tempting alternative is to use a hybrid method, which tries to exploit knowledge both of the known physical phenomena, and in the data available.

The goal of this project is to use a data driven model to estimate the multi-phase inflow rate, based on the outflow rate of the different elements in a separation tank.

1.2 System overview

In this section a brief overview of the system will be described. The goal of this section is to give a small understanding on how the oil production works, and what happens before the elements enters the gravity separator.

The multi phase-well is installed by drilling a hole into the earth. A pipe is installed in the borehole, to provide structural strength to the bore. Different boreholes are drilled into the ground and connected to the main well. Then e.g. platform can start pumping up the different elements inside the reservoir[1][3]. In Figure 1.1 an illustration of how a offshore platform may look like.

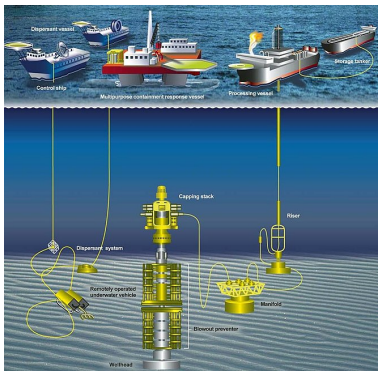

Figure 1.1: Overview of sub-sea well structure[2].

1.3 Objectives

A short summary over the objectives

- Overview of artificial neural networks, and how to train them
- Overview of physics-guided neural networks
- Estimate the inflow rate of a three-phase gravity separator using artificial neural networks
- Estimate the inflow rate of a three-phase gravity separator using a hybrid model

[1] subsurface pool

1.4 Outline

This document consist of five chapters, a short description of each chapter is described below.

Chapter 1 gives a short background and motivation for the project.

Chapter 2 provides relevant theory used.

Chapter 3 will describe the implementation for the experiment.

Chapter 4 shows the results.

Chapter 5 provides a conclusion and further work.

Chapter 2

Theory

In this chapter the relevant theory and some previously work will be presented, starting with some basic system identification, and general optimization problems. Later the framework of neural networks and physics-guided neural networks will be presented and last the three-phase gravity separator.

2.1 System identification

A cornerstone for this project is system identification. In this section will give a brief introduction to SI.

SI is a method for building mathematical model of dynamic systems. This is done by using the systems input and output relationship. The goal of SI is to understand the dynamics of the system. SI is used in fields e.q. damage detection and inspecting hidden parts of structures, and can be used to identify damage or cracks in underwater offshore parts [4]. Figure 2.1 shows the relationship between the input and output in a dynamic model.

Figure 2.1: Relation between output, input and the model

To classify dynamic systems there are three common ways, these three ways are described below:

1. **Black Box** models are the dynamics hidden, since this model exclusive relays on input and output data. This approach is used when only using ANN[5].
2. **White Box** models are based on physics equations. These models are fully comprehensible and can be modeled with adjusting the parameters[5].
3. **Grey Box** models are a combination of white and black box. Some priory knowledge is used to model the system with the black box approach. These models are often referd to as hybrid models[5].

2.2 Optimization

The goal to train a neural network is to minimize error, therefore a brief overview of optimization will be described.

Optimization is to select the best values for variables to minimize or maximize a cost function from a set of feasible values. Meaning choosing the optimal input values to get a desired cost function. Optimization is used in many fields, finans to engeneering systems[6]. The standard formulation of an optimization problem is stated below[7]:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0,\ i = 1, \ldots, m \\ & h_j(x) = 0,\ j = 1, \ldots, p \end{aligned} \tag{2.1}$$

Where f is the objective function, and g is the inequality constraints and h is the equality constraints. There are different optimization types, the most common ones are linear programming, quadratic programming and non linear programming. A small description of each are described below[7].

Convexity is an important role i optimization, if a problem is convex, means that if any local minimizer is also the global minimizer. The definition of convexity is shown in Equation (2.2) [7] [8].

$$f[\lambda y + (1 - \lambda)z] \leq \lambda f(y) + (1 - \lambda) f(z) \tag{2.2a}$$

1. **Linear programming** is the objective function and the constraints linear. This problems are convex and are usually easy to solve with e.q. simplex algorithm[7].
2. **Quadratic programming** is the objective function quadratic, but all the constraints are linear. This may or may not be easy to solve, depending if the objective is convex or not. Typically algorithm to solve quadratic programs are active set method[7].
3. **Nonlinear programming** is where either the constraints or the objective function is nonlinear. Typically algorithm to solve these problems are SQP[7].

2.3 Introduction to neural networks

Neural networks are used in many fields, e.q. image recognition, classification problems, and system identification[9]. In this project an artificial neural network is a central part for creating the data driven model. This section will give a detailed overview on how neural networks works and how to train them. In the last part some previously work will be presented.

ANN is a type of ML process that are inspired by the human brain. ANN are used when it is hard or almost impossible to express the problem in a algorithmic way. For example recognizing handwritten digits, identifying objects, self-driving cars, banking and many others. This method relies on a set of training data and a set of answers matching the training data. With this data sent through the network, the network infer rules to recognize patterns[10].

2.3.1 Structure of a neural network

A neural network is made of a set of neurons, these neurons are constructed layer by layer. The network consists of one input layer, one or more hidden layers and one output layer. Each node in the layer, are connected to every other node in the next layer. These connections are called weights and are unique. A diagram of neural network is shown under in Figure 2.2.

