Hybrid Modeling of a Production Wellbore

Morten Fredriksen

Supervisor: Professor Lars Imsland
Co-supervisor: Mathilde Hotvedt

NTNU
Norwegian University of
Science and Technology

Department of Engineering Cybernetics
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology
January 2021

Contents

Chapter 1

Introduction

In petroleum production systems, understanding the mass flow characteristics is of vital economic importance. By adjusting production choke openings and pressure regulators, petroleum engineers can regulate the petroleum output to meet demand. However, in order to efficiently optimize the system like this, accurate knowledge of the mass flow characteristics is crucial. In this sense, the potential mass flow governs the optimization potential of the production asset, as well as feasibility of future developments. In order to understand the mass flow behaviour of the production system as a whole, it may be more efficient to consider it as an assembly of smaller sub-systems, each with their own flow characteristic (Jansen, 2015). There are several established methods for determining the multi-phase flow through the various sub-assemblies. These include first-principle (mechanistic) models, data-driven models, and naturally, physical flow meters. Non-physical flow meters, that is data-driven and mechanistic models, are often referred to as virtual flow meters (VFMs). Each of these methods have their own strengths and weaknesses. Mechanistic models can be made quite accurate, but increase in complexity with more stringent accuracy requirements. This can can result in the models being unsuitable for real-time applications. Data-driven models have the potential to be a lot more computationally efficient than purely mechanistic models, but require a vast amount of data which may be either impossible or not economically viable to procure. Both of these techniques have been developed due to the limitations of physical multi-flow meters, which are often expensive to operate and re-tune, and are prone to erosion and damage. A full overview and comparison of virtual flow meters can be found in (Bikmukhametov and Jäschke, 2020).

Hybrid modeling is a modeling approach aimed at combining the strengths of mechanistic- and data-driven models. Although rare, some of the first instances of this type of modeling can be found in (Psichogios and Ungar, 1992), and have later been tried out in various industry processes, such as biochemical plants in (Solle et al., 2017) and petroleum production systems (Kanin et al., 2019). Recently, (Hotvedt et al., 2020a) used hybrid modeling to construct a virtual flow meter for a petroleum production choke with promising results. There are several advantages to using hybrid models, some of which are summarized in (Solle et al., 2017):

- For some systems, the underlying mechanistic relations for certain phenomena may not be known, or too computationally expensive to include in the mechanistic model. In such cases, this part of the model can be approximated with a data-driven model instead.
- In many processes, high-quality may not be available, and the performance of data-driven models are therefore limited. Incorporating already established relations through the use of mechanistic models can mitigate the effect of low signal-to-noise ratio training data.
- In addition to being prone to noisy input, data-driven models also need a sufficient amount of said data in order to generalize well. In many processes, data is either to scarce or too expensive to generate, thus making full-fledged data-driven models infeasible. In such cases, creating smaller, less data-hungry data-driven models and supplementing the gap with mechanistic models may be a suitable compromise.
- Purely data-driven models such as neural networks use internal parameters not available to us. Furthermore, NN-parameters have no physical meaning, and thus interpreting the methodology of the network is often not possible. Learnable parameters in hybrid models have the advantage of being based in real-world physical properties. This enables us to examine how well the inner-workings of the model correlates the values of the learnable parameters to the empirical values of the physical properties they represent.

  In this project, we will try to leverage these advantages to create an accurate, yet adaptable hybrid model of a production wellbore.

1.1 Scope of Thesis

In this paper a hybrid model for the mass flow through a wellbore will be developed. The hybrid model will based on a simplified mechanistic model of a wellbore, but where the friction factor is described with a neural network. Implementation of this model will be discussed, with regards to optimization, regularization and choice of hyperparameters. The model will then be trained on a dataset generated from the same simplified mechanistic model, but with a calculated friction factor as apposed to an estimated one. After training, the model performance will be evaluated with regards to accuracy, trainability and interpretability.

1.2 Method

As mentioned earlier, in order to model the multi-phase mass flow through a wellbore, we will develop a hybrid model. The mechanistic part of the model will be based on a simplified mechanistic first-principle model. This makes parameter estimation of the physical parameters easier, and the expectation is that this estimation can be made with a reasonable set of data points. Furthermore, the data-driven part of the model will be comprised of a neural network, and will estimate the friction coefficient used in the mechanistic model part. By estimating the friction factor in this manner, the model will be made explicit, making its calculation simpler. Both the parameter estimation for the mechanistic model and the data driven model will be conducted using an iterative, stochastic optimization algorithm. This way, the model should scale well even on large amounts of data.

1.3 Thesis Structure

The structure of this thesis will now be outlined. Chapter 1 introduced the project and motivated the use of a hybrid modeling technique. Chapter 2 will give an introduction to the relevant theory needed to understand the model and its implementation, starting with a brief introduction to relevant aspect of flow dynamics in Section 2.1. Following will be an overview of the first-principle wellbore model in Section 2.2 and an introduction to the data-driven model aspects in Section 2.3. Moving on to Chapter 3 the hybrid model specification will be constructed and its implementation discussed. Section 3.1 will describe the dataset used for training and how this was generated. Section 3.2 and 3.3 will briefly describe the mechanistic- and data-driven model components, respectively, before they are combined to form the hybrid model in Section 3.4. The optimization problem for this model will be calculated in Section 3.5 before the implementation details will be discussed in Section 3.6. The results of the model after training will be presented in Chapter 4. These results will be discussed in Chapter 5 with future works suggested in Section 5.4. Finally, a conclusion will be stated in Chapter 6.

Chapter 2

Theory

In this section, relevant theoretical aspects needed for the modeling will be briefly explained. First, an introduction to flow dynamics in Section 2.1 and its application to mechanistic modeling of wellbores in Section 2.2 will be presented. This can be considered the basis for the mechanistic model that will be presented in Section 3.2. Next, the basis for the data-driven models will be introduced in Section 2.3.

2.1 Flow Dynamics

The oil production rate of a petroleum asset is largely dependent on the potential mass flow through the production string. The properties of this mass flow are governed by fluid dynamics. In this section, the relevant concepts of fluid dynamics necessary to construct the first-principle model in Section 2.2 will be briefly introduced. This is done in order to provide an understanding of the assumptions that will later be made in the modeling process.

Compressible and Incompressible Fluids

When talking about a compressible or incompressible fluid, it concerns the properties of the density ρ of the fluid in question. An incompressible fluid is considered to have a constant density over varying conditions such as pressure changes. Although all fluids are compressible to some degree, assuming incompressibility greatly simplifies the mathematical modeling, and is a valid assumptions when dealing with real liquids such as water and oil where the density is almost constant (von Mises and Friedrichs, 1971). Compressible fluids exhibit density variations based on pressure- and temperature fluctuations, and thus the density $\rho = \rho(p)$ can no longer be considered constant. Gases are usually considered as compressible fluids.

Steady-State Homogeneous Flow

A flow is said to be steady-state if there are no change to the flow regime during the time interval considered. In practice, this means that the mathematical model of the flow is time-independent. Prerequisite assumptions necessary for steady-state flow are uniform thickness, incompressible fluid and flow across a constant circumference (Chaudhry, 2004). The mass flow from a production well is multi-phased. A multi-phased flow consists of a mixture of several fluids, in this case primarily water, oil and gas. In order for the mixture to meet the requirement of uniform thickness, it has to be thoroughly mixed and with homogeneous flow. Homogeneous flow here refers to the negligible relative motion between different phases of the fluid. Thus, the mass and moment conservation equations for single-phased fluids may be utilized for the fluid mixture as a whole (Brennen, 2005).

Temperature and Frozen Flow

Temperature plays an integral role in fluid dynamics. The state of matter and density of the fluid, as well as the pressure in the wellbore are all effected by temperature. However, in this model we will assume a properly insulated wellbore such that no thermal energy will be transferred to the surroundings. Then the only introduction of thermal energy to the system will be through the effects of friction from the pipe wall, however this amount is negligible. Thus, we can assume frozen flow, where the fluid mass fractions ϵ remain constant from bottom hole to wellhead. This makes the following mathematical modeling considerably simpler, but it should be noted that assuming a constant temperature is somewhat of a harsh assumption for real-world systems.

