Master's thesis

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

Petter Hjørungnes Jacobsen

Exploring the potential for improved performance of flow metering using hybrid modeling

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Imsland
Co-supervisor: Mathilde Hotvedt
June 2022

NTNU
Kunnskap for en bedre verden

Petter Hjørungnes Jacobsen

Exploring the potential for improved performance of flow metering using hybrid modeling

Master's thesis in Cybernetics and Robotics
Supervisor: Lars Imsland
Co-supervisor: Mathilde Hotvedt
June 2022

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

NTNU
Norwegian University of
Science and Technology

Abstract

This master's thesis researches the application of hybrid flow modeling in oil and gas flow metering. As computational power and technological understanding have increased over the years, virtual flow meters (VFMs) are a growing field within flow metering. VFMs are increasing in popularity as technology improves. There are several advantages to virtual flow metering over physical flow meters, such as cheaper maintenance costs and easier deployment. However, the deployment is conditioned on reliable technology, and the industry wants more research on the virtual flow metering topic before full deployment in the industry. The goal of this thesis was to research if hybrid flow modeling has the potential to perform better than a standard data-driven flow model when modeling the flow through five different choke valves producing oil and gas. Two approaches to hybrid flow modeling were developed and tested. The first approach was to pretrain a neural network with generated fake data drawn from two different distributions, one uniform distribution, and one well-specific uniform design to fit the provided real data from the five wells. The other hybrid flow modeling approach was to deploy a physics-informed loss function (PILF) within a neural network. The PILF penalized the neural network for deviations from a mechanistic flow model in the training phase of the hybrid flow model. The results show that there is potential for improved performance by using a hybrid model in some situations. Four out of five wells showed improved performance when using a hybrid flow model compared to a standard data-driven flow model, and the best model had an improved performance of 35%. One of the wells showed no improvement when using a hybrid flow model. The results show potential for improved performance when utilizing hybrid flow modeling, but more research should be conducted, for example by researching generalizable characteristics across several choke valves.

Sammendrag

Denne masteroppgaven utforsker anvendelsen av hybrid flytmodellering i olje- og gass-flytmåling. Ettersom beregningskraft og teknologisk forståelse har økt de siste årene, er virtuelle flytmålere (VFM) et voksende felt innen flytmåling, og populariteten øker ettersom teknologien forbedres. Det er flere fordeler med virtuell flytmåling fremfor fysiske flytmålere, for eksempel billigere vedlikeholdskostnader og enklere installering. Utrullingen er imidlertid betinget av pålitelig teknologi, og industrien ønsker mer forskning på temaet. Målet med denne oppgaven var å undersøke om hybridmodellering har potensial til å yte bedre enn en tradisjonell datadrevet modell når man modellerer flyten gjennom fem forskjellige strupeventiler som produserer olje og gass. To tilnærminger til hybridmodellering ble utviklet og testet. Den første tilnærmingen var å forhånds-trene et nevralt nettverk med kunstig generart data hentet fra to forskjellige distribusjoner, en enhetlig distribusjon og en brønnspesifikk distribusjon designet for å passe til de utleverte ekte dataene fra de fem brønnene. Den andre tilnærmingen til hybridmodellering var å implementere en fysikk-informert tapsfunksjon (PILF) i et nevralt nettverk, og straffe avvik fra en mekanistisk flytmodell i treningsfasen til hybridmodellen. Resultatene viser at det er potensial for forbedret ytelse i enkelte situasjoner. Fire av fem brønner viste forbedret ytelse ved bruk av en hybridmodell sammenlignet med en enkel datadrevet modell, og den beste modellen hadde en forbedret ytelse på 35 %. En av brønnene viste ingen forbedring ved bruk av hybridmodellering. Resultatene viser potensial for forbedret ytelse ved bruk av hybrid flytmodellering, men mer forskning bør gjennomføres, blant annet for å oppdage mulige generaliserbare egenskaper på tvers av flere strupeventiler.

Preface and Acknowledgments

The deliverance of this master's thesis completes a degree in Master of Technology at the Norwegian University of Science and Technology (NTNU) under the Department of Engineering Cybernetics in Trondheim in the spring of 2022.

The master thesis was supervised by Professor Lars Imsland and co-supervised by ph.d. candidate Mathilde Hotvedt. Your feedback, guidance, patience, and proofreading are very much appreciated. Solution Seeker AS provided the data used to explore the research question, and I hope my work is deserving of the trust you have given me.

When I received my certificate of apprenticeship in chemical processing in the spring of 2016, I actually felt quite lucky that the job market was rough, and the only viable option for me was to apply for higher education. The first stop was a precourse in mathematics and physics. During the precourse, it was not clear to me that a master's degree in cybernetics and robotics was the master's degree I would apply for, but due to one of the best teachers I have ever been taught by, Vegard, did it became clear that this master's was the right for me. These six years at NTNU in Trondheim have been the best years of my whole life.

I have met a lot of people that I am incredibly grateful for having met, both on campus, at parties, and especially at the student society. I would especially like to mention Jo, Hanna, Even, Oskar, Sif, Frida, and Helga, for keeping me happy, telling me what to do and not do, teaching me, drinking beer with me, discussing with me, and giving me the opportunities you have given me. Without you, my time in Trondheim would not have been as rich in memories and experiences as it has been. I am forever grateful.

Finally, I would like to thank my family for their endless support and cheering. I look forward to the increased amount of time we will spend together now that my education is finally complete.

Trondheim, NTNU main building
Petter Hjørungnes Jacobsen
June 2, 2022

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

NCS	Norwegian continental shelf
FPSO	Floating Production, Storage and Offloading
GD	Gradient descent
i.i.d	independent and identically distributed
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MAP	Maximum a posterior
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood Estimation
MPFM	Multiphase flow meter
MSE	Mean square error
NN	Neural network
PILF	Physics-Informed Loss-Function
PINN	Physics-informed neural network
ReLU	Rectificed linear unit
RNN	Recurrent neural network
SGD	Stochastic gradient descent
VFM	Virtual flow meter
VI	Variational Inference

Greek letters

α	Step size / learning rate
ε	Measurement noise

η	Mass fraction	
$\boldsymbol{\theta}$	Parameters in neural network	
$\hat{\theta}$	Parameter estimate	
κ	Adiabatic gas expansion coefficient	
λ	Regularization factor	
λ_H	Regularization factor of hybrid term	
μ_ϵ	Mean of measurement noise	
ρ	Density	$\mathrm{kg/m^3}$
σ	Standard deviation	
σ_ϵ	Standard deviation of measurement noise	

Symbols

A	Area	$\mathrm{m^2}$
a	Activation function	
$\boldsymbol{b}$	Bias vector	
$\mathcal{B}$	Batch size	
C_D	Discharge coefficient	
$\mathcal{D}$	Dataset	
E	Number of iterations	
$f_{\boldsymbol{\theta}}$	Parametric model	
J	Objective function	
L	The number of layers in neural network	
$\dot{m}$	Total multiphase flow rate	kg/s
M	Molar mass	kg/mol
$\mathcal{M}$	Step direction	
$N_{\mathcal{D}}$	Number of observations in the training dataset	
p	Pressure	Bar
p_r	Pressure ratio	
$P_{\boldsymbol{\theta}}$	Number of model parameters	
Q	Volumetric flow rate	$\mathrm{m^3/s}$

R	Universal gas constant	$\mathrm{J}/(\mathrm{K}\cdot\mathrm{mol})$
T	Temperature	K
u	Choke opening	%
$\boldsymbol{W}$	Weight matrix	
$\boldsymbol{X}$	Design matrix	
y	Output	
$\hat{\mathrm{y}}$	Output estimate	
Z	Gas compressibility factor	
$\nabla_{\boldsymbol{\theta}}$	First-order gradient with respect to parameters	

Chapter 1

Introduction

Countless products in our daily lives contain hydrocarbons from the oil and gas industry. From medicines, plastics, artificial flavors, artificial dyes, and artificial smell to lubricants, gasoline, and asphalt [1]. Before products originating from hydrocarbons reach the consumer or business market, they need to be processed, refined, and produced. In Norway, all oil and gas production is carried out offshore on the Norwegian continental shelf (NCS) [2]. Most of the production occurs on a variety of oil rigs, as can be seen in Figure 1.0.1. As brand new oil rigs are becoming increasingly rare, subsea oil and gas installations have become increasingly regular in the industry to keep costs down. They often reroute hydrocarbon from newer reservoirs to existing hydrocarbon production infrastructure [3].

Figure 1.0.1: Stock photo of an oil platform. Picture retrieved from Wikimedia commons.

Regardless of where the oil and gas production occurs, the production needs to be controlled and monitored to maintain a safe and stable operation. To achieve this, process data are used to control the flow rate through the production equipment. The process data is gathered with hundreds, if not thousands, of sensors that continuously monitor production variables. Disregarding fire and gas monitoring, one of the most critical variables to control with regards to production is the flow of matter through the inlet pipe [4].

To control the flow rate through the inlet, the flow needs to be measured such that the operators know in what direction to throttle the valve that controls the flow rate. The flow rate measurement may be done by a physical Multiphase flow meter (MPFM), by a

Virtual flow meter (VFM), or by well-testing. These measurements are commonly referred to as flow metering schemes. Most oil rigs produce both oil and gas, and one of the flow metering schemes mentioned above is required to achieve a satisfactory measurement that meets the necessary requirements. A MPFM does not always work optimally. MPFMs are expensive, and require frequent re-calibrations which include well intervention [5, 6]. Since MPFMs require frequent maintenance, and well-testing requires well (and therefore production) intervention, alternative ways of measuring flow rate are being researched, such as VFMs.

A VFM is a soft-sensor that estimates the flow rate by developing a model that interprets other measurements around the desired flow rate measurement location, such as pressure, temperature, choke valve position, and other variables [7, 8]. Mechanistic flow models derived from mass- and energy balances and data-driven flow models such as neural networks have been in operation for quite some time. However, recent research has indicated a potential for increased performance with VFMs that utilize a hybrid flow model instead of a data-driven flow model or mechanistic flow model [9, 10]. Therefore, hybrid flow modeling is increasing in popularity as oil companies are on the lookout for technology that may reduce costs and increase the lifespan of already existing oil fields.

Data-driven flow models for oil and gas production chokes show the potential for reliable flow metering in the oil and gas industry [7]. However, improvements are still necessary for the industry to fully adopt the technology. Hybrid flow modeling, a combination of a data-driven flow model and a mechanistic flow model, is possibly the key to improving the performance of data-driven flow models. Suppose that the parameters of a hybrid flow model are configured correctly. In that case, the model may extract the best of both models and show significantly improved results compared to a data-driven flow model.