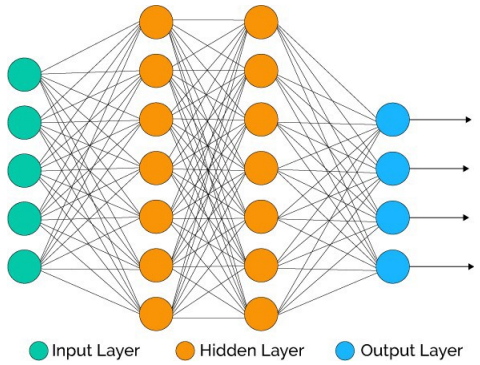

Figure 2.2: Diagram of a deep neural network[11]

The input layer in the network represent each element in the data-set. No computation is performed in these nodes. The middle layers / hidden layers consists of a set of neurons that has the input from previous layer. These nodes perform computations and address information to the next layer[10].

The last layer has the same function as the middle layers, except the output of these neurons are the final result[10].

2.3.2 Neurons

Each neuron in the network is a mathematically function. The framework of each neuron has three main parameters: input data, output data and an activation function. In the first layer the input to the neuron is the input data multiplied with its unique weight. In the next layers the input to the neuron are the output from the previous layer. The equation for each neuron is[10]:

$$\sum_{i}^{n}(X_i w_i) + b \tag{2.3}$$

Where n is the number of inputs to the neuron, X is the input value, w is each unique weight and b is the bias.

The output of each neuron goes through an activation function. A figure showing the information flow for a single neuron is shown in Figure 2.3[12].

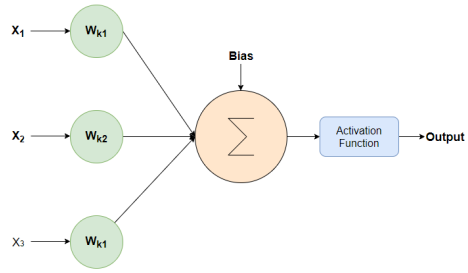

Figure 2.3: Figure showing information flow for a neuron

There are multiple reasons for why a neuron needs an activation function at its output. The activation function main purposes is:

1. **Making the output of the neuron nonlinear.** The reason for this is to make sure that the output cannot be reproduced from a linear combination of the input. If the the activation function is removed or changed with a linear activation function, it would not matter how many layers the network has, it would perform the same as a single neuron network. This is because when summing the layers, it would give another linear function[13].
2. **Enable gradient based methods.** By making the network continuous differentiable[13].
3. **Scales the output.** Making the output finite, this makes gradient based approaches tend to be more stable[13].

There are many different activation functions to use, in fig. 2.4 show the most common:

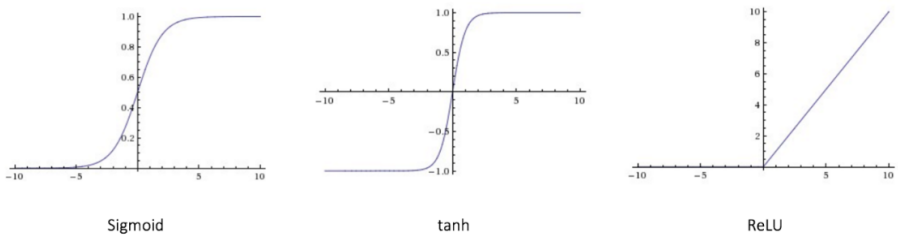

Figure 2.4: Three different activation functions (Sigmoid, tanh, ReLU)[13]

2.3.3 Feedforward

The ANN start by initializing all weights and biases as a random value. The input data is sent through the network. Each neuron in the first layer adds up input data multiplied with the weight associated with the neuron, plus the neurons bias. Then this value is sent through an activation function and fed forward to the next layer. The same approach is used in the rest of the layers [10].

2.3.4 Output and loss function

In the last layer the output from the neurons will be a prediction on what the answer should be. With this it is possible to calculate the loss in the ANN. The loss function in context of an optimization problem, is the function to evaluate a solution. In this chapter the loss function will be called the objective function. As in optimization problems the goal is to maximize or minimize the objective function. This means finding a solution that gives the highest or lowest score to the objective function[7]. In the context of neural networks, the goal is to minimize the error. There are many different ways to calculate the loss, but the core for calculating the error is the difference between the actual output and the predicted output, this is shown in the equation below:

$$\text{error} = p - \hat{p} \tag{2.4}$$

The are several difference objective functions to use, when selecting the objective function it depends on what kind of problem that needs solving. There are usually two categories for solving a problem:

Regression loss functions When using regression, the goal is to predict a real-valued quantity[14].

Binary classification loss functions Often framed to predict a value of zero or one, usually a classification problem[14].

In this paper it will be focused on regression loss functions. The most common objective function in regression problems is mean squared error[15]. The mean square error is the difference between the predicted output and the true value, squared. Then divided in the total number of data set. Shown in eq. (2.5)[15].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p - \hat{p})^2 \tag{2.5}$$

2.3.5 Minimizing the objective function

The goal for the ANN is to minimize the objective function. The optimization problem is shown **??**, when using mean square error.

This is a non-linear optimization problem. When training the neural network, the goal is to minimize the objective function. It is not possible to calculate the exact weights, it is to many variables. Typically a ANN is trained using stochastic gradient descent, which is the approach described below. ANN learn by their mistakes and uses back propagation to change the weights and biases through the network. Back propagation is the algorithmic way of computation back to input layer[10]. The method relies to see what happens to the objective function if some of the weights or biases are changed. An illustration of what happen if the weights or biases change will directly influence the objective function is shown in Figure 2.5[10].

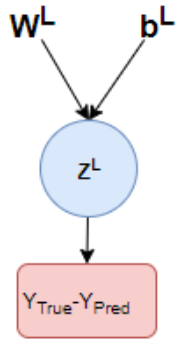

Figure 2.5: Illustration showing how the relationship between weights and biases and how they influence the objective function

To find out how much to change the weights and biases, means computing the

partial derivatives $\frac{\partial C}{\partial b}$ and $\frac{\partial C}{\partial w}$. The chain rule derivative expression, describing how sensitive the objective function is to a specific weight or bias[10].