2.2 First-principle Modeling of a Wellbore

In this section we will develop the mathematical model for the mass flow through a production wellbore, as depicted in Figure 2.1. First, a few assumptions regarding the fluid flow will be made. We will consider an incompressible liquid in a steady-state homogeneous flow through a streamline. The streamline will consist of insulated tubing with constant cross-sectional area A. Furthermore, there will be negligible temperature difference along the area of consideration. These concepts were discussed in Section 2.1.

All these assumptions considered, the steady-state Bernoulli's energy equation for flow along a streamline can be utilized (Cengel and Cimbala, 2014):

$$\frac{p_1}{g} + \frac{v_1^2 \rho_1}{2g} + z_1 \rho_1 = \frac{p_2}{g} + \frac{v_2^2 \rho_2}{2g} + z_2 \rho_2 + \rho_2 h_f \tag{2.1}$$

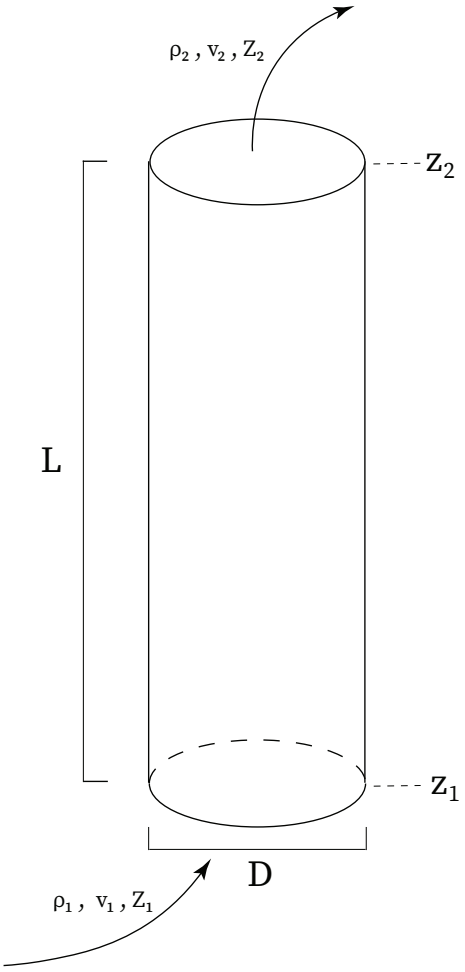


Figure 2.1: Illustration of the wellbore. It can be seen as a vertical pipe with length L and diameter D.

where,

i : index 1 for bottomhole and index 2 for wellhead
p : pressure$[Pa]$
ρ : fluid density $\left[\frac{kg}{m^3}\right]$
z : height$[m]$
v : fluid velocity $\left[\frac{m}{s}\right]$
g : gravitational constant $\left[\frac{m}{s^2}\right]$
h_f : head loss due to pipe friction$[m]$

The head loss h_f can be expressed as follows [S.L. and C.A. (2010)]:

$$h_f = f\frac{Lv^2}{2Dg} \tag{2.2}$$

where,

f : friction factor$[-]$
L : total length of wellbore$[m]$
D : hole diameter$[m]$

Rearranging the Bernoulli equation (2.1) and adding the head loss expression (2.2) yields:

$$p_1 + \frac{v_1^2\rho_1}{2} + z_1\rho_1 g = p_2 + \frac{v_2^2\rho_2}{2} + z_2\rho_2 g + f\frac{L\rho_2 v^2}{2D} \tag{2.3}$$

Since the ultimate goal is to model the mass flow through the wellbore, the conservation of mass from bottom hole to wellhead may be utilized. Assuming the cross-sectional area of the pipe to be constant yields:

$$\dot{m} = \rho_1 v_1 A = \rho_2 v_2 A \tag{2.4}$$

where $\dot{m}$ is the rate of change in mass through the pipe and A is the cross-sectional area of the pipe. Adding the mass balance (2.4) to the Bernoulli expression (2.3) and solving for the mass flow rate $\dot{m}$ yields:

$$p_1 + \frac{\dot{m}^2}{2\rho_1 A^2} + z_1\rho_1 g = p_2 + \frac{\dot{m}^2}{2\rho_2 A^2} + z_2\rho_2 g + f\frac{L}{D}\frac{\dot{m}^2}{2\rho_2 A^2}$$

$$\dot{m} = \sqrt{\frac{2\rho_1\rho_2 D(p_1 - p_2 + z_1\rho_1 g - z_2\rho_2 g)}{D\rho_1 + fL\rho_1 - D\rho_2}} \tag{2.5}$$

The friction factor f may be determined in multiple ways (Guo et al., 2007). In this model we will utilize the method outlined in Awad and Muzychka (2008) as it is suited for homogeneous flow:

$$f = \begin{cases} \frac{16}{Re} & Re < 2300 \\ \frac{0.079}{Re^{0.25}} & Re \geq 4000 \end{cases} \tag{2.6}$$

where Re is the Reynolds number expressed as

$$Re = \frac{\dot{m}D}{A\mu} \tag{2.7}$$

with μ denoting mixture viscosity.

Next, we will find an expression for the fluid mixture density ρ and viscosity μ. However, some preliminary relations must first be established. Starting with the assumption of frozen flow and a fluid without contamination, the mass fractions ϵ of gas(G), oil(O) and water(W) constitutes the entire fluid mass, that is:

$$\epsilon_O + \epsilon_W + \epsilon_G = 1 \tag{2.8}$$

Using the water-to-oil ratio, or water cut(wc), expressions for the liquid(L) density and viscosity are found:

$$\begin{aligned} wc &= \frac{\epsilon_W}{\epsilon_W + \epsilon_O} \\ \mu_L &= wc * \mu_W + (1 - wc)\mu_O \\ \rho_L &= wc * \rho_W + (1 - wc)\rho_O \end{aligned} \tag{2.9}$$

The gaseous fluid density is found using the real gas law:

$$\rho_G = \frac{pM_G}{ZRT} \tag{2.10}$$

where,

$$\begin{aligned} M_G &: \text{molar mass of the gas } \left[\frac{g}{kmol}\right] \\ R &: \text{ideal gas constant } \left[\frac{J}{kmol}\right] \\ T &: \text{temperature}[K] \\ Z &: \text{compressibility factor}[-] \end{aligned}$$

The compressibility factor $Z = Z(p, T)$ may be calculated using the correlation in Sutton (1985). Considering homogeneous fluid mixture, the density ρ and mixture viscosity μ, may be calculated using Awad and Muzychka (2008):

$$\frac{1}{\rho} = \frac{x_G}{\rho_G} + \frac{x_L}{\rho_L} \tag{2.11}$$

$$\frac{1}{\mu} = \frac{x_G}{\mu_G} + \frac{x_L}{\mu_L} \tag{2.12}$$

2.3 Machine Learning and Neural Networks

The second component necessary to build a hybrid model is the data-driven model. In this section, a short summary of relevant concepts in machine learning and neural networks necessary to build the data-driven part of the hybrid model in Section 3.3 will be presented. The following content is primarily sourced from Goodfellow et al. (2016).

2.3.1 Definition of Machine Learning

A machine learning algorithm can be defined as being able to learn a task from experience, at a certain performance level. The defining characteristic of machine learning is that the performance level increases as more experience is fed to the algorithm. The terms "task", "experience" and "performance" in this definition are quite vague, allowing machine learning algorithms to be applicable to many applications. In this project machine learning will be used for regression, that is, for finding an output to an unknown function based an input of features $\mathbf{x}$. An 'unknown function' here refers to a function $f(\mathbf{x})$ in which its mathematical structure is not completely known, or assumed not to be. However, using machine learning algorithms a function $g(\mathbf{x})$ can be constructed that has the same input-output characteristics as $f(\mathbf{x})$, such that $g(\mathbf{x}) \approx f(\mathbf{x})$. It should be noted that the mathematical definition of function $g(\mathbf{x})$ may not be available to us, especially when it is constructed using a neural network. Implementation of a regression algorithm for the friction factor f from equation (2.6) will be discussed in Section 3.3.