In July 2020, an extensive literature search was conducted by Willard *et al.* [11] where different kinds of machine learning models, including hybrid modeling, were examined. Willard *et al.* [11] dives into 228 research papers where 99 of them fall into the three categories "Physics-Guided Loss Function", "Physics-Guided Initialization", and "Hybrid modeling". As hybrid modeling is a combination of other models, a base model within the hybrid model is usually apparent. Hybrid models are therefore often designed by using either a mechanistic model or a data-driven model as the base and then introducing elements of the other model into the base model [11].

When examining the 99 articles in the three categories mentioned, it becomes clear that very little research has been done on VFM in the oil and gas industry. Some research has been developed, such as Hotvedt *et al.* [8], Staff *et al.* [10], AL-Qutami *et al.* [12], Psichogios and Ungar [13], and Hotvedt *et al.* [14], but no research that develops a hybrid flow model by integrating a physics-informed loss function or pretraining a Neural network (NN) based on fake data generated by a mechanistic flow model.

1.1 Contribution

This thesis explores the potential for improved performance of a neural network-based data-driven flow model trained to estimate the flow rate through an oil and gas production

choke. This exploration is done by implementing two hybrid flow modeling approaches. The performance of the two hybrid flow model approaches (and their combination) is compared with several configurations of data-driven flow models. The combination of the two hybrid flow model approaches is also evaluated. Towards the end of the thesis, several directions for further work and development are proposed.

Therefore, the hypothesis for this thesis is that a hybrid flow model has the potential to outperform a traditional data-driven flow model. The hypothesis will be tested on five separate wells from an anonymous oil rig on the NCS. The data used in this thesis to carry out the experiments presented in Chapter 3 were provided by Solution Seeker AS [15].

1.2 Thesis outline

The thesis starts with Chapter 1, Introduction, where a short introduction to why petroleum production and petroleum products are essential. The importance of the choke valve as a vital part of the production pipeline is also presented. The background theory to follow the workflow of the experiments is presented in Chapter 2, Theory, which gives an introduction to the science behind mechanistic flow modeling, data-driven flow modeling, and hybrid flow modeling, as well as a short introduction to hydrocarbon production.

Chapter 3, Methodology and setup, present how the different experiments and research in this thesis are conducted and how the results are produced. The results are presented in Chapter 4, before being discussed in Chapter 5. A conclusion and several directions for potential future work are presented in Chapter 6, Conclusion and Future work.

Chapter 2

Theory

This chapter presents the background theory necessary to follow the methodology and discussion of results.

2.1 Hydrocarbon production

Hydrocarbon systems are production facilities that produce and process hydrocarbon products. An example of a hydrocarbon production facility is the onshore gas refinery Kårstø in Norway, seen in Figure 2.1.1. A facility like Kårstø typically preprocesses hydrocarbons by removing particles, quicksilver, hydrogen sulfides, and water before the hydrocarbons are refined into different products. The products are thereafter transported to the consumer market either by subsea pipelines or tank vessels.

Figure 2.1.1: An overview of the Kårstø gas processing refinery in Rogaland, Norway. Picture taken by Markus Johansson / © Equinor.

The origin of these hydrocarbons may vary, but all hydrocarbon production in Norway takes place off-shore. When hydrocarbons are produced, the flow of fluid from the offshore well reservoir is controlled by a choke valve, usually placed on the platform deck [4]. Several

reservoirs with different product compositions are often connected to the same platform, so there may be several choke valves that control several different flow rates. The purpose of choke valves is to control the flow rate to a set point decided by the operations main control room. The composition of the material produced can differ significantly from well to well but consists mainly of oil, gas, and water, often referred to as three different phases [4].

Accurate and precise control of the choke valve based on the total flow rate is vital for safe and stable operation. To achieve this, the total flow rate through the choke must be continuously monitored, and the measurements need to be accurate and reliable to maintain stable operation and flow control.

One of the most common ways of metering flow through an oil and gas producing well is with a MPFM [16]. However, the MPFMs are not perfect. Due to their downsides, such as lack of accuracy in complex multiphase flow modeling and vulnerability to flow obstructions, such as wax or hydrates as described by Falcone and Alimonti [17], alternatives such as VFMs are researched.

2.1.1 The choke valve

In an oil and gas producing facility, hundreds of valves are installed in the process to control the production. The pros and cons of different valves and their characteristics is a field too large to summarize in this thesis. In short, valves can be grouped into safety valves, control valves, and production valves [18], where all of them may play a crucial role in the safe operation of hydrocarbon production. Of all critical control valves in a hydrocarbon production system, the choke valve is one of the most exciting and essential as the choke valve directly controls the flow rate from the reservoir to the production facilities [4]. The production facility is often an offshore oil rig or an floating production, storage, and offloading (FPSO) unit. A schematic overview of a generic choke valve can be seen in Figure 2.1.2.

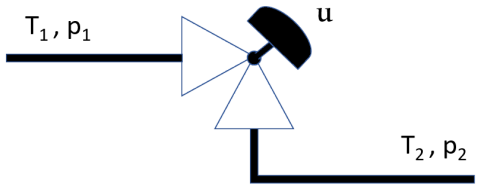


Figure 2.1.2: A simple illustration of a choke valve, with measurements T_1 and p_1 upstream the choke valve, and T_2 and p_2 downstream the choke valve. The percentage opening of the choke valve is denoted by u.

It is common to measure the temperature upstream and downstream of the choke valve, as illustrated in Figure 2.1.2. However, due to the pressure drop over the choke valve and the temperature variable naturally being a variable that takes some time to settle (also known as a slow characteristic process), the downstream temperature is often omitted due to uncertain and unstable measurements [19].

2.2 Neural networks

A neural network is a concept within machine learning that is specialized in processing historical data and is programmed to learn how the structures and correlations within the data are built up [20]. Suppose that the neural network perfectly learns the structure of the data. In that case, the neural network will output the correct answer according to the input, even though the network has never received any additional prior information about correlations between the variables in the data, but just the data in itself. It should be kept in mind that the neural network will not be able to learn any correlations that are not present in the training data. If the training data are of low quality, for example due to noise in the measurements, the output of the network may be correct according to the input data. However, it may not be desirable regardless because of the low quality of the data.

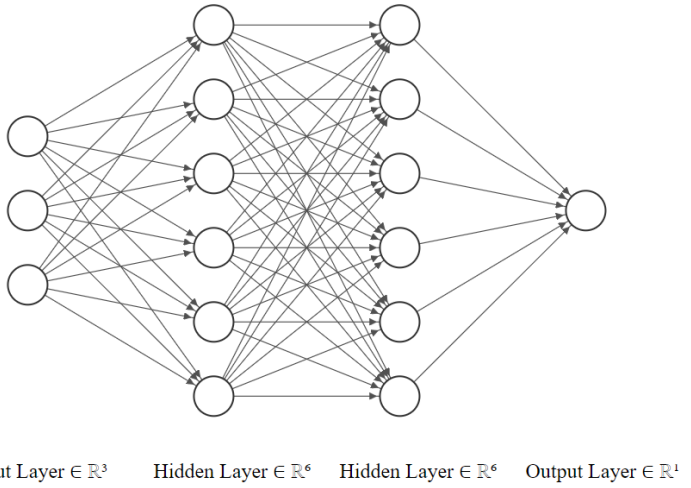


Figure 2.2.1: An example of how a neural network may look like with an input layer of width three, two hidden layers with width six, and an output layer with width one. When all nodes from the previous nodes is connected with all nodes in the next layer, the neural network is called a fully connected neural network.

The parameters inside the neural network, together with the input and output of the network, are an example of a data-driven model, as the neural networks aim to model the equation or process that generated the data based only on the data. A visualization of a neural network with an input width of three, an output width of one, and two so-called hidden layers (which are the layers between the input and the output layer) with a width of six, can be seen in Figure 2.2.1. The way the layers and nodes in the neural network are built is mathematically described in (2.2.1), where $\boldsymbol{x}$ is the input, $\hat{y}$ is the output, L is the total number of layers, $\boldsymbol{W}_i$ is a weight matrix containing the multiplicative parameters, $\boldsymbol{b}_i$ is a bias vector containing the bias parameters, and $a(*)$ is the activation function that maps the output from each node to the next [20]. A single node and the parameters related to it can be seen visualized in Figure 2.2.2. A neural network that only flows in the forward direction (i.e. without feedback within the network) and where every node in the previous layer is connected with every node in the next layer is called a fully connected feed-forward neural network [20].

$$
\begin{aligned}
&\text{Input layer } z_0 = \mathbf{x} \\
&\text{Hidden layer(s) } z_i = a\big(\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i\big), \quad i \in \{1, \ldots, L-1\} \\
&\text{Output layer } \hat{y} = \mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L.
\end{aligned} \tag{2.2.1}
$$

A generalized mathematical description of a neural network can be seen in (2.2.2). The parameters $\boldsymbol{\theta}$ contain the weights $\boldsymbol{W}$ and the biases $\boldsymbol{b}$ that correspond to the values seen in Figure 2.2.2 where w_i are the weights with which each input x_i is multiplied with, and b is the bias that is added to the sum before the activation function is applied to produce the output of the node. The activation function is a mathematical function that maps the output of the network to a certain range. One of the most common activation functions in use is called the Rectified linear unit (ReLU) and can be seen in (2.2.3) [21].

$$
\hat{y} = f\left(\boldsymbol{x}; \boldsymbol{\theta}\right) \tag{2.2.2}
$$

$$
a(x) = \max\left(0, x\right) \tag{2.2.3}
$$

The learning process in the neural network is achieved by updating the parameters $\boldsymbol{\theta}$ within the network according to a specified loss function and a specified parameter updating algorithm called back-propagation [22]. This will be discussed further in Section 2.2.1.

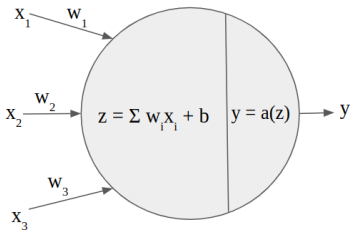


Figure 2.2.2: An illustration of how the output y of a node in the neural network is calculated, with a node input width of three, and an activation function a that maps z to y.

Before the parameters $\boldsymbol{\theta}$ may be updated, $\boldsymbol{\theta}$ needs to be initialized. In principle, $\boldsymbol{W_0}$ and $\boldsymbol{b_0}$ that the first generation of $\boldsymbol{\theta}$ represents can be initialized to any number. However, parameter initialization is a field of study with a lot of research involved, as initialization may have a big impact on the final performance of the model. Kumar [23] shows that certain initialization methods prove to be more advantageous than others, depending on the activation function with which the neural network is initialized.