There are three fundamental equation for calculating the change in weights and biases, this are shown in Equation (2.6)[10].

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L} \tag{2.6a}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{2.6b}$$

$$\frac{\partial C}{\partial a^{L-1}} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}} \tag{2.6c}$$

Where L indicating which layer its in, C is the objective function, z is the sum in the neuron, and a is the output of the neuron.

With Equation (2.6) the gradient can be calculated in the last layer, and use this to update the weights and biases connected to the last layer. How the weights and biases are changed depends on which optimizer the ANN use. A optimization algorithm is responsible for reducing the objective function making the ANN as accurate as possible. There are several different optimizers to use, in Table 2.1 is list on different optimization algorithms. The equation to update the weights and biases are shown below[10][16]:

$$w = w - LR * \frac{\partial C}{\partial w} \tag{2.7}$$

$$b = b - LR * \frac{\partial C}{\partial b} \tag{2.8}$$

Where LR is the learning rate.

This was the last layer, to update the weights and biases in the next layer. The same approach is used, the only difference is that the change in weight and biases are also given by the change in the previously layer. This is done until the first layer.

1	Stochastic gradient descent
2	RMSprop
3	Adagrad
4	Adam

Table 2.1: Different optimizers[17]

2.3.6 Previously work with Neural networks

There has been many cases where ANN has been used to identify dynamic systems. In this section some of the work will be presented. [18] compared different ANN architecture and used this to identify nonlinear systems. While [19] used ANN to classify nonlinear auto-regressive models. [20] used NN to identify transfer functions.

2.4 Framework of physics-guided Neural Network

In this section an overview on physics-guided neural networks will be explained. This is a subject that will be implemented in my master thesis, in the end of this section will some previously work be presented.

PGNN is a form of neural networks that is trained to solve supervised learning tasks, while respecting any law of physics. It combines prior scientific knowledge of a system, and using this to create a physic based model, then combining this with a NN to advance the results. This approach also need a physics-based loss functions, that forces the system to follow the known physics[21].

There are two steps for creating a PGNN. First step is creating a hybrid model, then the second step is to use scientific knowledge as a physic based loss function.

2.4.1 Hybrid model

A hybrid model is a combination of regular ANN and physic-based models. Both of these models can be used to predict a value based on their input. These methods require training or calibrating, ANN training procedure is mentioned in section 2.3.5. When training the physics model it comes in the form of calibrating, finding parameters that is most accurate for the model, this can be very time consuming. A hybrid model is a combination of both, neural networks and physic-based models. To overcome the faults in both approaches, the hybrid model exploit knowledge both of the known physical phenomena, and in the data available[22][23][21].

When creating the physic based model there are often two possibilities[22].

1. Using physics-based rules to model a process, or equations that describes the dynamic model.
2. Use numerical models to model behaviour over time. This can be e.q. simulations.

The framework of PGNN is shown in Figure 2.6. The input drivers are sent in to the physics model, creating an output Y_{phy}. The same input drivers are sent to the ANN along with the output of the physics model Y_{phy} creating a hybrid model.

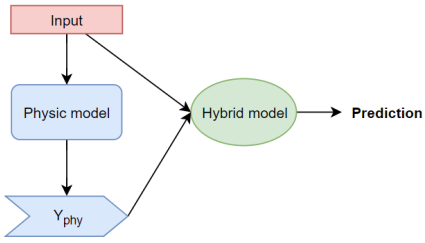

Figure 2.6: Information flow of a hybrid model[24]

2.4.2 Physic-based loss function

If the system does not include a physic based loss function, the predictions may violate the physical relationships and other variables. If the predictions of the physic model is almost identical to the labeled data, the ANN will try to match the predictions of the physics model. If opposite the physics model will systematically make some mistakes, then the predictions will learn the bias using the input drivers[22].

The framework for using a physic-based loss function is shown in Equation (2.9)[22][21].

$$\min \quad \underbrace{\text{training loss}(Y_{true}, Y_{pred})}_{\text{Empirical Error}} + \underbrace{\lambda R(W)}_{\text{Structual Error}} + \underbrace{\lambda_{PHY}\text{Loss.PHY}(\hat{Y})}_{\text{Physical Inconsistency}} \tag{2.9}$$

Simply by minimising the training loss and the model complexity, which is the traditional way for training a machine learning model. The model also need to include some sort physic based loss that is what ever physics you want to corporate in the ML model[22][21].

There are two big advantages for using this physic based loss function. The first advantage is that the output becomes more consistent. The second advantage is how the model can learn for samples that has no observations. Since the physic-based loss does not need the labeled data. The model can check if the output consistent physically, and can still improve the ML model, just by checking if the its physically consistent[21].

Figure 2.7 showing the PGNN in context of black box ANN and physic model. Where the x axis show the use of data and the y axis shows the use of scientific knowledge[21].

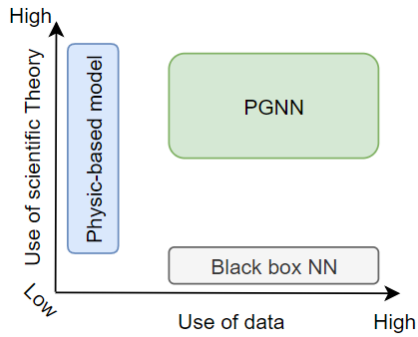

Figure 2.7: Figure showing the use of data in context of PGNN

2.4.3 Previously work

In [21] uses a PGNN to estimate different different temperatures at different depths. They use a very similar approach as this project does. Using a lake model [25] as a physics-based model, and the relationship between density and temperature to develop a physic-based loss function.