2.3.2 Components of a Machine Learning Algorithm

The key components necessary to transfer the experience into a usable output of a machine learning algorithm can be summarized as follows:

- **Dataset:** The dataset is a set of features $\mathbf{x}$, and sometimes outputs $\mathbf{y}$, that constitutes the experience a machine learning algorithm processes in order to improve its performance. The dataset available to us in this project includes both features and outputs, and may thus be used in so-called *supervised* learning algorithms, such as regression. A more detailed overview of the features and generation of this dataset is available in Section 3.1. The kind of data stored in the dataset has a large influence on how the machine learning algorithm should be designed. For instance, a dataset consisting of pictures with tags specifying the motif of the picture (such as MNIST) will require a completely different type of algorithm and output than a dataset containing data points to fit to a curve.
- **Model:** A model specification of the system. Choice of model greatly impacts the effectiveness of the optimization algorithm. More effective optimization is often achieved using linear models, such as linear regression $\hat{y} = \mathbf{w}^T\mathbf{x} + b$, but these usually have limited capacity, meaning the model is limited in the amount of functions it can approximate. Thus, choosing a model is often a balancing act of finding a model with enough capacity, while still being able to be optimized at a reasonable computational cost.
- **Cost Function:** The cost function represents the value of the model we want to minimize. For instance, when the objective of the model is to fit a function to data (regression), an appropriate cost function might be to calculate the mean squared error between the model output $\hat{y}$ and the output from the training data y. Another choice could be mean absolute error for this type of task. Different models and objectives require different cost functions. Another important aspect of cost functions is that they can be modified in order to *regularize* the model. More on this in Section 2.3.3.
- **Optimization Algorithm:** The role of the optimization algorithm is to minimize the cost function. Nearly all deep learning algorithms use some variant of stochastic gradient

descent (SGD). This optimization algorithm is a variation of gradient descent, a well known optimization technique. Gradient descent becomes stochastic when only parts of the data is used to calculate the gradient in each iteration. SGD uses only one data sample at the time whilst fixed-size mini-batch SGD uses a randomly drawn sub-set of samples to calculate the gradient. This reduces the computational complexity of the optimizer at every iteration, making it possible to find low values of the cost function in reasonable time. Note however, that gradient descent does not guarantee this value to be a global minimum. Alternatives to stochastic gradient descent include RMSProp and Adam. However, all gradient-based optimization algorithms need the gradients of the objective function for every training cycle. These gradients are calculated using back-propagation, which is already implemented in most machine learning libraries.

Mixing and matching these four components, a large set of machine learning algorithms are obtainable for a multitude of applications.

2.3.3 Generalization and Regularization

The ultimate goal of any machine learning algorithm is to *generalize* well. Generalization here refers to the algorithms ability to predict a correct output given a set of inputs not included in the training data. This performance measure governs the practical usability of the algorithm. If the algorithm correctly predicts the output values in the training set, but fails to predict any new or slightly anomalous inputs, the practical application of the algorithm on new real-world data is limited. This is often referred to as overfitting. Critically, the objective of minimizing the error on training data is not directly positively correlated to the minimization of the generalization error. In other words, any increase in training error does not necessarily cause an increase in generalization error. This is where *regularization* becomes relevant. Regularization is any change done to the machine learning algorithm that improves the generalization error, but not necessarily the training error. As mentioned earlier, adjusting the cost function is a common way of achieving regularization, usually with parameter norm penalties. A popular such method is L^2 parameter regularization. Other popular regularization techniques include early-stopping and data augmentation.

2.3.4 Network Architecture

Neural Networks are a type of machine learning algorithm developed to solve more abstract generalization tasks. Instead of using a single model, the model is divided among many 'neurons' in a network, connected by activation functions. The most common type of network is a so-called feedforward neural network. It is called feedforward due to the unidirectional flow of data from input to output of the network. An example of an activation function might be the rectified linear unit (ReLU). This function is an almost completely linear function, making it easy to optimize over. A more in-depth description of activation functions can be found in most literature on deep learning, such as Goodfellow et al. (2016). Every node in the network is a seperate, smaller model. By using a simple linear function in every node, we have created a *linear* feedforward network, which in practice is a set of piecewise linear equations.

An important design choice for neural networks is the network architecture. The architecture defines how many layers the network will consist of (depth) and how many nodes every layer

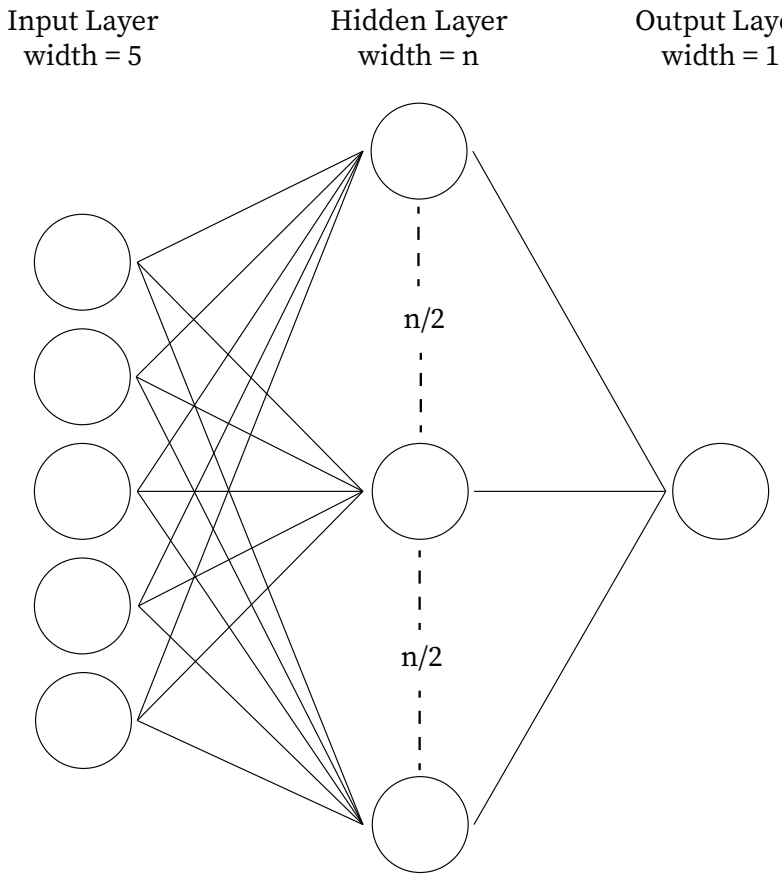


Figure 2.2: Fully connected feed forward neural network. Every line is an instance of activation function.

will consist of (width). The first layer is commonly referred to as the input layer, and should have width corresponding to the number of system inputs. The last layer is often referred to as the output layer, and should have as many nodes as desired outputs. The layers in between are referred to as hidden layers, seeing as the training data does not contain a desired output for the nodes in these layers. An illustration of the type of model that will be used in this project can be seen in Figure 2.2. This is a fully connected neural network, meaning that all nodes from one layer is connected to all nodes of the next.

Chapter 3

Method

This section will describe the implementation and simulation of a hybrid model for mass flow estimation in a wellbore, based on the principles and theory outlined in Section 2. First, a description of the dataset used in training the model will be provided in Section 3.1. Next, Sections 3.2 and 3.3 will present the mechanistic- and data-driven model components, respectively. These will be combined to form the hybrid model in Section 3.4. The optimization problem for this hybrid model will then be calculated in Section 3.5 before the implementation details are discussed in Section 3.6.

3.1 Dataset

Before handling the model specification, a brief look at the available data is prudent. The data is organised as a set of features $\mathbf{x}$ and an output y corresponding to a steady state of the wellbore system. The features consists of bottomhole pressure p_1, wellhead pressure p_2, temperature T, bottomhole compressiblity factor Z_1 and wellhead compressiblity factor Z_2. The output for every datapoint is either the calculated mass flow $\dot{m}$ or the calculated mass flow with added noise $\dot{m}_{noise}$. Both are included in the dataset such that the robustness of the hybrid model can be more thoroughly tested. The dataset can be expressed as:

$$\begin{aligned} \mathcal{D} &= \{\boldsymbol{x}_i, y_i\}_{i=1}^{n} \\ \mathbf{x}_i &= [p_{1,i}, p_{2,i}, T_i, Z_{1,i}, Z_{2,i}] \\ y_i &= [\dot{m}_i] \vee [\dot{m}_{noise,i}] \end{aligned} \tag{3.1}$$

The dataset was generated by implementing the first-principle model outlined in Section 2.2. Upon closer inspection, it is clear that this first-principle model is implicit, seeing as the friction factor f necessary to compute the mass flow $\dot{m}$ is also dependent on the same mass flow. The model was therefore implemented in a Python program using the symbolic mathematics library "Sympy", capable of solving implicit systems. In order to generate the data, the inputs p_1, p_2 and T were drawn from normal distributions, with means and variances

based on empirical evidence. The compressibility factors Z_1 and Z_2 were calculated from correlations as outlined in Sutton (1985). Well specific parameters were assumed constant with values specified in Table 4.2. Turbulent flow was assumed, and this assumption was then checked for correctness by calculating the Reynolds number based on the produced mass flow $\dot{m}$. Finally, an additional output was calculated by adding noise to $\dot{m}$: $\epsilon \sim N(0, 1)$.