Two common initialization methods are the He initialization and the Xavier initialization [24]. Both of these initialization methods consist of randomly drawing initial values from a normal distribution such that $\boldsymbol{\theta}_0 = \mathcal{N}\left(\mu, \sigma^2\right)$ with mean $\mu = 0$ and $\sigma^2 = v^2$, where v^2 is decided at a preliminary level according to the initialization scheme that one wants to follow. For the Xavier initialization, Glorot and Bengio [25] suggests an initialization where $v^2 = 1/N$ where N in the number of nodes that feed into each respective layer. However, He *et al.* [24] argues that the He initialization works better than the Xavier initialization when using the ReLU activation function, and argues and proposes $v^2 = 2/N$.

2.2.1 Learning and loss function

The learning process in the neural network occurs when the parameters in the network are updated according to the value of the loss function and the optimizer algorithm. The loss function seen in (2.2.4) measures how well the neural network performs, or in other words, how well it models the dataset. If the model performs poorly, the loss function will output a higher number, and if the model performs quite well, the loss function will output a lower number. There are several ways to configure a loss function, but the most common is the Mean square error (MSE) as shown in (2.2.4) where n is the number of data points, y_i is the true label, and $\hat{y}_i$ is the estimated label.

$$L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2 \tag{2.2.4}$$

In order to update the parameters $\boldsymbol{\theta}$ with the aim of improving the performance of the model, the performance of the model with the set of parameters $\boldsymbol{\theta}$ must be evaluated. To validate the performance of the model, all collected data are divided into three different datasets. These sets are disjoint and are called the training set, the validation set, and the test set [20]. During training, data from the training dataset are sent through the network and updated to improve the performance of the model. The loss calculated during training is called the training loss. To evaluate the performance of the model, data from the validation dataset (which the model has never processed before) is sent through the network and the loss that is calculated by the loss function is called the validation loss. The performance on the validation dataset allows the programmer to evaluate how well the model performed and may update the network configuration with the goal of improving the performance even more. When satisfactory performance is achieved, the configuration is saved, and the training dataset and the validation dataset are merged into a new dataset. When the final results of the experiments (i.e., the final performance of the neural network is about to be tested), the neural network is trained with the merged dataset with data from both the training dataset and the validation dataset, before it is tested on the test dataset. To evaluate the performance of the model, the error in the test dataset is referred to as the final performance of the model [20]. Out of all the data, a typical split between training data, validation data, and test data are to extract between 5% and 10% as test data (depending on the size of the total dataset) and split the rest of the data as 80% training data, and 20% validation data [26]. However, assessments need to be made from dataset to dataset.

The learning objective of the network can be modeled as an optimization problem, shown in (2.2.5) where $J(\boldsymbol{\theta})$ is the cost function given by (2.2.6), and $\hat{\boldsymbol{\theta}}$ are the parameter estimates.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{2.2.5}$$

$$J(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + R(\boldsymbol{\theta}) \tag{2.2.6}$$

The cost function $J(\boldsymbol{\theta})$ is the metric the optimizer will minimize during training of the model, and may include a regularization term $R(\boldsymbol{\theta})$ and the loss function $L(\boldsymbol{\theta})$ as shown in (2.2.6) [27]. The goal of the algorithm is to find the set of parameters $\boldsymbol{\theta}$ that minimize

the cost function $J(\boldsymbol{\theta})$, which implies that the output of the neural network is as close as possible to the output. However, if $J(\boldsymbol{\theta})$ becomes too small and too well adapted to the training data, overfitting to the training data is prone to occur and will make the model perform poorly on the unseen test data. The cost function (2.2.6) also has an $R(\boldsymbol{\theta})$ term called the regularization loss. The regularization loss term may be seen in (2.2.7), where $\boldsymbol{\theta}$ is the parameters, and λ is a regularization factor that is used to tune the regularization loss and hence used to minimize the overfitting. The regularization factor is typically relatively small in the range from 1 to 0.0001.

$$R(\boldsymbol{\theta}) = \lambda \sum_{i=0}^{n} ||\boldsymbol{\theta}_i||_2^2 \tag{2.2.7}$$

When the features of a data point have been sent through the network, the output is compared to the label of the data points by the loss function $L(\boldsymbol{\theta})$ as seen in (2.2.4). How many data points are sent through the network before the loss (and cost) is calculated is determined by the pre-defined batch size $\mathcal{B}$. The MSE in each batch is called batch MSE.

When the network has processed a batch size $\mathcal{B}$ number of data points and the batch cost is calculated, the network will calculate the gradient of each parameter in the network such that the parameters may be updated according to what minimizes the cost function $J(\boldsymbol{\theta})$.

The updating of the parameters $\boldsymbol{\theta}$ takes place in the optimizer according to the gradients calculated by the back-propagation algorithm using the chain rule [22]. A mathematical description of the parameter updating may be seen in (2.2.8), where α is the learning rate (sometimes referred to as step size), $\mathcal{M}$ is the set of equations that calculates in what direction to update the parameters $\boldsymbol{\theta}$, and $E/\mathcal{B}$ is the number of times the update occurs, where E is referred to as the number of epochs.

$$\hat{\boldsymbol{\theta}}_{i+1} = \hat{\boldsymbol{\theta}}_i - \alpha \mathcal{M}\left(\mathcal{B}_i, \hat{\boldsymbol{\theta}}_i\right), \quad i = 1, \ldots, E \tag{2.2.8}$$

When the entire training dataset has been processed, the model is tested on the validation dataset, and a new MSE is calculated for the validation dataset.

An example graph showing how the loss of training and validation data can decrease during neural network training can be seen in Figure 2.2.4. The red line marking the early stopping checkpoint will be discussed in Section 2.2.5.

2.2.2 Parameter estimation

Another way of mathematically describe the neural network seen in (2.2.2) can be seen in (2.2.9), where $\boldsymbol{\theta}$ are the model parameters, x_i is the input of the model, and ϵ_i is the measurement noise that is normally distributed with zero mean and standard deviation of σ_ϵ. The subscript i refers to the data point number i. For (2.2.9), the independent and identically distributed (i.i.d) assumptions are assumed valid [27].

When estimating a model, it is desired that the estimated model has high generalization properties. Let $\mathcal{D} = (\boldsymbol{X}_i, y_i)_{i=1}^{N_\mathcal{D}}$ be a dataset that will be modeled with the function $f_{\boldsymbol{\theta}}$. Then the data within $\mathcal{D}$ should have a certain distribution and richness such that the model produced when learning from $\mathcal{D}$ will be generalizable to a data point that may be produced by the same process that produced the data points in $\mathcal{D}$, but are not in $\mathcal{D}$ themselves.

$$\hat{y}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right) \tag{2.2.9}$$

The goal of estimating the parameters $\boldsymbol{\theta}$ is to make the estimates the neural network produces (based on a data points input features) as close to the data points label as possible. The measure of how close the output produced is to the correct values is calculated by the cost function (2.2.6), and the optimization problem of making $\boldsymbol{\theta}$ as optimal as possible given a specific dataset is described by (2.2.5).

From a Bayesian point of view, the a priori probability distribution $p(\boldsymbol{\theta})$ of the model parameters can be updated to a posterior model parameter distribution $p(\boldsymbol{\theta}|\mathcal{D})$ as shown in (2.2.10), where $p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood function and $p(\mathcal{D})$ is the true distribution of $\mathcal{D}$, sometimes referred to as the evidence.

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{2.2.10}$$

A conditional model such as (2.2.9) can be trained with both a Maximum likelihood estimation (MLE) and a Maximum a posteriori (MAP) estimation scheme. In MAP, the mode of the posterior parameter distribution is maximized, while in MLE, the parameters are derived by maximizing the likelihood function. The MLE problem may be derived as seen in (2.2.11).

$$\hat{\boldsymbol{\theta}}_{MLE} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}). \tag{2.2.11}$$

By using the i.i.d assumption, which states that the observations in the dataset $\mathcal{D}$ are independent of each other and identically distributed, the log-likelihood function of the model in (2.2.9) can be written as

$$\begin{aligned}
\log p(\mathcal{D}|\boldsymbol{\theta}) &= \log \prod_{t=1}^{N_\mathcal{D}} p(y_t|\mathbf{x}_t, \boldsymbol{\theta}) = \sum_{t=1}^{N_\mathcal{D}} \log p(y_t|\mathbf{x}_t, \boldsymbol{\theta}) \\
&= \sum_{t=1}^{N_\mathcal{D}} \log \left[\frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} e^{\frac{(y_t - f_{\boldsymbol{\theta}}(\mathbf{x}_t))^2}{2\sigma_\varepsilon^2}} \right] \\
&= -\sum_{t=1}^{N_\mathcal{D}} \frac{1}{2\sigma_\varepsilon^2} \left(y_t - f_{\boldsymbol{\theta}}(\mathbf{x}_t)\right)^2 - N_\mathcal{D} \log \sqrt{2\pi\sigma_\varepsilon^2}
\end{aligned} \tag{2.2.12}$$

where $N_\mathcal{D}$ is the number of data points in the dataset, which when (2.2.12) is inserted into (2.2.11) results in the estimation problem

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MLE} &= \arg\max -\sum_{t=1}^{N_{\mathcal{D}}} \frac{1}{2\sigma_{\varepsilon}^2} \left(y_t - f_{\boldsymbol{\theta}}\left(\mathbf{x}_t\right)\right)^2 - \underbrace{N_{\mathcal{D}} \log \sqrt{2\pi\sigma_{\varepsilon}^2}}_{\text{constant}} \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{N_{\mathcal{D}}} \frac{1}{2\sigma_{\varepsilon}^2} \left(y_t - f_{\boldsymbol{\theta}}\left(\mathbf{x}_t\right)\right)^2, \end{aligned} \tag{2.2.13}$$

where the constant term can be removed as it is independent of $\boldsymbol{\theta}$, and maximizing a negative expression is equivalent of minimizing a positive expression. The $\hat{\boldsymbol{\theta}}_{MLE}$ as seen in (2.2.13) is equal to the cost function (2.2.6) with the regularization term $R\left(\boldsymbol{\theta}\right)$ discarded, and the loss function $L\left(\boldsymbol{\theta}\right)$ as shown in (2.2.4).