2.5 Tank Model

In this section, the goal is to give an overview over what physical process the project is around. This section will describe a three-phase separation tank.

The separation of oil, gas and water can be done in a three-phase separation tank. This is usually the first step to roughly separate the elements. It works on the principle that different elements have different densities. A separation tank is shown in Figure 2.8 [26].

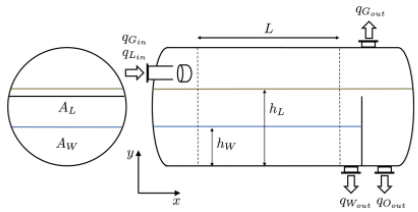

Figure 2.8: Simplified figure of a three-phase gravity separator[27]

As seen in the figure above, the tank has one inlet, and the elements flowing in is split into three phases. One water phase, one oil phase, and one gas phase. Because of the different densities of the elements, the three substances will stack on top of each other. Water at the bottom of the tank, oil over water and gas on top. In the tank there is a barrier that separates oil from water[26].

Chapter 3

Implementation

In this chapter the implementation will be explained. The first two sections will describe how the ANN was made. Then the physics model will be introduced, and last the hybrid model.

3.1 Keras and Tensorflow, neural networks

To build and train the ANN for the gradient descent approach, Python version 3.7 was used. The libraries Keras with Tensorflow as back-end was used to make the ANN. **Keras** is a open source ANN library used in Python [28]. **Tensorflow** is a open source platform for ML [29]. The structure of the ANN is shown below in table 3.2 and in figure 3.1. With the use of the function sequential from Keras, makes it possible to construct the ANN in a user friendly way.

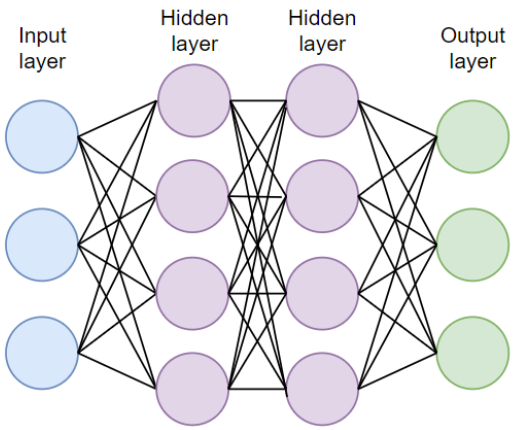

Figure 3.1: Structure of Neural network

Epochs	50
Optimizer	Adam
Activation	Sigmoid
Loss	MSE

Figure 3.2: Training parameters

3.2 Using Casadi - solve NLP

In the second approach the goal was to minimize the objective function using a NLP solver. The same size of the ANN as in figure 3.1 was used, only now everything was designed from scratch in Python. To minimize the objective function Casadi was implemented in Python. **Casadi** is an open source library that can be used in Python as a tool for nonlinear optimization [30]. The parameters that was used with was Interior Point OPTimizer (IPOPT) that is a software library for nonlinear optimization. Here the loss was calculated for every set of data points, then divided by the number of training data. Then all the weights and biases were changed to the optimal solution for minimizing the objective function.

3.3 Physics based model

For the physic-model, a model of the three-phase gravity separator made by Backi[27] was used. This model was created in Matlab and Simulink, in this implantation a few modifications were made. First modification was changing the input to a random input instead of a step response, setting the simulation time long enough to get a big enough data-set, and inserting noise sources to the measurements.

3.4 The hybrid model

The creation of the hybrid model is shown in Figure 3.3. Here the tank model made by Backi acts as a physic model. Using the output of the different elements as an input to the model and estimating the inflow rate of oil, gas and water. The same signal is fed to the ANN along with the output from the physics-model. The model then estimates the inflow rate of water, gas and oil. For training and composition of the ANN Keras and tensorflow was used, with MSE loss function, sigmoid as a activation function and 50 epochs.

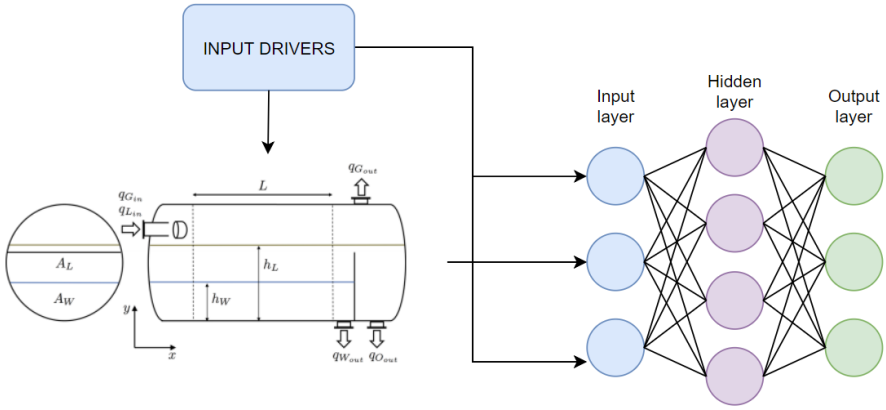

Figure 3.3: Schematic for the hybrid model

3.5 Data-set

All the labeled data used to train the ANN is collected from the tank model made by Backi. Two data-set was collected, one to train the network and one to validate the network afterwords. The data set contained the outflow of each valve, with their respective inflow rate. The training data set consists of 30001 data samples, and the test data consists of 3201 test data samples. This data sets are saved in mat files and open in Python using the library Scipy.