The dataset $\mathcal{D}$ will be split in 3 parts; the training set $\mathcal{D}_{train}$, the validation set $\mathcal{D}_{val}$ and the test set $\mathcal{D}_{test}$. Dataset $\mathcal{D}$ will be randomly split 80/20 between $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$. The training set $\mathcal{D}_{train}$ will then subsequently be split again at a 80/20 ratio with the validation set $\mathcal{D}_{val}$ such that the number of training samples in each set is:

$$\begin{aligned} \text{len}(\mathcal{D}) &= 5000 \\ \text{len}(\mathcal{D}_{train}) &= 3200 \\ \text{len}(\mathcal{D}_{val}) &= 800 \\ \text{len}(\mathcal{D}_{test}) &= 1000 \end{aligned}$$

3.2 Mechanistic Model Component

The mechanistic part of the model will be based on the first-principle wellbore model introduced in Section 2.2, with a few modifications. Recall that two variables were yet to be defined in this model; the compressibility factor Z and the mixture viscosity μ. As mentioned previously, the compressibility factor Z is already a part of the available dataset, so in the mechanistic model implementation this variable will be considered a system input. This simplifies the model and decreases the computational load. The only parameter related to the viscosity μ in the mass flow system (2.5) is the friction factor f, as defined in equation (2.6). By defining the variable f as an output of a data-driven model, the system dependency on the viscosity μ will be represented in the data-driven model rather than in the mechanistic one. Thus, the final model will be independent on μ. We are left with the following mechanistic model description, referred to as g_{MM}:

$$\begin{aligned} \dot{m} &= g_{MM}(\mathbf{x}, \mathbf{\Phi}_{MM}) \\ &= \sqrt{\frac{2\rho_1\rho_2 D(p_1 - p_2 + z_1\rho_1 g - z_2\rho_2 g)}{D\rho_1 + fL\rho_1 - D\rho_2}} \end{aligned} \tag{3.2}$$

where the mechanistic model parameters $\mathbf{\Phi}_{MM}$ are:

$$\mathbf{\Phi}_{MM} = [x_G, x_O, \rho_O, \rho_W, M_G, L, D] \tag{3.3}$$

and the system inputs $\mathbf{x}$ are:

$$\mathbf{x} = [p_1, p_2, T, Z_1, Z_2] \tag{3.4}$$

In this model implementation, it will be useful to define another set of parameters as "learnable" parameters $\mathbf{\Phi}_{LP}$. This will be a subset of the mechanistic parameters $\mathbf{\Phi}_{MM}$ such that $\mathbf{\Phi}_{LP} \subseteq \mathbf{\Phi}_{MM}$. The learnable parameters will be estimated using the dataset, while the remaining mechanistic parameters will be presumed constant. More on initialization of these parameters follows in Section 3.6.1.

3.3 Data-Driven Model Component

The data-driven part of the hybrid model consists of a fully connected feed-forward neural network, used to estimate the friction coefficient f from equation (2.6). The chosen network is a 3-depth linear feed forward neural network with 5 input nodes in the input layer (corresponding to the 5 system inputs), 100 nodes in the hidden layer and a single output node, as seen in Figure [2.2]. We use the rectified linear unit (ReLU) as activation function on each layer except the output layer. A summary of these machine learning concepts were provided in Section 2.3. Mathematically, we can express the neural network g_{NN} as:

$$f = g_{NN}(\mathbf{x}', \mathbf{\Phi}_{NN}) \tag{3.5}$$

where the neural network parameters $\mathbf{\Phi}_{NN}$ are comprised of the node weights ($\mathbf{W}$) and biases ($\mathbf{b}$) such that:

$$\mathbf{\Phi}_{NN} = [\mathbf{W}, \mathbf{b}] \tag{3.6}$$

and the inputs $\mathbf{x}'$ are:

$$\mathbf{x}' = [p_{1,s}, p_{2,s}, T_s, Z_1, Z_2] \tag{3.7}$$

The subscript s here denotes that the inputs have been scaled. This is done using a min-max scaler function in Sklearn with the purpose of normalizing the individual input variations' impact on the network performance. The scaler has been initialized on the training samples, and the same distribution has been used on the validation- and test set. The compressibility factors Z_i do not need to be scaled, as they are already in range $(0, 1)$.

3.4 Complete Hybrid Model Description

There are several ways to hybridize the first-principle wellbore model in Section 3.2. In this project we will create a serial hybrid model using the mechanistic model as a baseline, and then introduce the data-driven model in Section 3.3 in order to estimate the friction factor f. This model architecture is illustrated in Figure 3.1, where the friction factor f is first estimated using the neural network, before the estimation $\hat{f}$ is fed to the mechanistic model. Modeling the wellbore this way lets us train a data-driven model without the use of the actual friction coefficient f, which may be hard to accurately calculate using first-principles. Instead, the system can be trained on $\dot{m}$, which is more readily available. By combining the two mathematical model expressions (3.2) and (3.5) we obtain the final mathematical expression for the hybrid model:

$$\begin{aligned} \hat{\dot{m}} &= g_{HM}(\mathbf{X}, z, \mathbf{\Phi}_{MM}) \\ &= g_{HM}(\mathbf{X}, \mathbf{\Phi}) \\ z = f &= g_{NN}(\mathbf{X}', \mathbf{\Phi}_{NN}) \end{aligned} \tag{3.8}$$

where the model parameters is the set of both the mechanistic and neural network parameters:

$$\mathbf{\Phi} = [\mathbf{\Phi}_{MM}, \mathbf{\Phi}_{NN}] \tag{3.9}$$

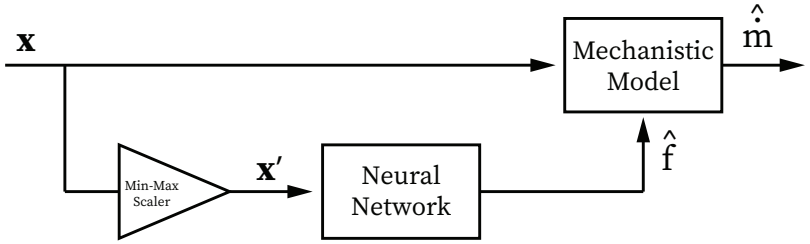


Figure 3.1: Block diagram depicting the hybrid model architecture and data flow.

3.5 Defining the Optimization Problem

With the complete hybrid model defined, we can also put together the optimization problem for the machine learning algorithm to solve. As mentioned in Section 3.3, the model has a set of learnable parameters $\mathbf{\Phi}_{LP}$ as well as a set of neural network parameters $\mathbf{\Phi}_{NN}$. The task of the machine learning algorithm is to optimize over this set of parameters $\mathbf{\Phi}^* = [\mathbf{\Phi}_{LP}, \mathbf{\Phi}_{NN}]$. Optimizing with regards to the mean squared error and adding separate L^2 parameter regularization for $\mathbf{\Phi}_{LP}$ and $\mathbf{\Phi}_{NN}$ yields the following optimization problem:

$$\begin{aligned}\hat{\mathbf{\Phi}}^* &= \arg\min_{\mathbf{\Phi}^*} J\left(\mathbf{\Phi}^*, \boldsymbol{\lambda}^*\right) \\ &= \arg\min_{\mathbf{\Phi}^*} \left(\frac{1}{n} \sum_{i=1}^{n} \left(\dot{m}_i - g_{HM}(\mathbf{X}_i, \mathbf{\Phi})\right)^2 + \frac{1}{n} \sum_{j=1}^{m_{NN}} \lambda_{NN} \left(\Phi_{NN,j} - \bar{\Phi}_{NN,j}\right)^2 \right. \\ &\qquad \left. + \frac{1}{n} \sum_{k=1}^{m_{LP}} \lambda_{LP,k} \left(\Phi_{LP,k} - \bar{\Phi}_{LP,k}\right)^2 \right)\end{aligned} \tag{3.10}$$

where

n : the number of training samples in the batch
m_{NN} : the number of neural network parameters
m_{LP} : the number of learnable parameters
λ_{NN} : the L^2 regularization parameter for the neural network parameters
λ_{LP} : the L^2 regularization parameters for the neural learnable parameters
$\bar{\Phi}$: the mean value of the parameter Φ

Furthermore,

$$\boldsymbol{\lambda}^* = [\boldsymbol{\lambda}_{LP}, \lambda_{NN}]$$

The computation of these regularization parameters will follow in Section 3.6.3.