To counteract the possibility of overfitting since MLE only minimizes the squared deviation between the measurements and the predictions, MAP estimation may be used to train the model. A derivation of MAP estimation may be seen in (2.2.14).

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MAP} &= \arg\max_{\boldsymbol{\theta}} p\left(\boldsymbol{\theta}|\mathcal{D}\right) = \arg\max_{\boldsymbol{\theta}} \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \arg\max_{\boldsymbol{\theta}} \log \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p\left(\boldsymbol{\theta}\right) - \underbrace{\log p\left(\mathcal{D}\right)}_{\text{constant}} \\ &= \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p\left(\boldsymbol{\theta}\right). \end{aligned} \tag{2.2.14}$$

If the priors $p\left(\boldsymbol{\theta}\right)$ are assumed to follow the Gaussian distribution $\theta_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right),\ i = 1, \ldots, P_\theta$ where P_θ is the number of parameters in the model, the log-prior parameter distribution is given as

$$\log p\left(\boldsymbol{\theta}\right) = \log \prod_{i=1}^{P_\theta} p\left(\boldsymbol{\theta}_i\right) = -\sum_{i=1}^{P_\theta} \frac{1}{2\sigma_i^2} \left(\theta_i - \mu_i\right)^2 - P_\theta \log \sqrt{2\pi\sigma_i^2}. \tag{2.2.15}$$

Inserting (2.2.12) and (2.2.15) into (2.2.14), the MAP estimation problem becomes

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MAP} &= \arg\max_{\boldsymbol{\theta}} \log p\left(\mathcal{D}|\boldsymbol{\theta}\right) + \log p\left(\boldsymbol{\theta}\right) \\ &= \arg\max_{\boldsymbol{\theta}} -\sum_{t=1}^{N_{\mathcal{D}}} \frac{1}{2\sigma_{\varepsilon}^2} \left(y_t - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t\right)\right)^2 - \underbrace{N_{\mathcal{D}} \log \sqrt{2\pi\sigma_{\varepsilon}^2}}_{\text{constant}} \\ &\quad - \sum_{i=1}^{P_\theta} \frac{1}{2\sigma_i^2} \left(\theta_i - \mu_i\right)^2 - \underbrace{P_\theta \log \sqrt{2\pi\sigma_i^2}}_{\text{constant}} \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{N_{\mathcal{D}}} \frac{1}{\sigma_{\varepsilon}^2} \left(y_t - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t\right)\right)^2 + \sum_{i=1}^{P_\theta} \frac{1}{\sigma_i^2} \left(\theta_i - \mu_i\right)^2. \end{aligned} \tag{2.2.16}$$

where the two constant terms may be discarded as they are independent of $\boldsymbol{\theta}$. The $\hat{\boldsymbol{\theta}}_{MAP}$ estimation as seen in (2.2.16) is equal to the cost function (2.2.6) with the regularization term $R(\boldsymbol{\theta})$ equal the second term in (2.2.16), and the loss function $L(\boldsymbol{\theta})$ as shown in (2.2.4).

From (2.2.16) we see that MAP estimation is a trade-off between minimizing the squared errors (the first term) and the deviation of the parameters from their prior mean (the second term). On the contrary, MLE as presented in (2.2.13) exclusively minimizes the squared errors. Since the MLE approach is prone to overfitting, MAP estimation is often preferred.

Comparing the final MAP estimation expression (2.2.16) multiplied with $\sigma_\varepsilon^2/N_\mathcal{D}$ with (2.2.5) when (2.2.6), (2.2.7), and (2.2.4) are inserted, we get

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=0}^{n} ||\boldsymbol{\theta}_i||_2^2. \tag{2.2.17}$$

where $\lambda = \sigma_\varepsilon^2/N_\mathcal{D}\sigma^2$, the prior mean distribution of the parameters is $\mu_i = 0$, and the prior standard deviation of the parameters is $\sigma_i = \sigma$. The equation (2.2.17) is a common implementation of parameter adaptation in neural networks where λ is used as a regularization factor. The MAP estimation scheme seen in (2.2.17) will be used in the experiments conducted in this thesis.

There are some disadvantages of MLE and MAP, such that neither of them includes uncertainty in the variables but rather regards the modes of the variable distributions. There are also more complex alternatives, such as Markov Chain Monte Carlo (MCMC) and Variational Inference (VI) [28], but none of these will be explored further in this thesis.

2.2.3 Implementing the parameter estimation

For each iteration in the neural network where the parameters $\boldsymbol{\theta}$ are updated, the direction and magnitude of the update must be decided before the update occurs. The update is based on the estimating schemes discussed in the previous section. Consider a model

$$\hat{y}_t = \boldsymbol{x}_t^T \boldsymbol{\theta} \tag{2.2.18}$$

where MLE is used to estimate the variables. When gathering all inputs $\boldsymbol{x}$ in a matrix $\boldsymbol{X} \in \mathbb{R}^{N_\mathcal{D} \times P_\theta}$ and all outputs $\hat{y}_t$ in the vector $\boldsymbol{Y} \in \mathbb{R}^{N_\mathcal{D}}$, the MLE of the model becomes

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}). \tag{2.2.19}$$

The solution to (2.2.19) becomes

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) = 0, \\ \Rightarrow\ &2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}\boldsymbol{y} = 0, \\ \Rightarrow\ &\hat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}, \end{aligned} \tag{2.2.20}$$

where a unique solution exists if $\boldsymbol{X}^T\boldsymbol{X}$ is of full column rank [29]. For highly complex and nonlinear systems, the solution to (2.2.18) may be difficult and very demanding to calculate exactly. Therefore, iterative gradient-based numerical optimization algorithms are the most common to use when solving such models [30].

Although there exist complex optimization algorithms that, in theory, should perform quite well, first-order optimization algorithms are the most common due to the trade-off between computational complexity and accuracy [20]. One of the most common first-order optimization algorithms is Gradient descent (GD) that updates the parameters based on the gradient of the parameters $\nabla_{\boldsymbol{\theta}}$ for a given batch size $\mathcal{B}$. The optimization of GD is called Stochastic gradient descent (SGD) if the batch size is set to one, batch GD if the batch size includes the entire training dataset, and minibatch GD if the batch size is between one and the entire dataset [20].

A disadvantage of (S)GD is that both are prone to not utilizing the full potential of the data set in terms of finding an as optimal minimum as possible and may not converge if the parameter space is of a high dimension and contains many local minima. One of the many other widely used optimization algorithms is called Adam [31]. Kingma and Ba [31] must be consulted for the technical details within the Adam optimizer algorithm, but in short takes Adam advantage of momentum in the gradients of the parameters, which gives the algorithm the potential for overshooting a local minimum if there is a lower minimum close by. An illustration of how Adam may find a more optimal minimum than (S)GD may be seen in Figure 2.2.3.

A stepwise overview of the Adam optimization algorithm can be seen in Appendix A.1.

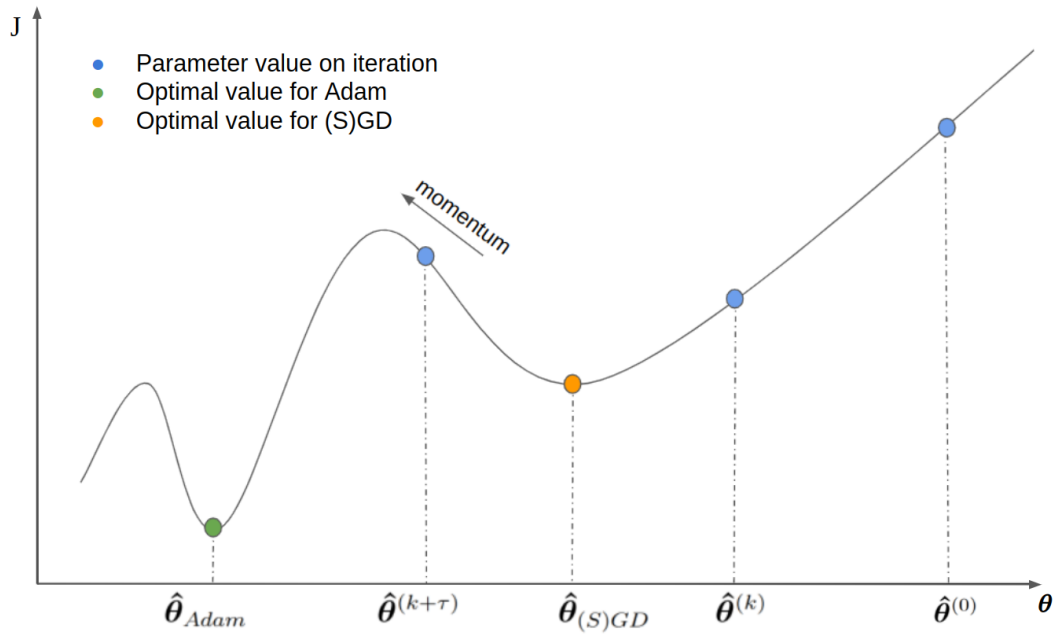


Figure 2.2.3: An illustration of how different optimization algorithms may find different optimums. The (S)GD may converge to the local minimum at the orange dot, but an optimizer like Adam may be able to use the gradient momentum to overshoot the local minima and converge to the global minima marked with a green dot.

2.2.4 Hyperparameters

Not all parameters in the neural network are updated during training. Such parameters are called hyperparameters and are defined before the training starts. Examples of hyperparameters can be the learning rate α, the regularization factor λ, the maximum number of epochs E, the depth and width of the neural network, and others [20]. All of them have in common that they are generally kept constant during the entire training and testing of the model.

There exists research such as Baik *et al.* [32] that has experimented with updating of hyperparameters. The so-called learning rate schedulers (which update the learning rate during training) exemplify how the inclusion of hyperparameters during training develops in a more dynamic direction. However, stationary hyperparameters from start to finish are the most common approach. When training and testing are over, the parameters $\boldsymbol{\theta}$ are the central part of the model produced. At the same time, the hyperparameters (except for the width and depth of the neural network) are not part of the final model. Tuning of the hyperparameters based on the performance of the model is common to use as a method of improving the performance of the neural network. It is crucial to not look at the model performance on the unseen test data during the tuning of the hyperparameters. Evaluating the performance of the model on the test data during training and tuning is known as data leakage. It may result in an unrealistic good performance on the test data, which will make the model seem better than it actually is [20].