Chapter 4

Results

In this chapter the result will be presented. The first experiment will be using Casadi as an optimization framework for training the network. The next results shows when using Keras and Tensorflow and a gradient descent method. After this two black box approaches the result of the hybrid model will be presented. All these plots are from the test data-set of 3201 data points.

4.1 Using Casadi as an optimazation framework

Results using NLP to train the ANN are shown in Figure 4.1, Figure 4.2 and Figure 4.3 below.

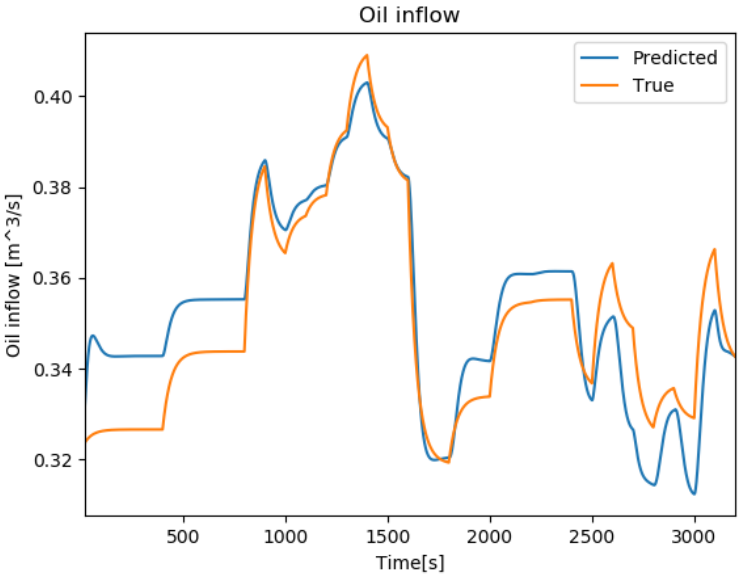

Figure 4.1: Predicted vs true values for oil inflow using Casadi

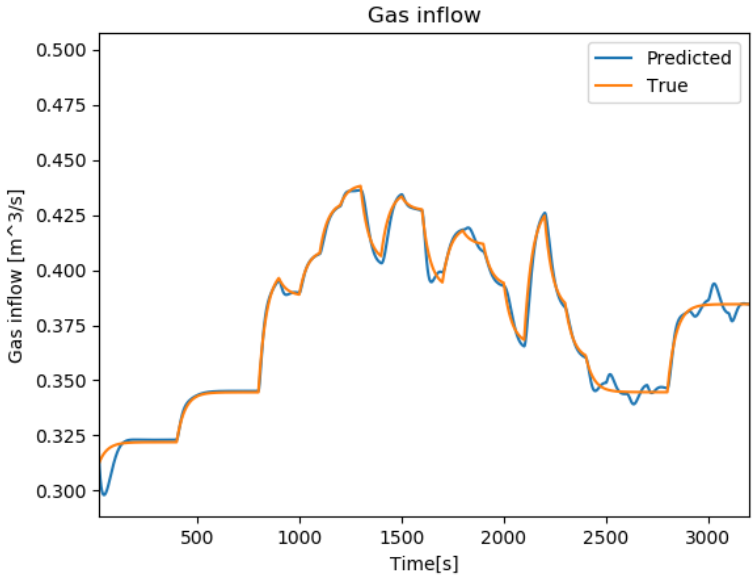

Figure 4.2: Predicted vs true values for gas inflow using Casadi

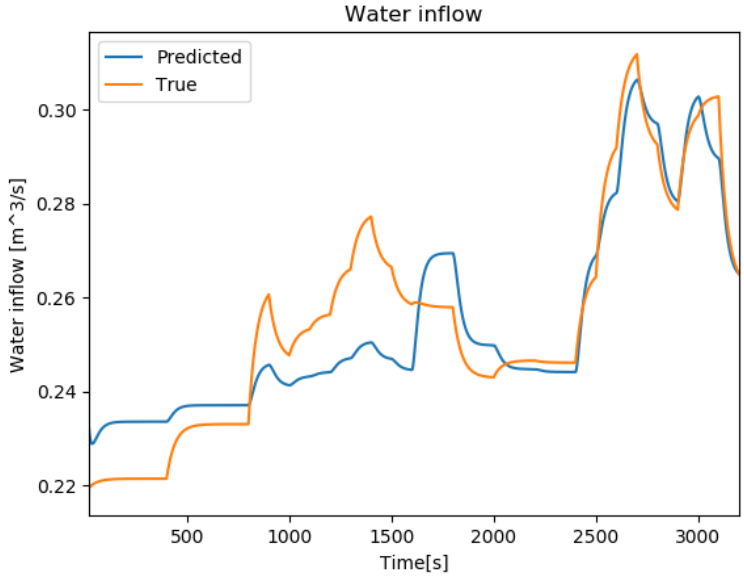

Figure 4.3: Predicted vs true values for water inflow using Casadi

As seen in the plots above the prediction follows the real value quite good for oil and gas, it keeps the same form as the real value, but misses in some places. For the prediction for water, the result were more poorly.

4.2 Results with Keras - Tensorflow

Figure 4.4, Figure 4.5 and Figure 4.6 shows the result of using gradient descent method.