3.6 Implementation Details

Now that the hybrid model has been properly defined in Section 3.4 and the corresponding optimization problem has been derived in Section 3.5, only a few implementation details

remain to be discussed. These details will be discussed here before the result of the model training is presented in Section 4.

3.6.1 Parameter Initialization

There are in total three different types of parameters in the hybrid model; the mechanistic parameters $\mathbf{\Phi}_{MM}$, the learnable parameters $\mathbf{\Phi}_{LP}$ and the neural network parameters $\mathbf{\Phi}_{NN}$. Before training on the model starts, these parameters have to be initialized. In real- world applications, all parameters used have some degree of uncertainty associated with them, and should thus be estimated. Usually this is done using gradient descent. However, in order to simplify the model some of the well known wellbore-specific parameters, such as the diameter D and height L of the wellbore, will initialized at their known value. Other mechanistic parameters may have a higher degree of uncertainty associated with them, such as the oil density ρ_O or the mass fractions ϵ_O, ϵ_W and ϵ_G. These can either be initialized at some presumed value, or be considered learnable parameters. The learnable parameters will be initialized by drawing an initial value from a normal distribution about the parameters' empirical mean value and standard deviation. Finally, as common practice suggests, the neural network parameters will be initialized using Kaiming initialization, as outlined in He et al. (2015). A positive bias will be used, seeing as the mass flow is biased in positive flow direction.

3.6.2 Training

Training the model (3.8) refers to solving the optimization problem (3.10) by use of the dataset $\mathcal{D}_{train}$ defined in Section(3.1). The optimization algorithm to be used in this implementation is the adaptive moments optimization scheme (Adam), mentioned in Section 2.3.2. When training the model, the hyperparameters can be tuned in order to achieve better model performance. However, in order to prevent poor generalization performance, all hyperparameter-tuning will have to be conducted using the output of the validation set $\mathcal{D}_{val}$. The test set $\mathcal{D}_{test}$ is reserved for analysing the model performance once tuning is completed.

3.6.3 Hyperparameters

In this section we will establish an overview of the hyperparameters in the implementation of the hybrid model. First, the L^2 regularization scheme added to the cost function in (3.10) also adds m_{LP} hyperparameters in the form of the regularization parameters λ_{LP}, in addition to one parameter for the neural network λ_{NN}. These are defined according to standard as follows:

$$\begin{aligned} \lambda_{NN} &= 0.001 \\ \lambda_{LP,i} &= \frac{\sigma_e^2}{\sigma_i^2} \end{aligned} \tag{3.11}$$

where σ_e^2 is the standard deviation of the measurement noise and σ_i^2 is the standard deviation of the learnable parameter. If the measurement noise is not available to us, this may also be treated as a hyperparameter. Other hyperparameters include the training batch size B,

the number of training epochs E, the learning rate l_{rate} and the neural network architecture specifications such as depth and width. For hybrid models, it can be argued for that the the choice of learnable parameters is also a hyperparameter in of itself. In this implementation, the learnable parameters will be ρ_O and ρ_W such that:

$$\begin{aligned} \mathbf{\Phi}_{LP} &= [\rho_O, \rho_W] \\ m_{LP} &= 2 \end{aligned}$$

Chapter 4

Results

In this section, the training results and the final model performance will be presented. The model has been evaluated in 4 configurations. These configurations originate from two variable configuration settings: whether the output y of the dataset $\mathcal{D}$ is $\dot{m}$ or $\dot{m}_{noise}$, and whether L^2 parameter regularization has been implemented on the learnable parameters $\mathbf{\Phi}_{LP}$ or not. The configuration definitions are given in Table 4.1.

Model Configuration	L^2 regularization	Output y
$c_{-,-}$	no	$\dot{m}$
$c_{-,noise}$	no	$\dot{m}_{noise}$
$c_{reg,-}$	yes	$\dot{m}$
$c_{reg,noise}$	yes	$\dot{m}_{noise}$

Table 4.1: Configuration specification for the 4 possible model implementation configurations.

All the parameter values used in the simulation of this model are summarized in Table 4.2. A description of these parameters where given in Section 2.2.

Parameter	Value
g	9.81
R	8.314
M_G	0.0269
L	2000
D	0.178
ϵ_O	0.9
ϵ_W	0.0
ϵ_G	0.1
z_1	0
z_2	2000
$\bar{\rho}_O$	830
$\bar{\rho}_W$	1020

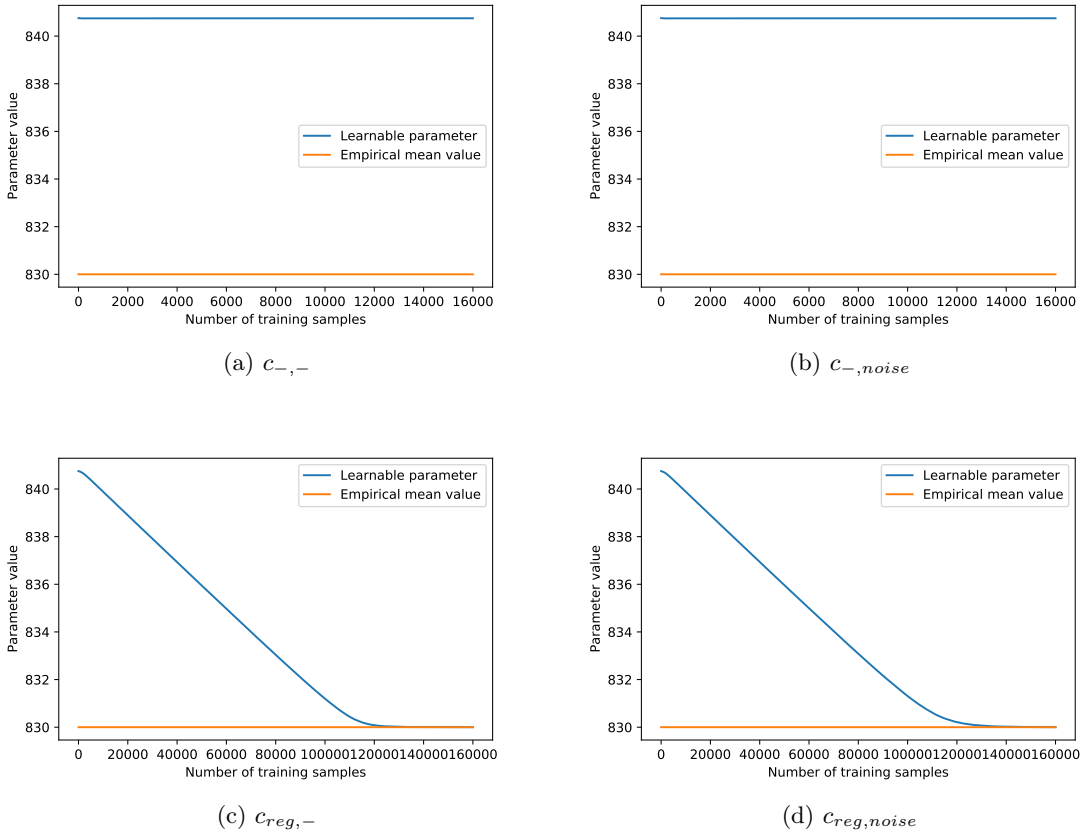
Table 4.2: Parameter values used in this implementation.

The training results have been evaluated in two ways. First, since the dataset $\mathcal{D}$ is generated using the empirical mean values for the learnable parameters, this mean value has been plotted against the learnable parameter value over the training cycle. This is shown in Figure 4.1 and Figure 4.2. Second, the total loss function value, the mean squared error (MSE) loss, the learnable parameter loss and the neural network(NN) loss have been plotted over the course of the training cycle. This result can be found in Figure 4.3.