2.2.5 Early stopping

Early stopping is a neural network regularization technique used to avoid overfitting the data to which the algorithm is trying to adapt [20]. The early stopping algorithm keeps track of the training algorithm's validation loss in each epoch. It would increment a counter if the validation loss of an epoch were not better than the previous epoch. Suppose that the counter reaches a predefined limit (often referred to as the patience) without improvement. In that case, the neural network stops and returns the model parameters at the point where the validation loss was at its lowest (the red striped line in Figure 2.2.4) and returns those parameters as the finished model. If the validation loss decreases (from the last checkpoint) before the counter reaches the patience, the counter is reset to zero and the loop continues [20, 33]. Suppose the counter does not reach the patience before the algorithm has trained for the pre-defined maximum number of iterations. In that case, the model will stop and return the model parameters as the finished model.

If the algorithm does not use early stopping, the model may be overfitted to the training data and perform poorly on the validation data, as the model is too well adapted to the training data. In that case, the orange graph in Figure 2.2.4 would start to increase and the model would probably perform worse and worse on the validation data if the training continues. Regularization is a measure to avoid overfitting regardless of early stopping. However, regularization may not be sufficient and may result in a model that is too poorly adapted to the training data and hence performs poorly on the validation dataset. Therefore, early stopping may allow for a slightly lower regularization factor without the risk of overfitting.

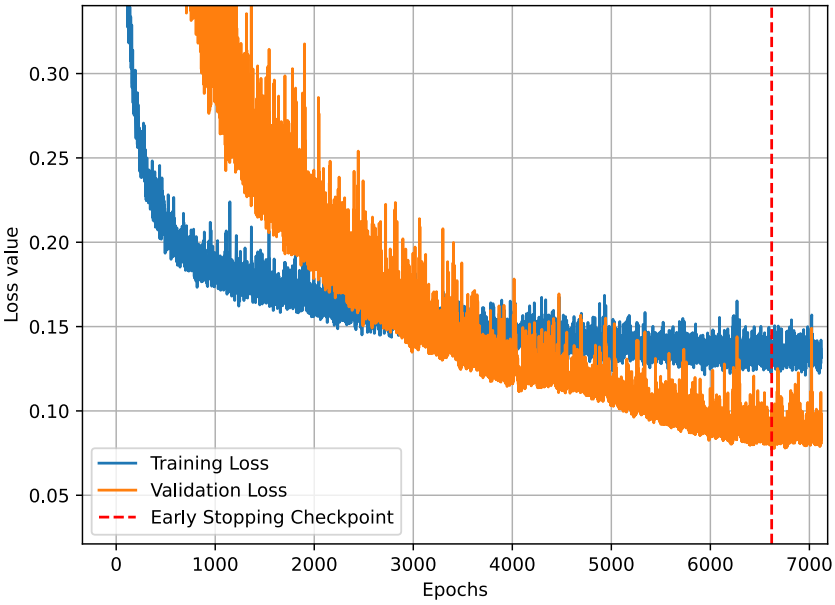


Figure 2.2.4: An example of a graph showing the validation and training loss. The red striped line marks the early stopping checkpoint.

2.3 Flow modeling

In this thesis, three approaches to flow modeling will be used. Mechanistic flow modeling as discussed in Section 2.3.1, data-driven flow modeling as discussed in Section 2.3.2, and a combination of data-driven flow modeling and mechanistic flow modeling will be elaborated on in Section 2.3.3. Mechanistic flow modeling is traditionally used, but as computing power and complexity in computers have increased, data-driven flow modeling has become more and more popular over the years. Research to find out whether it is possible to combine the best of both regimes has increased in popularity over the last few years [34, 35].

Flow models may be relatively simple to very complex. A flow model may be based on singlephase or multiphase flow, and some models estimate the total flow rate measured in mass, while others estimate the total flow rate in volume. These varieties of flow models result in some models being better suited for some systems than others. The purpose of a flow model is almost always to measure how much flow that passes a point in a system, and is therefore often referred to as flow metering. The flow metering regime can be illustrated as seen in Figure 2.3.1 and will be briefly discussed in the following sections.

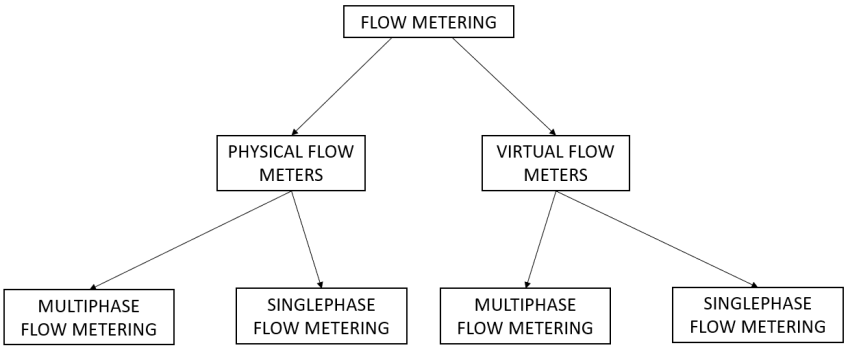


Figure 2.3.1: A hierarchical illustration of the flow metering regime. The illustration shows that all flow meters are either physical or virtual, and that both of these either measures singlephase flow or multiphase flow.

2.3.1 Mechanistic flow modeling

Many mechanistic flow models have been developed, as discussed by Guo *et al.* [4], and are sometimes referred to as white-box models as an analogy to the extent of prior knowledge. Mechanistic models are often derived from mass balances or energy balances, and several different influential physical phenomena and effects may be taken into account depending on the intended use of the model [4, 36]. Mechanistic flow models that can consider multiple phases are more complex than flow models that are based only on a single phase. Therefore, single-phase mechanistic flow models are preferred in some situations because it may not be necessary to implement a more complex model. A simple flow model may, in many situations, be sufficient, as they are easy to implement and do not need in-depth knowledge about the system that is being modeled [37].

A simple flow model can be seen in (2.3.1) where Q is the total volumetric flow rate measured in m^3/s, C_D is the valve discharge coefficient, A is the inner cross-section area of the choke, ρ is the homogeneous density, and P_{wh} and P_{dc} are the pressure upstream and downstream of the choke respectively [38]. The more complex Sachdeva model can be seen in (2.3.2), where $\dot{m}$ is the total multiphase flow rate measured in kg/s, C_D is the valve discharge coefficient, A is the area of the inner cross-section of the choke, ρ_{dc} is the downstream choke density, p_{wh} is the wellhead pressure, p_{dc} is the downstream choke pressure, κ is the adiabatic gas expansion coefficient, $\eta_i, \rho_i, i \in \{G, O, W\}$ is the mass fractions and the gas densities of gas, oil, and water respectively, and p_r is the downstream to upstream pressure ratio [39]. A unit derivation for both models can be seen in Appendix A.2. Both the simple flow model and the Sachdeva flow model assume that the flowing matter is homogeneous and therefore has a constant density ρ estimated with (2.3.4), that there is no drag and no slip through the restriction, incompressible liquids, and that the flow is frozen, which means that the pressure and temperature drop cause no phase transitions [37, 39].

$$Q = C_d A \sqrt{\frac{2}{\rho}(P_{wh} - P_{dc})} \tag{2.3.1}$$

$$\dot{m} = C_d A \left(2\rho_{dc}^2 p_{wh} \left(\frac{\kappa}{\kappa - 1} \eta_G \left(\frac{1}{\rho_{G,wh}} - \frac{p_r}{\rho_{G,dc}} \right) + \left(\frac{\eta_O}{\rho_O} + \frac{\eta_W}{\rho_W} \right) (1 - p_r) \right) \right)^{1/2} \tag{2.3.2}$$

To calculate the homogeneous density (ρ in (2.3.1) and ρ_{dc} in (2.3.2)), the equation seen in (2.3.4) is used.

$$\frac{1}{\rho_{G,dc}} = \frac{1}{\rho_{G,wh}} \left(\frac{p_{wh}}{p_{dc}} \right)^{\frac{1}{\kappa}} \tag{2.3.3}$$

The dimensionless discharge coefficient C_D contains the summation of the effects of several theoretical (but not negligible) factors such as a change in diameter ratio, Reynolds number, roughness of the pipe, and sharpness of the edges inside the choke valve [37].

$$\frac{1}{\rho} = \frac{\eta_O}{\rho_O} + \frac{\eta_G}{\rho_G} + \frac{\eta_W}{\rho_W}, \quad \eta_W = 1 - \eta_G - \eta_O \tag{2.3.4}$$

To calculate the density of the gas, it is assumed that the real gas law seen in (2.3.5) is valid, where M_G is the molar mass of the gas, Z is the gas compressibility factor, and R is the universal gas constant. Furthermore, it is assumed that the gas expansion through the choke is adiabatic, as seen in (2.3.3) where κ is the adiabatic gas expansion coefficient. Both models also assume a linear relationship between choke opening and flow rate through the choke.

$$\rho_G = \frac{pM_G}{ZRT} \tag{2.3.5}$$

Due to the simplicity and ease of implementation, (2.3.1) will be used and referred to as the mechanistic model for the rest of the thesis.

2.3.2 Data-driven flow modeling

In data-driven flow modeling, historical data is vital to establish a generalizable model that does not only apply to the data that have already been gathered. If the goal only was to find a model that fit the pre-gathered data, one could have constructed a look-up table with the pre-gathered data points and hoped that the future data points would not stretch out of the domain of the data in the look-up table. There are several methods for developing a data-driven model, where one of the most versatile is the neural network as described in Section 2.2 [27].

The data-driven approaches are often referred to as black-box models as a reference to not being able to know what happens inside the box. These models typically do not know anything about the underlying physics of the process from which the data originate, and the parameters of a data-driven model do not necessarily have any physical meaning [27]. Data-driven models are also better suited for adapting to complex correlations in the

model that one does not necessarily know about beforehand, and the development and maintenance costs often show to be lower than for mechanistic models [40].

In this thesis, a neural network as described in Section 2.2 will be used as the data-driven flow model.

2.3.3 Hybrid flow modeling

A hybrid model combines a mechanistic model and a data-driven model, also called a gray-box model. The way the two models are combined may vary, but the combination of elements from data-driven and mechanistic models is typical for all forms of hybrid models. The goal of a hybrid model is to combine the best of both worlds and perform better than a standard data-driven model or a simple mechanistic model. A physics-informed neural network (PINN) may be an example of a hybrid model [11].

Hybrid modeling may be thought of as a grayscale, where the leftmost side of the grayscale is the white side where the mechanistic models lie, while the right side of the grayscale is the black side where the data-driven model lies [14]. An illustrative figure of the hybrid model domain may be seen in Figure 2.3.2.