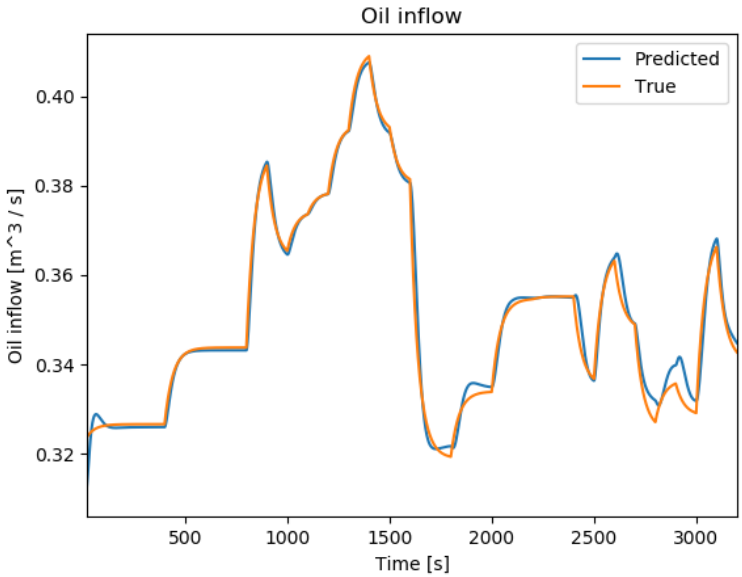

Figure 4.4: Predicted vs true values for oil inflow using Casadi

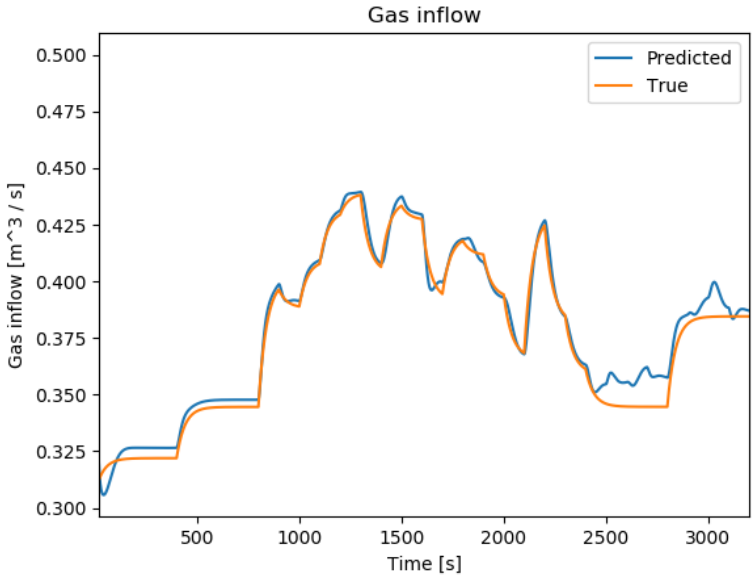

Figure 4.5: Predicted vs true values for gas inflow using Casadi

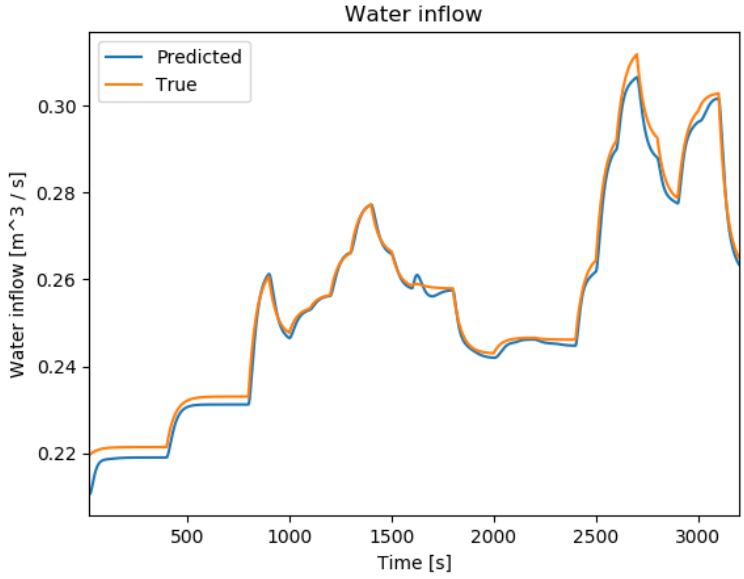

Figure 4.6: Predicted vs true values for water inflow using Casadi

The results with using gradient descent shows that the predictions follows the same path as the true value. It misses some places, but overall a quite good result.

4.3 Noise added

Figure 4.7 shows when noise is added to the output measurement of the tank. Only one plot i shown, since the result was the same using both procedures with noise added.

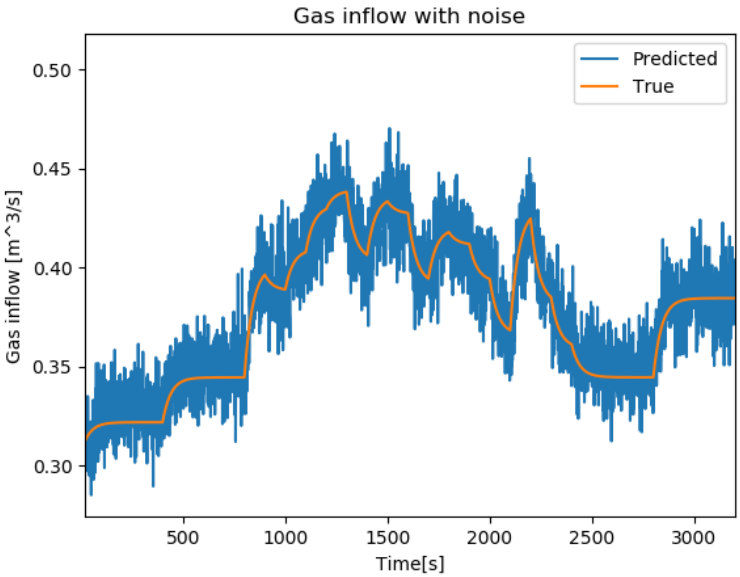

Figure 4.7: Predicted vs true values for gas inflow with white noise on the output measurement

4.4 Hybrid model

The folowing fig. 4.8, Figure 4.9 and Figure 4.10 show when using the physics-guided model.