The performance of the model is measured in three ways. First, a cumulative performance plot is given for each of the configurations, as seen in Figure 4.4. This plot shows what percentage of test points fall within the deviation stated on the x-axis. The closer the graph is to a step function, the better. More on cumulative performance plots can be found in (Corneliussen et al., 2005). Second, a histogram of the friction coefficient f, both estimated by the neural network and the friction model (2.6) - (2.7), is given for each configuration in Figure 4.5. Finally, a table summarizing the mean absolute error (MAE) over f and $\dot{m}$ in the test set $\mathcal{D}_{test}$, as well as the final absolute errors (AE) of the learnable parameters ρ_O and ρ_W after training can be seen in Table 4.3.

Model Configuration	MAE(f)	MAE($\dot{m}$)	AE(ρ_O)	AE(ρ_W)
$c_{-,-}$	4.803e-6	0.235	10.75	6.023
$c_{-,noise}$	3.941e-6	1.055	10.74	6.023
$c_{reg,-}$	3.609e-6	0.406	0.011	0.0
$c_{reg,noise}$	3.550e-6	0.832	0.002	0.0

Table 4.3: Performance metrics for 4 different model configurations: with/without added noise to $\dot{m}$ and with/without L^2 regularization on the learnable parameters $\mathbf{\Phi}_{LP}$. The performance metrics are: Mean absolute error (MAE) of the friction factor f and model output $\dot{m}$ over all the samples in the test set, as well as the absolute error (AE) of the densities ρ_O and ρ_W compared to their mean values.



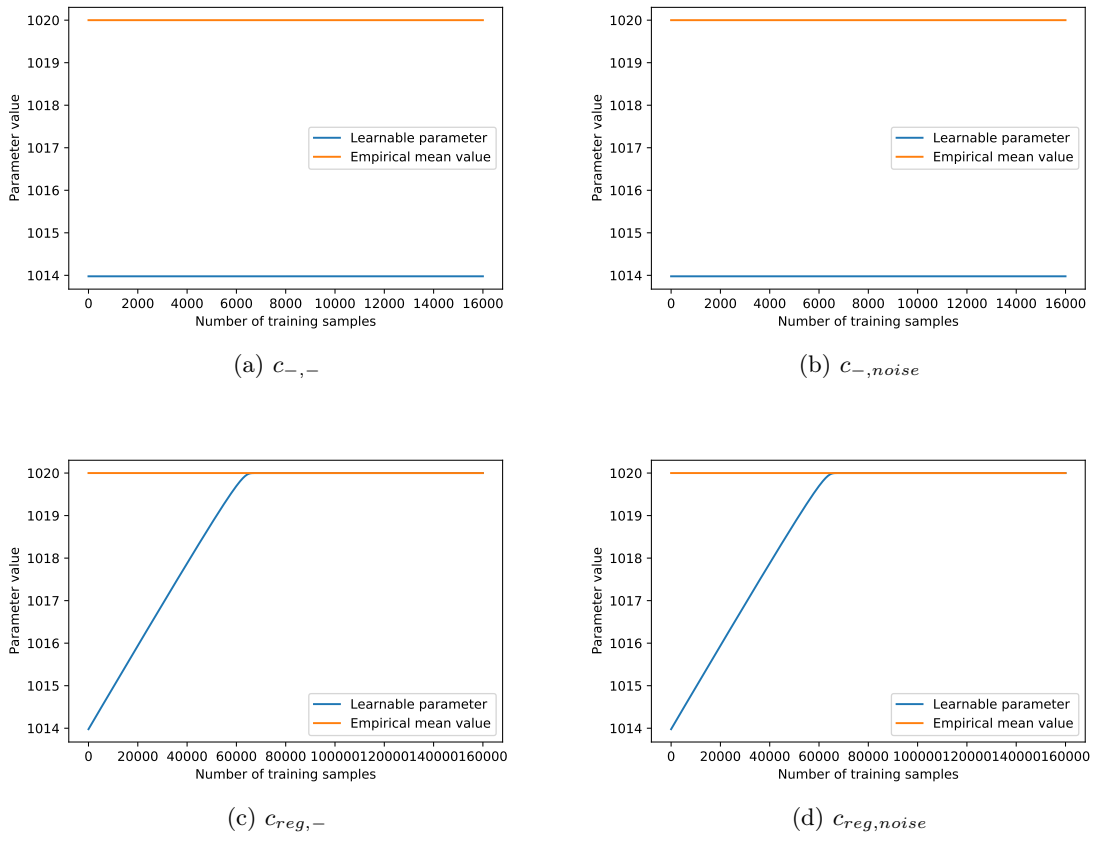
(a) $c_{-,-}$

(b) $c_{-,noise}$

(c) $c_{reg,-}$

(d) $c_{reg,noise}$

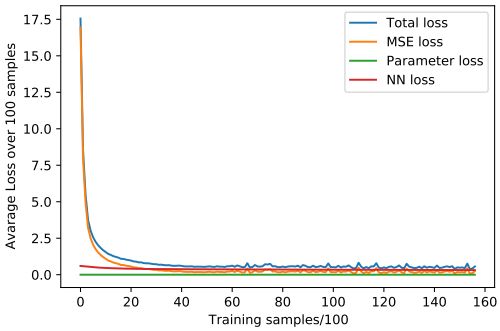
Figure 4.1: The learnable parameter value of ρ_O over the number of training samples considered. The empirical mean value of ρ_O is plotted for reference.



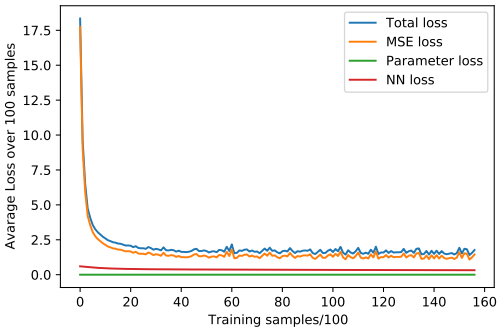
(a) $c_{-,-}$ (b) $c_{-,noise}$

(c) $c_{reg,-}$ (d) $c_{reg,noise}$

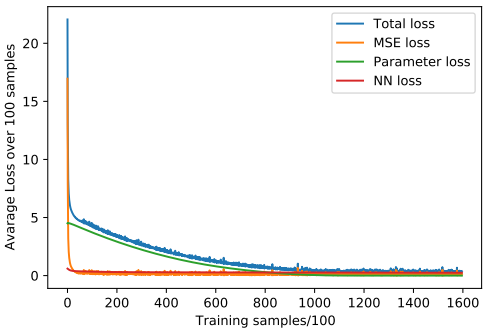
Figure 4.2: The learnable parameter value of ρ_W over the number of training samples considered. The empirical mean value of ρ_W is plotted for reference.



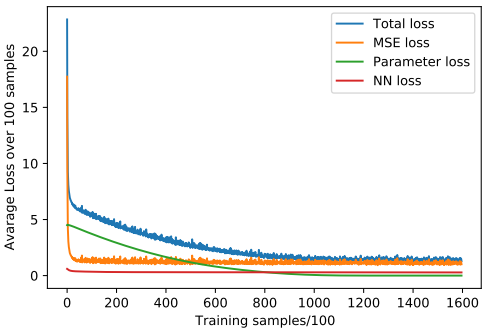
(a) $c_{-,-}$



(b) $c_{-,noise}$

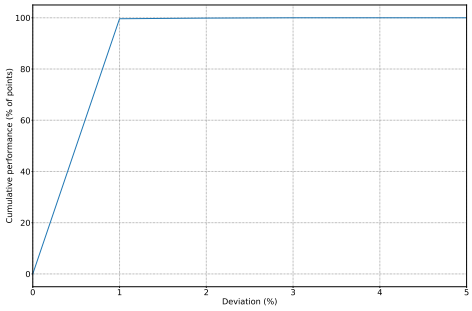


(c) $c_{reg,-}$

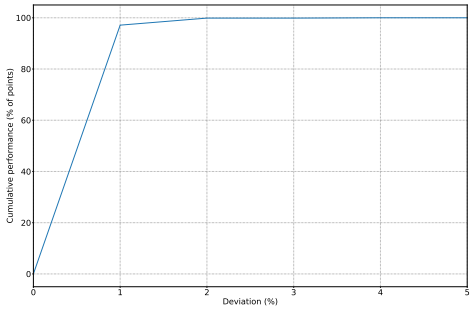


(d) $c_{reg,noise}$

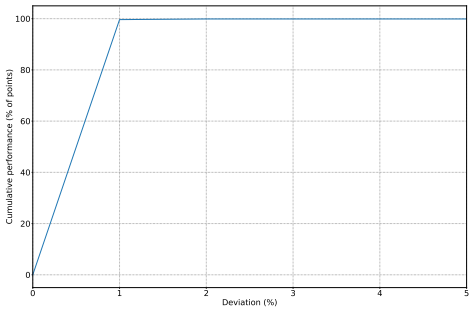
Figure 4.3: Training loss as represented by total loss, mean squared error loss, learanable parameter loss and neural network parameter loss. The loss has been smoothed out by taking the mean over every 100 iteration.