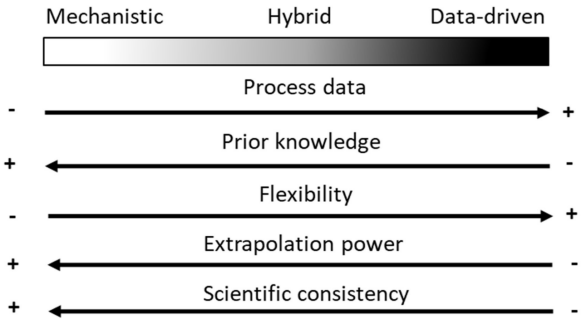


Figure 2.3.2: An illustration of how the hybrid model grayscale may be illustrated where the mechanistic model and the data-driven model are on the edges, and everything in between may be thought of as a hybrid model between the two. The five characteristics underneath illustrate the properties of each of the models. The illustration is reprinted from [14] with permission.

The characteristics in Figure 2.3.2 are descriptive of the advantages and disadvantages of the different models. The process data characteristic indicates that a mechanistic model requires very little prior process data to be implemented. On the contrary, a data-driven model needs a lot of historical process data to learn how to behave to adapt to the characteristics of the true process. The prior knowledge characteristic describes how much prior knowledge is needed to be able to develop a model in the first place. In a mechanistic model, it is crucial to know how the inner mechanics of the process work, how everything is bound together and is typically represented with a model equation (as in Section 2.3.1) where it is easy to interpret how the output changes if one of the model parameters is increased or decreased. In a data-driven model, on the other hand, very little prior knowledge is needed to model the process, as the data-driven model often learns

the correlations within the process without being interpretable by a human. However, no prior knowledge may cause the model to perform very poorly and do not adhere to physical expectations. A flow model as an example clarifies that the model probably would perform quite poorly if the features given to the neural network only were the wellhead temperature and the time of day. However, intuition tells us that the same features used in a mechanistic model are a good starting point and probably result in a better-performing data-driven model.

The flexibility characteristics describe how well the model can adapt to complex correlations and patterns. The extrapolation power characteristics describe how well the model is able to accurately adapt to inputs that were not previously observed while maintaining a reasonable output of the model. A mechanistic model only describes the phenomena and correlations present in the model equations. In contrast, the inner workings of a data-driven model are practically impossible to interpret. The data-driven model can even adapt to very complex correlations and patterns that one may not even know exist in the data, which may revolutionize how the process is interpreted by humans [41]. A data-driven model that has been trained to operate within certain boundaries of variable values is prone to behave strangely and non-consistent outside of those boundaries. On the contrary, a mechanistic model will follow the correlations within the equation regardless of previously observed data.

The scientific consistency characteristic describes how well the model follows the physical principles of the process it represents. When a mechanistic model is derived from mass, energy, and momentum balances, physical principles are naturally accounted for, which is not the case for data-driven models that may work poorly for specific domains in the dataset and are not scientifically consistent.

One way to construct a hybrid model is to modify the cost function in the neural network such that the cost increases if the output of the network for a certain data point differs from the output of the mechanistic model for the same data point [11]. The data-driven part of the model is the neural network which works as previously described. However, the mechanistic part is added to penalize the network if it deviates from the mechanistic model. A modified cost function may be seen in (2.3.6), with $H(\boldsymbol{\theta})$ as described in (2.3.7). The regularization factor λ_H is a hyperparameter that may be tuned when configuring the neural network.

$$J(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + R(\boldsymbol{\theta}) + H(\boldsymbol{\theta}) \tag{2.3.6}$$

$$H(\boldsymbol{\theta}) = \lambda_H \left(y_i - y_{i,\text{mech}}\right)^2, \quad y_{i,\text{mech}} = Q_i = C_d A_i \sqrt{\frac{2}{\rho_i}(P_{i,wh} - P_{i,dc})} \tag{2.3.7}$$

Another way of constructing a hybrid model is to pretrain the model with generated fake data that has been constructed and retrieved from a distribution that has been designed with regard to the real training data [42]. The distribution of the data that are being generated may vary, and it may be essential to utilize the potential for gained performance by pretraining. The distribution may be uniform such that the generated fake data is dense and covers the whole domain that the real data cover, or by other experimental

methods. Another potential distribution to use is a custom distribution that better reflects the distribution in the real data. However, such a distribution may be quite sparse and do not pave the way for good generalization properties of the final model. The generated fake data should be sent through the mechanistic model such that the label of the constructed data point is in line with the mechanistic model.

When fake data has been generated, the neural network may be pretrained on the fake data such that the parameters $\boldsymbol{\theta}$ in the neural network adapt to and learn the correlations in the mechanistic model. After the model has been trained with the generated fake data and the corresponding label, the network may be trained with real training data.

Chapter 3

Methodology and setup

3.1 Software

The software used to develop the code used in the experiments in this master thesis can be seen in Table 3.1.

Table 3.1: The Python and python package versions used in the code used to do the experiments in this master thesis.

Software	Version
Python	3.8.0
PyTorch	1.10.2+cpu
Matplotlib	3.5.1
Pandas	1.4.1
Numpy	1.22.1
Scipy	1.8.0

3.2 Dataset and preprocessing

The dataset that was used in this report was provided by Solution Seeker AS [15]. The data was extracted from five different oil wells from an anonymous oilfield on the NCS with Solution Seekers steady-state extraction technology [43]. Each data point contained two timestamps in which the data point was obtained as a stable production period within that time interval, the choke opening (CHK), the wellhead pressure (PWH), the downstream choke pressure (PDC), the wellhead temperature (TWH), a precalculated compressibility factor (Z), the oil and gas mass fractions (FOIL and FGAS), and the flow rate variables for the three phases (QOIL, QGAS, and QWAT), as well as the total flow rate (QTOT) in a scaled version of cubic meters per hour (see Section 3.2.1). QTOT is the sum of the three phase flow rates. All variables at each data point except the four flow variables were scaled to an interval between zero and one. Each data point also contained information on which of the five wells the data point belonged to and whether the data were obtained

by a MPFM or by a well test. The data were gathered from the five different wells as steady-state intervals, but may still contain time-varying correlations such as slow pressure decrease in the reservoir, or wear and tear on the choke and other process equipment that e.g. may affect the resistance through the choke. Significant disturbances in the process, start-up periods, or shutdown periods are excluded from the dataset.

The dataset for the five wells is split into a training set, a validation set, and a test set. The number of data points in each dataset corresponding to each well, as well as the total number of data points in each well and the total number of data points in each dataset, can be seen in Table 3.2. The split between the validation set and training set was 20% to 80% over all the data, which is in line with the rule of thumb of a split of 20/80, but the split varied between the five different wells. The size of the test set ranged from 3.9% to 11.2% of the total data amount for each well, but for all wells in total, the test dataset was equal to approximately 5%.

Table 3.2: A table showing the number of data points in the training dataset, validation dataset, and test dataset for each well, as well as their totals.

Well	Training data	Validation data	Test data	Total
W1	2773	694	148	3615
W2	1919	480	154	2553
W3	992	249	156	1397
W4	2779	695	141	3615
W5	2635	659	136	3430
Total	11098	2777	735	

The data was sampled from the 20th of December 2015 to the 1st of March 2021. The validation set was separated out of the total dataset in random time intervals, ranging from the very start to the very end of each period in which the data from each well was sampled. The test set was approximately equal in size for all wells, and the extraction of test data was done by defining a limit where all data gathered after the limit was defined as test data.

3.2.1 Scaling and filtering

The data was scaled before it was provided by Solution Seeker AS. The factors with which the different variables were scaled were provided so that realistic calculations with the data were possible. Therefore, the data were scaled back to unscaled values before any calculations were conducted, but scaled back again to exist in the same domain as the provided data. All plots of data in the thesis will hence be scaled according to the provided scaling such that the true values of the data remain anonymous.

During the generation of fake data, the variables were drawn to exist in the same domain as the real variables. The label in each data point was scaled the same way as the real data such that the generated fake data would be comparable to the real data.

All real data were filtered with the filters shown in Table 3.3 before the data was used in the training of the neural network to not corrupt the algorithm with obviously faulty data

points. When fake data was generated, the intervals the fake data was within were also within the filter limits.

Table 3.3: The data points that fulfilled the conditions shown in the table below was removed from the dataset. Of all the data, there was removed 18 data points with the 1st filter, one data point with the 5th filter, two data points with the 6th filter, and one data points with the 7th filter.

Filter	Condition	Removed data points with ...
1	PDC < 0.03	very low pressures after the choke valve
2	PWH < 0.03	very low pressure before the choke valve
3	PWH $<$ PDC	lower pressure before than after the choke valve
4	TWH < 0	negative temperature before the choke valve
5	CHK < 0.05	choke valve opening lower than 5%
6	(CHK < 0.2) & (QTOT > 10)	high total flow rate with a low choke valve opening
7	(CHK > 0.5) & (QTOT < 5)	low total flow rate with a high choke valve opening

The first filter was set such that very low pressure values after the choke were filtered out. Without spoiling the true range of the downstream choke pressure variable, the downstream pressure of the choke is the same pressure as the separator pressure, and scaled downstream choke pressure values below 0.03 were defined as too low as the separator pressure would be higher than this. Values below 0.03 would therefore indicate flow in the negative direction which is not possible. Consequently, the second filter was also set to exclude values below 0.03 for the wellhead pressure variable, since the downstream choke pressure is always lower than the wellhead pressure, which is also ensured by filter number three. The fourth filter was set to ensure that there were no erroneous temperature measurements, as these need to be positive to make sense. The fifth filter was set to exclude very low choke percentage openings, as choke openings lower than 5% would results in very little to know flow due to the resistance through the valve. The sixth filter was included to discard data points with very high total flow rate values, even though the choke percentage opening was relatively low. The seventh filter was included to remove data points that had very large choke openings but relatively low total flow rates as a large choke percentage opening would result in a substantial amount of flow.

3.3 Data-driven flow modeling

The neural network used in this master thesis was a fully connected feed-forward neural network trained with back-propagation. The same neural network design was used for all the experiments conducted. The input features were choke opening, the wellhead pressure, the downstream choke pressure, and the wellhead temperature. The number of hidden layers in the neural network was set to three, with a width of 50. The ReLU activation function and the Adam optimizer were used for all experiments. MAP estimation was used to find the parameters $\boldsymbol{\theta}$ in the model. The parameters in the network were randomly initialized with the He-initialization as described in Section 2.2, and with a Pytorch random seed of "12345". The input features and the output label of the neural network were for all experiments the same. The choke opening, the downstream choke pressure, the wellhead

pressure, and the wellhead temperature as the input, and the total flow rate through the choke as the output.