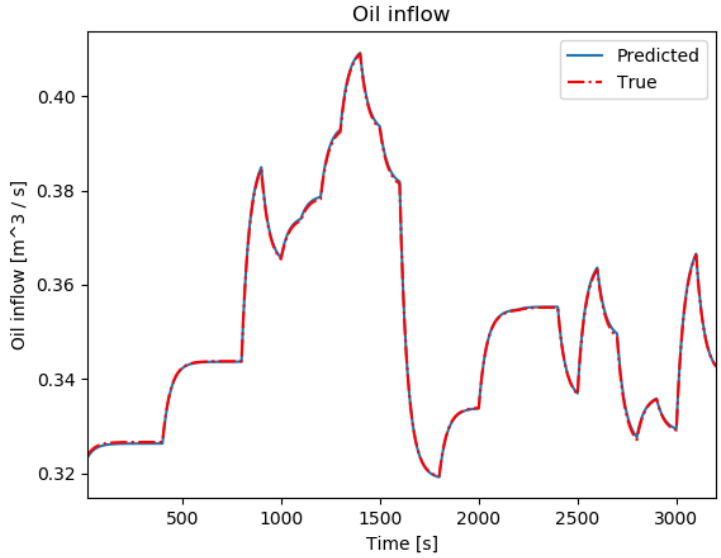

Figure 4.8: Predicted vs true values for oil inflow using a hybrid model

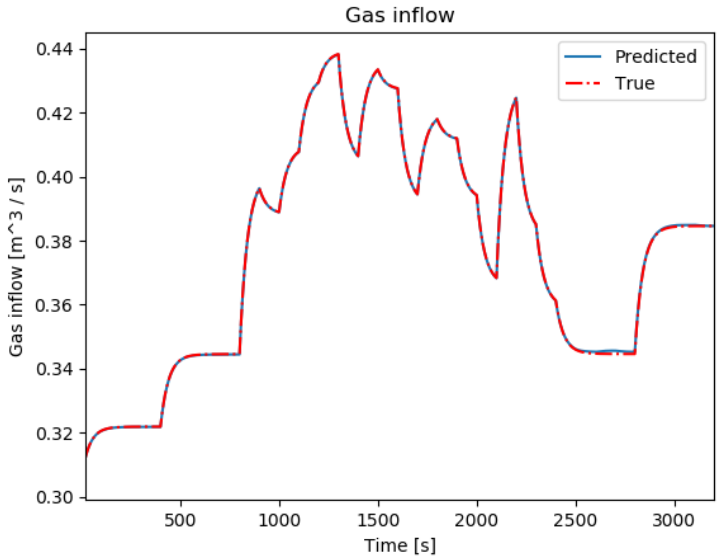

Figure 4.9: Predicted vs true values for gas inflow using a hybrid model

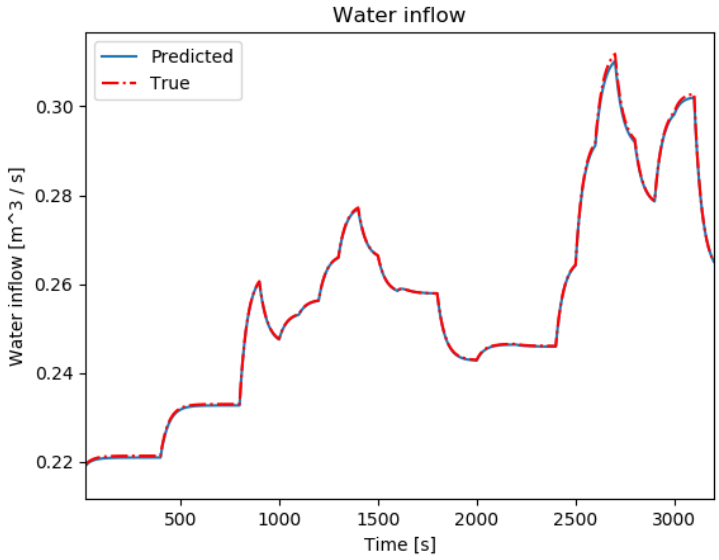

Figure 4.10: Predicted vs true values for water inflow using a hybrid model

The result of using a hybrid model, was very accurate. The prediction follows the true value without deviation.

4.5 Discussion

In this section the result from above will be discussed.

Casadi gave the worst result. When trying to improve the performance of Casadi by building deeper ANN, Casadi had an attendance to crash, when to many variable were implemented. Therefore there were some limitations when trying to solve the NLP. Also the run-time for minimizing the objective function was significant longer than the other approach. The reason for using Casadi for training the network in addition to the gradient descent method, is because in future work a MPC might be used, and Casadi is often used when implementing an MPC, meaning it is an advantage to have everything in the same framework. Also when using Casadi, you have more freedom when creating the network, and this might come as an advantage when implementing the physic loss Equation (2.9).

Keras and Tensorflow approach gave some good results. Also with using these library's it became quite easy to change and test different sizes of the ANN structure, also changing the training parameters. One downside using these libraries was that it made it hard to change something that was not included in the library. E.q. the loss function, to a physics based loss function.

Hybrid approach gave the best results, but might not be surprising when the data for training the network was created in the physics model used. In a real problem this will not be the case, the data will be noisy and measurements might not be available. Since the predictions of the physic model is identical to the labeled data, will the ANN then try to match the predictions of the physic model, this is one reason why the result became so accurate.