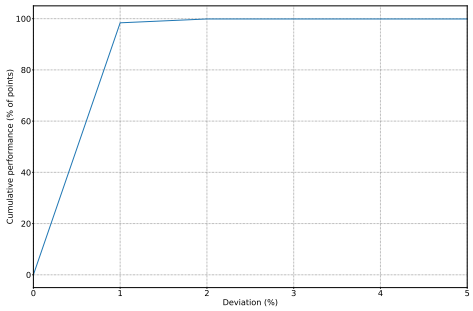
(a) $c_{-,-}$



(b) $c_{-,noise}$

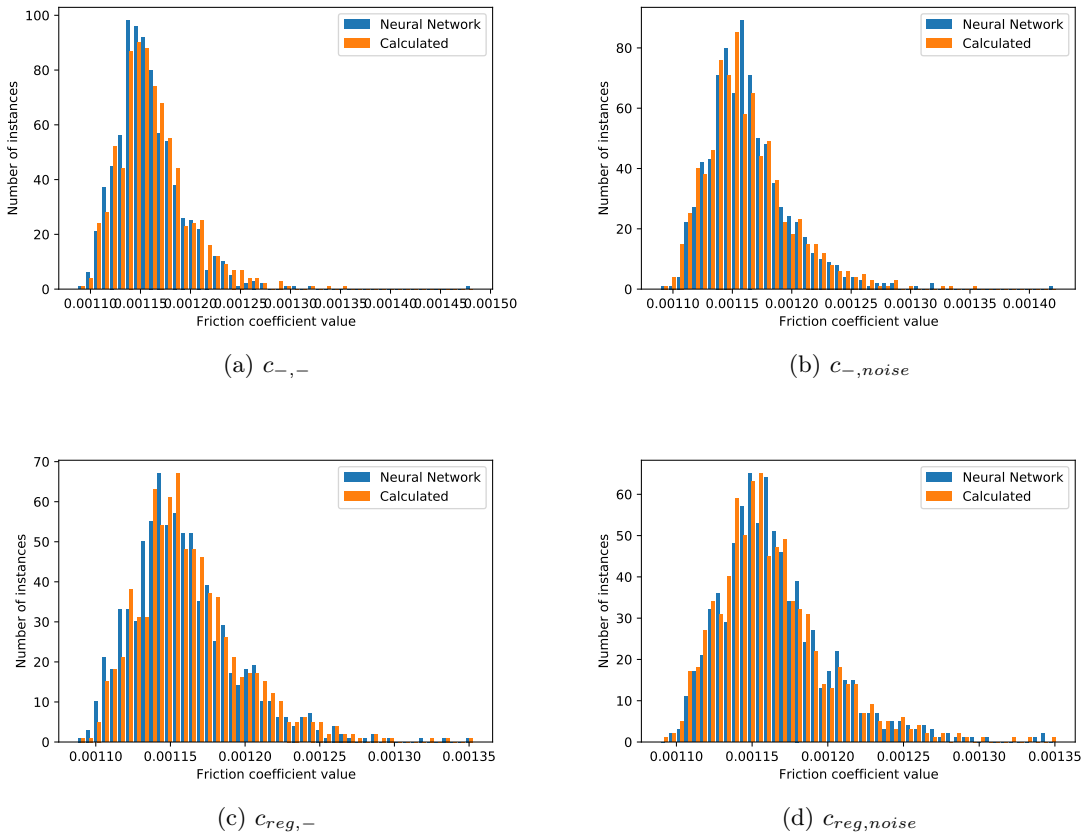


(c) $c_{reg,-}$



(d) $c_{reg,noise}$

Figure 4.4: Cumulative performance plot for the four model configurations.



(a) $c_{-,-}$

(b) $c_{-,noise}$

(c) $c_{reg,-}$

(d) $c_{reg,noise}$

Figure 4.5: Histogram of the friction factor f as calculated mechanistically and estimated by the neural network.

Chapter 5

Discussion

In this section, the results from Chapter 4 will be given a more in-depth analysis. We will start by analysing the effect of parameter regularization in Section 5.1 before analysing the effect of output noise on the system in Section 5.2. A short analysis of the total system performance will be conducted in Section 5.3. Finally, suggestions for future work will be given in Section 5.4.

5.1 The Effects of Parameter Regularization

First we will analyse the effect parameter regularization on the learnable parameters $\mathbf{\Phi}_{LP}$ have on the model performance. This is most clearly visible in Figures 4.1 and 4.2. We will now be comparing Figure 4.1a and Figure 4.1c. First, a key observation to make is that the model without regularization in Figure 4.1a has only been trained on 1/10 of the number of iterations. This is due to the fact that the model converges much faster without parameter regularization, and thus the number of training iterations needed are much fewer. This however does not change the fact that the parameters in the models without regularization have converged to a value close to their initialized value, which is different from their true underlying values. We see this for all the non-regularized models, both with and without noise for both learnable parameters, as seen in Figures 4.1a, 4.1b, 4.2a and 4.2b. We would expect this difference to affect the predictive performance of the model. We investigate this by looking at Figure 4.3. This figure shows the various loss components over the course of the training cycle. Looking closer at Figure 4.3a reveals that the mean squared error of the output still converges to ~ 0 quite rapidly. We know however from the Figures 4.1a and 4.2a that the learnable parameters are indeed not at the 'correct' values. This can also be seen in Table 4.3 where the final absolute errors of the learnable parameters after training are quite large, but the MSE of the model output $\dot{m}$ is still reasonable. One hypothesis might be that the neural network used to calculate the friction factor f compensates for this discrepancy. We investigate this hypothesis further by comparing the output of the neural network without regularization shown in Figure 4.5a and with regularization in Figure 4.5c. By paying close attention to the

neural network output (colored blue), it can be seen that without regularization the estimated friction factor $\hat{f}$ has a somewhat lower value than that calculated with regularization. Since the two distributions are not the same, it suggests that the neural network has indeed compensated for the error in the learnable parameters. The difference in correctness of the estimated $\hat{f}$ between the two model configurations can also be seen in Table 4.3. As now expected, the mean square error of the friction estimation over the test set is higher for the model without regularization, where we know the learnable parameters to have an error. Consequently, it would seem that the hybrid model successfully exhibits some of the adaptable characteristics associated with data-driven models.

Another consideration to make regarding the impact of regularization on the model is the training iterations needed to achieve reasonable performance. As mentioned above, without parameter regularization of the learnable parameters the model converges much faster, and thus fewer training epochs are necessary. This can clearly be seen by comparing Figure 4.3a and 4.3c. The model with regularization $c_{-,-}$ has been trained on 10 times more iterations than the model without regularization $c_{reg,-}$. This is done so that the learnable parameters can converge to their correct values. This convergence can be confirmed by looking at the absolute errors of the parameters for the regularized model configurations in Table 4.3, and also in the plots 4.1c, 4.1d, 4.2c and 4.2d. The convergence might have been achieved quicker however by tuning the hyperparameters. As mentioned in Section 3.6.3, the regularization coefficients $\boldsymbol{\lambda}_{LP}$ are calculated using the measurement noise σ_e. When regularization is on, this parameter is considered a hyperparameter, and can be tuned. In the results seen in Section 4 this parameter was set to $\sigma_e = 5$. A higher value for the noise might have yielded faster convergence. The learning rate was also experimented with, but was found to quickly cause divergent behavior in the model with too high values. For this dataset we were able to find a compromise between training time and stability, but for real-world datasets the data points may be more noisy and more limited, necessitating more thorough hyperparameter tuning.