The hyperparameters used in the training of the neural network were mainly adopted from the work done in the project report [7]. It was not spent much more time finding the optimal hyperparameters since that was not the goal of the thesis. The best set of hyperparameters was used for the standard feature combination (learning rate 0.002 and l2-regularization of 0.003 for well 1, 3, and 5, and learning rate 0.003 and l2-regularization of 0.005 for well 2 and 4). The batch size of 2048 used in the project report was lowered to 512 as this turned out the work as well as 2048, with a slight improvement for some wells.

During training, early stopping was used with a patience of 500. Even though the data was extracted as steady-state intervals, the order of the data was randomized before training such that any remaining time-varying correlations hopefully faded out. Before the final experiments that produced the final results were performed, the training data was merged with the validation data according to the normal neural network practice as described in Section 2.2.1.

During development of the code for this thesis, minor similarities between some of the wells were found. Figure 3.3.1 shows that well 1, 4, and 5 (red, blue, and purple) have data points that lie in approximately the same domain, and the same can be said for well 2 and 3 (green and yellow). Therefore, an experiment was conducted where training data for those wells were merged and tested on the test dataset for the five individual wells.

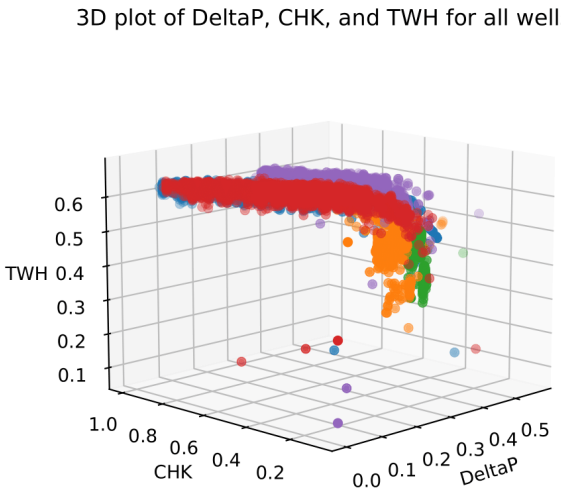


Figure 3.3.1: Scatter plot with data points from all wells, plotted against the choke opening, the wellhead temperature, and the differential pressure over the choke (DeltaP).

Although it is noise originating from the measurement equipment on the provided data, an experiment where normally distributed noise with a mean of zero and standard deviation of 0.5 and 1 respectively, was added to the label of the training data. The performance

of this experiment can be seen in Table 4.4. A plot of the total flow rate in well 3 (used as an example) can be seen in Figure 3.3.2, and a graph with noise with zero mean and 0.5 standard deviations can be seen in Figure 3.3.3. The plots of the other wells (without noise) can be seen in Appendix A.4

To evaluate the performance of the data-driven flow model on the test data, three performance measures have been developed to quantify the performance. These performance measures can be seen in (3.3.1).

$$\textbf{MAPE} = \frac{1}{n}\sum_{i=0}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \cdot 100, \quad \textbf{MAE} = \frac{1}{n}\sum_{i=0}^{n}|(y_i - \hat{y}_i)|, \quad \textbf{MSE} = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2. \tag{3.3.1}$$

Mean absolute percentage error (MAPE) was used as the primary performance measure, while Mean absolute error (MAE) and MSE were used as a supplement. When MAPE was calculated, the y_i values below 1 were discarded due to the consequence of making MAPE misleadingly large. More details and considerations on the use of MAPE as the primary performance measure are provided in De Myttenaere *et al.* [44].

Figure 3.3.2: The blue graph shows the total flow rate label from the dataset of well 3 plotted in time. The red graph shows the calculated total flow rate using (2.3.1) where the C_D parameter has been set to 0.2835 such that the calculated total flow rate better fits the data from the dataset. The total choke inner cross-section area was assumed to be $0.003m^3$. The green graph shows the difference between the real and the estimated total flow rates.

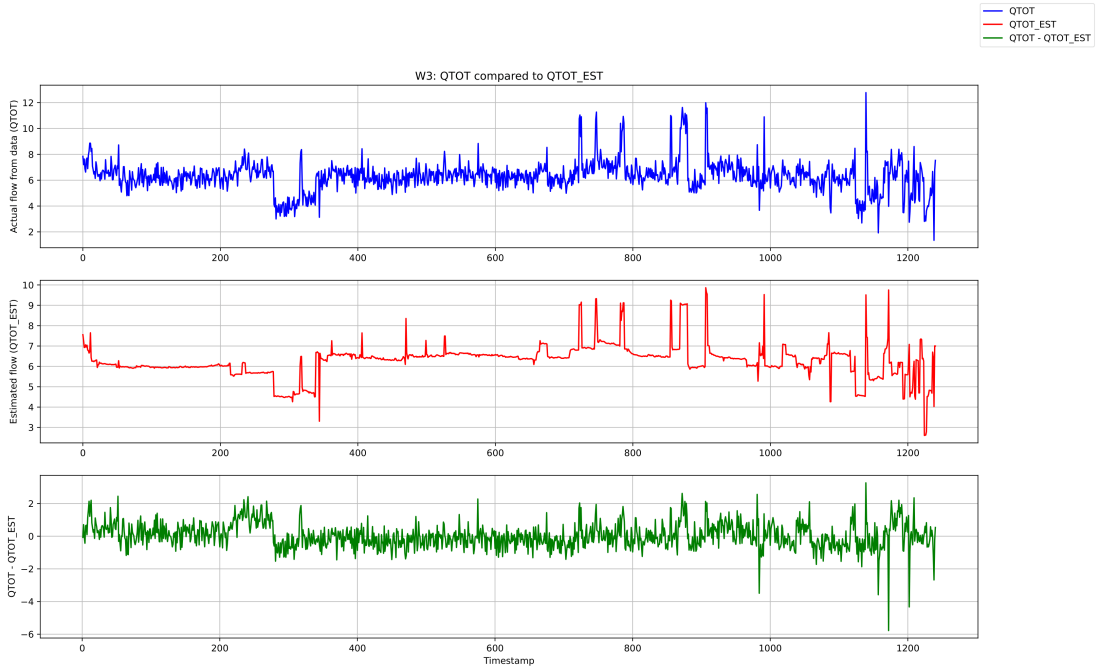


Figure 3.3.3: The blue graph shows the total flow rate label from the dataset from well 3 plotted in time where noise with zero mean and 0.5 standard deviations has been added. The red graph shows the calculated total flow rate using (2.3.1) where the C_D parameter has been set to 0.2835 such that the calculated total flow rate better fits the data from the dataset. The total choke inner cross-section area was assumed to be $0.003m^3$. The green graph shows the difference between the real and the estimated total flow rates.

3.4 Hybrid flow modeling

Two approaches to hybrid flow modeling were tested in this thesis. The first hybrid flow modeling approach was to use a Physics-informed loss-function (PILF) within the neural network, presented in Section 3.4.1. The second approach was to pretrain the neural network on generated fake data, presented in Section 3.4.2. In addition to testing these two, a combination of them was also tested, presented in Section 3.5.

3.4.1 Physics-informed loss function

The first approach to hybrid flow modeling was to add a term to the loss function of the neural networks that penalizes deviation from the total flow rate calculated by the mechanistic flow model as described in (2.3.6) and (2.3.7). This was done by precalculating the theoretical total flow rate with the mechanistic flow model using the provided data and penalizing deviations from the precalculated total flow rate multiplied by a regularization factor. During some experimentation with the data (before the final testing), the regularization factor λ_H was found to work best when it was set to 0.001.

When constructing the total flow rate variable with the mechanistic flow model, the flow

model presented in (2.3.1) was used, where ρ was calculated using (2.3.4), and ρ_G was calculated using the real gas law seen in (2.3.5) and the inner choke cross-section area was assumed to be $0.003m^3$. The total flow rate calculated with (2.3.1) was multiplied by 3600 to convert the flow per second to flow per hour. After the total flow rate variable was constructed, it became apparent that the flow model did not fit very well with the provided data. The valve discharge coefficient C_D was therefore adjusted so that the generated data better fit the provided data. The C_D variables used for each well may be seen in Table 3.4.

Table 3.4: The valve discharge coefficient values for the five wells when precalculating the total flow rate with the mechanistic flow model.

Well	C_D values
W1	0.6550
W2	0.3550
W3	0.2835
W4	0.6310
W5	0.5725

After the total flow rate was calculated with (2.3.1) for the five wells, the total flow rate was divided into Q_O, Q_G, and Q_W and scaled in the same way as the provided data before summed to a scaled version of the total flow rate.

3.4.2 Pretraining

The second approach of hybrid flow modeling was to pretrain the neural network with generated fake data to update the parameters $\boldsymbol{\theta}$ according to the mechanistic flow model to adapt the model to the correlations in the mechanistic flow model before the network was trained on real data. When the neural network was pretrained, an l_2-regularization of 0.0005 was used, and a learning rate of 0.005. The same early stopping patience of 500 as for training on real data was used during pretraining. When the pretraining was done, the parameters developed in the pretraining were returned and used as initial parameters for the training on real data.

The l_2-regularization of 0.0005 during pretraining was lower than when training on real data, as overfitting to the generated fake data would not be a problem since the network was further to be trained on real data. With that, the l_2-regularization could also be set to 0, but to avoid nodes in the network from converging to 0 (and never being able to change from 0), a small amount of l_2-regularization was kept during pretraining.

3.4.2.1 Construction of fake data

The purpose of the generated fake data (or rather the pretraining) was to hopefully provide a parameter initialization that is closer to the optimal value. To pretrain the network, fake data was generated with the mechanistic flow model. For every well that was pretrained, 2000 fake data points were generated. Each data point included what well the data point

belonged to, the choke opening, downstream choke pressure, wellhead pressure, wellhead temperature, a fraction of oil, a fraction of gas, a fraction of water (derived from fractions of oil and gas), and the total flow rate.

The variable values in each data point (except for the total flow rate and the well) were drawn from two different distributions. One uniform distribution with distribution limits set to a defined limit based on the ranges in which the real data lived in (elaborated upon in Section 3.4.2.2) and a well-specific distribution.

When fake data was generated both with the uniform and well-specific distributions, the data was generated for a specific well (1 through 5), and the corresponding total flow rate was calculated with the mechanistic flow model seen in (2.3.1).