The **data-set** used to train the network is made from perfect measurements simulated in Matlab. Also with the Matlab model, it is possible to collect data from anywhere in the process, this might not be possible in a real life. Often the biggest challenges using ANN is to find a good enough data set, to train the network. In this project a simulated model made perfect data sets, and large enough to get accurate results. This might not be the case, especially for a three-phase gravity separator, since it is hard to measure. Without this the black box approaches becomes useless since it requires a good data-set to train on.

The **objectives** for this project was to create a data driven model of a gravity separator. This was implemented, with a black box approach and with a hybrid solution. As expected the result for the hybrid model gave significant better results. In further work a physic based loss function should be implemented.

Chapter 5

Future work and Conclusion

5.1 Further work

Further work in this project will be implementing the physic-based loss function to the hybrid model making it a PGNN. The first physic equation that should be tested is the mass balance. The mass balance equations is an application of conservation of mass, and this can be used to analyse physical systems. The principle of mass balance is that enters the tank must either leave the system or accumulate within the system[31]. Mathematically, mass balance for a system with no chemical reaction is stated below in eq. (5.1).

$$\text{Input} = \text{Output} + \text{Accumulation} \tag{5.1}$$

The second equation to look into is Bernoulli's principle. The principle states that an increase in speed of a fluid will happen simultaneously with a decrease in static pressure. Bernoulli's equation is given in Equation (5.2).

$$\text{p} + \frac{1}{2}\rho V^2 + \rho g h = constant \tag{5.2}$$

Where p is the pressure, ρ is the density, V is the velocity, h is the elevation and g is the gravity acceleration[32].

5.2 Conclusion

Using only black-box data driven models, require often a huge data set. Also when using a black-box model for supervised learning problem, the result can only be as good as the representative data-set. If the size of the data-set available is limited, the model often learn a relationship between the input data and the labeled data, that can perform surprisingly well, but when testing it outside available labeled data the result shows otherwise. A second matter is that black box models, is the absence of known laws of physics. Meaning that the predictions made from the black box network can break the laws of physics[22]. This matter can be improved by using a physic guided neural network, but this approach also demands some insight on the physics in the dynamic system. As shown above a hybrid model gave the best results, but since the data used was created in the physic model, means that the model is perfectly tuned, and the neural network will learn to match predictions done by the physic model.

References

[1] Atalla F. Sayda and James H. Taylor. "Modeling and Control of Three-Phase Gravity Separators in OilProduction Facilities." In: ().

[2] URL: https://commons.wikimedia.org/wiki/File:Figure_2-_Subsea_Well_Containment_Response_Equipment_(7045774079).jpg (visited on 12/12/2019).

[3] *Petroleum production*. URL: https://www.britannica.com/technology/petroleum-production/Recovery-of-oil-and-gas (visited on 12/12/2019).

[4] Hossein Kiamehr and Amin Ghafouripour. "AN OVERVIEW OF SYSTEM IDENTIFICATION METHODS AND APPLICATIONS." In: (2000).

[5] Lev V. Kalmykov1and Vyacheslav L. Kalmykov. "A white-box model of S-shaped anddouble S-shaped single-speciespopulation growth." In: (2015).

[6] Ding-Zhu DuPanos M. PardalosWeili Wu. "History of Optimization." In: (2008).

[7] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Science and Business Media, 2006.

[8] Stephen J. Wright. *Optimization*. URL: https://www.britannica.com/science/optimization (visited on 12/12/2019).

[9] H. J. Sjoberg and L. Ljung Hjalmarsson. "Neural Networks in System Identication." In: ().

[10] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[11] URL: https://medium.com/udacity-pytorch-challengers/hyperparameters-for-neural-networks-c50ab565ee3d (visited on 12/12/2019).

[12] URL: https://www.altoros.com/blog/introduction-to-neural-networks-and-metaframeworks-with-tensorflow/ (visited on 12/12/2019).

[13] Ujjwalkarn. 2016. URL: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/ (visited on 12/12/2019).

[14] URL: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/ (visited on 12/12/2019).

[15] URL: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0 (visited on 12/12/2019).

[16] .

[17] URL: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f (visited on 12/12/2019).

[18] M. O. Efe and K. Kaynak. "P Comparative Study of Neural Network Structures in Identification of Nonlinear Systems." In: (1999).

[19] P.K. Dash H.K. Sahoo and N.P. Rath. "NARX model based nonlinear dynamic system identification using low complexity neural networks." In: ().

[20] T. A. Tutunji. "Approximating Transfer Functions using Neural Network Weights." In: (2009).

[21] Jordan Read Anuj Karpatne William Watkins and Vipin Kumar. "Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling." In: (2018).

[22] Anuj Karpatne. "How can physics inform deep learning methods in scientific problems." In: 2018.

[23] P. Perdikaris M. Raissi and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." In: (2018).

[24] URL: https://towardsdatascience.com/physics-guided-neural-networks-pgnns-8fe9dbad9414 (visited on 12/12/2019).

[25] L. Bruce M. Hipsey and D. Hamilton. "General Lake Model." In: (2014).

[26] N. Makwashi T. Ahmed1 and M. Hameed. "A Review of Gravity Three - Phase Separators." In: (2017).

[27] Brian A. Grimes Christoph J. Backi and Sigurd Skogestad. "A Control- and Estimation-Oriented Gravity Separator Model for Oil and Gas Applications Based upon First-Principles." In: (2018).

[28] URL: https://keras.io (visited on 12/12/2019).

[29] URL: https://www.tensorflow.org (visited on 12/12/2019).

[30] URL: https://web.casadi.org (visited on 12/12/2019).

[31] Jayaprakash.J and Hari Kumar M.E. “State Variable analysis of four tank system.” In: (2014).

[32] URL: http://hyperphysics.phy-astr.gsu.edu/hbase/pber.html (visited on 12/12/2019).