Finally, the effect of parameter regularization on the total performance of the model will be considered. Taking a look at Figure 4.4 reveals that regularization does not seem to have a significant impact on the noise-less models, but does improve the performance slightly when noise is considered. Of course, all the performance plots in Figure 4.4 are already quite good, so maybe the impact would have been clearer with a more challenging dataset. Furthermore, the parameters were initialized close to the mean values. If they had been initialized further away, the model without regularization might have had a harder time finding correct system outputs. Table 4.3 also shows the mean squared error for the model output $\dot{m}$ for the different model configurations. Interestingly, when noise is not considered, the regularized model has a slightly worse performance on the test set. However, in accordance with the result we just found, regularization seems to benefit the model when trained on a noisy output. Regardless, having regularization is beneficial to the interpretability of the model. As engineers, we want to make sure that the model is using parameters that make physical sense, so regularizing the learnable parameters gives more confidence in the model even though it might perform slightly worse on perfect datasets.

5.2 The Effects of Output Noise

Next, we will investigate the effect of the output noise on the model performance. Looking at the learnable parameters in Figure 4.1 and Figure 4.2, it is clear that output noise has very little effect on the behaviour of the learnable parameters. However, by comparing Figure 4.1c and Figure 4.1d closely, it can be seen that the parameter converges slightly slower when noise is introduced. Overall though, the behaviour characteristic is much the same. This is promising, seeing as this suggests the model might be able to handle higher degrees of noise on the dataset.

A more clear image of the effects of noise on the system can be observed in Figure 4.3. Comparing the configurations with and without noise, it is clear that the MSE is higher for both configurations with noise. This is especially clear when comparing Figure 4.3a and 4.3b. In the configuration $c_{-,noise}$ the MSE error is not able to reach ~ 0. This would suggest that the model accuracy is decreased with the introduction of noise. This suspicion is confirmed in Table 4.3, where the MSE of the model output $\dot{m}$ is higher for both configurations with noise when compared to the configurations without noise. In order to investigate further, we will now focus on the model configuration with both noise and regularization $c_{reg,noise}$. As mentioned above, the MSE loss in Figure 4.3d does not converge to ~ 0. However, the learnable parameters do converge to their true values, as seen in Figure 4.1d and 4.2d. This would suggest that the cause of the constant loss delta is to be found in the neural network component of the hybrid model. Interestingly, this corresponds to the findings presented in Figure 4.5. The distribution of the estimated friction coefficient $\hat{f}$ in Figure 4.5d is quite skewed in comparison with the noise-less distribution found in Figure 4.5c. This suggests that the neural network is indeed the cause of the MSE loss.

It can be tempting to conclude that the model performance is weakened by the inclusion of noise. This is made quite apparent both from the cumulative plots in Figure 4.4 and the outputs in Table 4.3. However, a more careful analysis of the performance metrics may be in order. Specifically, when noise is introduced to the system, it is done by adding the noise to the output $\dot{m}$. Thus, when training the model, the assumed correct output of the model is $\dot{m}_{noise}$. Crucially, the noise is not added to the model inputs. The mechanistic model component will naturally generate a noise-less output given noise-less inputs. However, the model is expected to calculate an output with noise, so this discrepancy is compensated for in the neural network. Consequently, we would expect there to be a significant mean squared error between the model output and the noisy data output, seeing as the mechanistic model part of the hybrid model produces a noise-less output. This MSE is indeed present in both configurations with noise, as seen in Figures 4.3b and 4.3d. An interesting follow-up to this result would be to train the model on noisy data, but evaluate the model performance on noise-less data.

5.3 Overall Results

Overall, the results are promising. We have shown that the model adapt well to both incorrect input parameters, as well as noisy measurements. Although not perfect, the most divergent result from Table 4.3 is still within 0.3% of the average mass flow in the dataset. The most promising results however, might be the ones shown in Figure 4.5. The fact that the

distributions of the estimated and calculated friction coefficients are so similar, even though the estimated $\hat{f}$ is not estimated using $\dot{m}$, suggests that the hybrid model may work well on more complicated models and datasets. It should be mentioned that the current model performance might be somewhat inflated due to the simplicity of the dataset.

Another important result to consider is that of identifiability (Hotvedt et al., 2020b). From Figures 4.3c and 4.3d, it is clear that the model may reach a similar performance level for many different values of the learnable parameters. This does obscure some of the physical interpretability of the model.

5.4 Future Work

Having presented the results of this project so far, places for improvement have also been revealed. These improvements may come in future works, and some suggestions for improvements include:

- As mentioned in Section 5.2, evaluating a model trained on noisy data with noise-less data will show more clearly the actual performance of the hybrid model. Furthermore, this would also reveal how susceptible the data-driven model component is to noisy training data in terms of performance loss.
- The model can be initialized with learnable parameters further from their true value. This would check the validity of model configurations without regularization more thoroughly.
- The learnable parameters used in this implementation were ρ_O and ρ_W, as discussed in Section 3.6.3. However, investigating how increasing the level of 'hybridity' by including more learnable parameters would impact the model performance might be an interesting experiment. Candidates for the new learnable parameters might be the mass fractions ϵ_O, ϵ_W and ϵ_G.
- The dataset used in this project was generated using a similar first-principle model as the one used in the hybrid model, as discussed in Section 3.1. However, using a dataset generated on a more sophisticated model, or even better, real production data might reveal more of the limitations or advantages of the hybrid model suggested here.

Chapter 6

Conclusion

In this project a hybrid model of a production wellbore was created, combining a mechanistic first-principle model with a data-driven model component. Using a dataset generated on first-principles, different configurations of the model implementation was trained and evaluated with regards to training- and prediction performance. The prediction results are promising, indicating that the model has maintained some of the adaptability of data-driven models, while still maintaining high accuracy and resistance to noise. The data-driven model is also able to correctly predict the output $\hat{f}$ without using information on the output $\dot{m}$, as was done when generating the dataset. This suggests that this modeling technique may be adaptable to more advanced datasets and models. In conclusion, this project successfully lays the groundwork for future projects where more advanced wellbore models may be estimated using the hybrid modeling approach presented here.

Bibliography

Awad, M. and Muzychka, Y. (2008). Effective property models for homogeneous two-phase flows. *Experimental Thermal and Fluid Science*, 33(1):106 – 113.

Bikmukhametov, T. and Jäschke, J. (2020). First principles and machine learning virtual flow metering: A literature review. *Journal of Petroleum Science and Engineering*, 184:106487.

Brennen, C. E. (2005). *Fundamentals of Multiphase Flow*. Cambridge University Press.

Cengel, Y. A. and Cimbala, J. M. (2014). *Fluid Mechanics: Fundamentals and Applications, 3rd ed.* McGraw Hill.

Chaudhry, A. U. (2004). Chapter 2 - fundamentals of reservoir oil flow analysis. In Chaudhry, A. U., editor, *Oil Well Testing Handbook*, pages 13 – 43. Gulf Professional Publishing, Burlington.

Corneliussen, S., Couput, J.-P., Dahl, E., Dykesteen, E., Frøysa, K.-E., Malde, E., Moestue, H., Moksnes, P. O., Scheers, L., and Tunheim, H. (2005). Handbook of multiphase flow metering.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

Guo, B., Lyons, W., and Ghalambor, A. (2007). *Petroleum Production Engineering, A Computer-Assisted Approach*. Elsevier.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

Hotvedt, M., Grimstad, B., and Imsland, L. (2020a). Developing a hybrid data-driven, mechanistic virtual flow meter – a case study.

Hotvedt, M., Grimstad, B., and Imsland, L. (2020b). Identifiability and interpretability of hybrid, gray-box models.

Jansen, J.-D. (2015). *Nodal Analysis of Oil and Gas Wells – Theory and Numerical Implementation*. Addison-Wesley, Delft University of Technology (TU Delft) Department of Geoscience & Engineering, The Netherlands.

Kanin, E., Osiptsov, A., Vainshtein, A., and Burnaev, E. (2019). A predictive model for steady-state multiphase pipe flow: Machine learning on lab data. *Journal of Petroleum Science and Engineering*, 180:727 – 746.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511.

S.L., D. and C.A., H. (2010). *Fluid Mechanics and Thermodynamics of Tur- bomachinery, 6th ed.* Elsevier.

Solle, D., Hitzmann, B., Herwig, C., Pereira Remelhe, M., Ulonska, S., Wuerth, L., Prata, A., and Steckenreiter, T. (2017). Between the poles of data-driven and mechanistic modeling for process operation. *Chemie Ingenieur Technik*, 89.

Sutton, R. P. (1985). Compressibility factors for high-molecular-weight reservoir gases. SPE Annual Technical Conference and Exhibition.

von Mises, R. and Friedrichs, K. O. (1971). *Applied Mathematical Sciences 5: Fluid Dynamics*. Springer.