The mechanistic flow model was not perfectly fitted to each well. Therefore, it was expected beforehand that the generated total flow rate would not precisely match the real data. Thus, the generated total flow rates were multiplied by a factor to better fit the real data after the generated data had been scaled to live in the same domain as the real data. The factors with which the total flow rate was downscaled with may be seen in Table 3.5. These factors are not identical to the ones used as discharge coefficients in the precalculated total flow rate used in the PILF (Table 3.4). However, they are approximately in the same range. The discharge coefficient C_D was set to 1 when data was generated from the uniform and well-specific distribution.

Table 3.5: The factors that was used to downscale the total flow rate generated by the mechanistic flow model in the fake data generation.

Well	Factors
W1	0.5923
W2	0.3779
W3	0.3263
W4	0.6131
W5	0.7116

A boxplot of the total flow rate generated by the uniform distribution compared to the total flow rate in the real data may be seen in Figure 3.4.1, while the total flow rate generated by the well-specific distribution compared to the total flow rate in the real data may be seen in Figure 3.4.2. Boxplots of all variables for both the uniform and well-specific distributions may be seen in Appendix A.5.

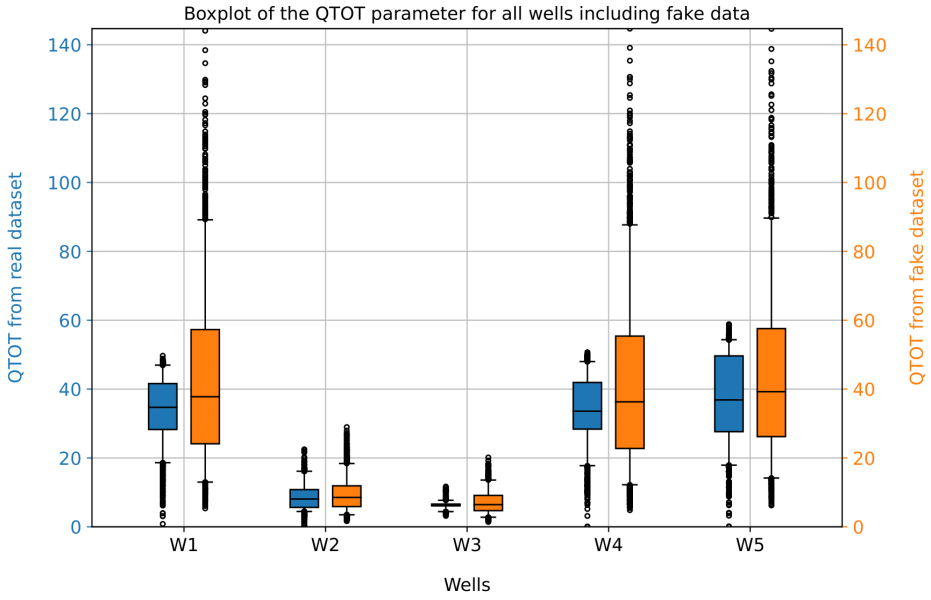


Figure 3.4.1: The boxplot shows a comparison between the real total flow rate variable and the total flow rate variable generated by mechanistic flow model when the variables used to calculate it were drawn from a uniform distribution.

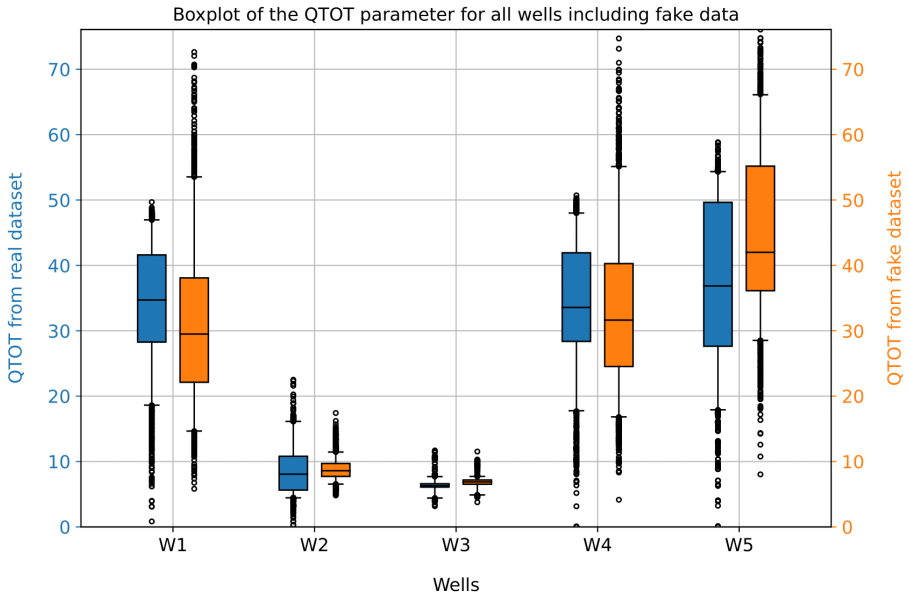


Figure 3.4.2: The boxplot shows a comparison between the real total flow rate variable and the total flow rate variable generated by mechanistic flow model when the variables used to calculate it were drawn from a well-specific distribution.

The factors could have been perfectly calculated to center the total flow rate error around zero. However, the parameters were instead found by trial and error to achieve a more realistic flow model that does not live in the perfect equilibrium between positive and negative error.

3.4.2.2 Distributions of fake data

The two distributions used to obtain the values of the variables for the generated fake data were a uniform distribution and a well-specific distribution. The well-specific distribution was developed by sorting the real data for a well into a histogram with 100 bins and fitting a probability density function to the histogram with the $stats.rv\_histogram()$ function in Scipy. When data was drawn from a well-specific distribution, the distribution of values was more similar to the real data distribution than the variables drawn from a uniform distribution. Scatter plots of data generated with a well-specific distribution for well 2, 3, and 4 can be seen in Figure 3.4.3, Figure 3.4.4 and Figure 3.4.5 respectively. The remaining scatter plots can be seen in Appendix A.6.

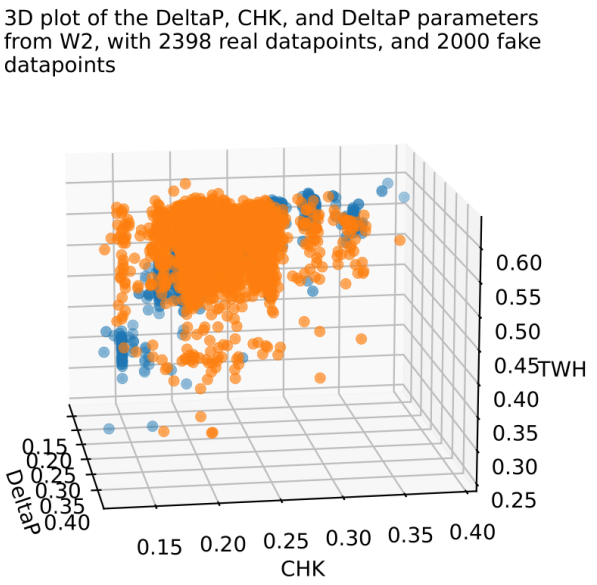


Figure 3.4.3: The real data from well 2 plotted in a scatter plot together with the generated data drawn from a well-specific distribution. The real data appear as blue dots, while the generated data appear as orange dots.

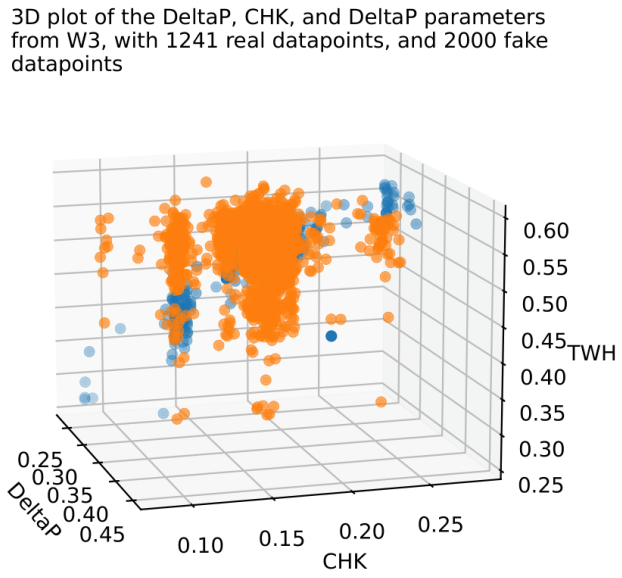


Figure 3.4.4: The real data from well 3 plotted in a scatter plot together with the generated data drawn from a well-specific distribution. The real data appear as blue dots, while the generated data appear as orange dots.

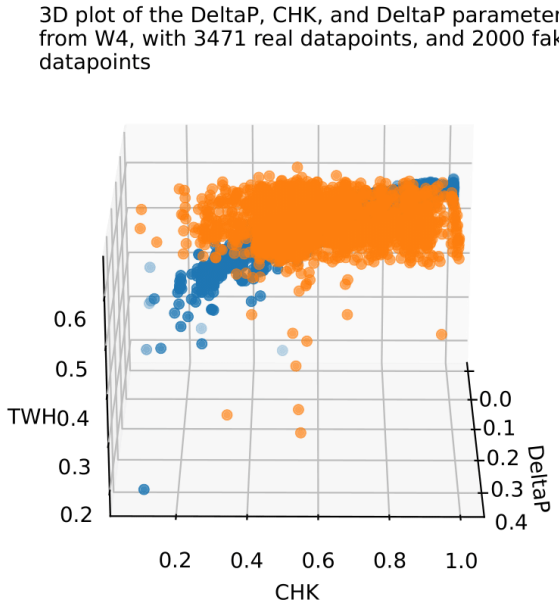


Figure 3.4.5: The real data from well 4 plotted in a scatter plot together with the generated data drawn from a well-specific distribution. The real data appear as blue dots, while the generated data appear as orange dots.

The uniform distribution had a specified upper and lower limit, where all values inside the limits had the same probability of being drawn, hence a uniform distribution. The upper and lower limits were chosen by inspecting a histogram of each variable for each well and defining the limits as an upper and lower percentile of the real data, such that most of the data were kept within the distribution, but outliers fell outside of the distribution.

Figure 3.4.6 is an excellent example of the advantage of defining such limits and not including all data points when setting the limits on the uniform distribution. The plot has a logarithmic scale on the y-axis to increase the visibility of the bins that represent very few data points. If the limits had been set to include all the data points, it is easy to see that many values between 0.55 and 0.65 would have been drawn as generated values, but those values would not be present in the real data. On the other hand, Figure 3.4.7 is a good example of a drawback of this approach, as it is challenging to determine where to put the limits.

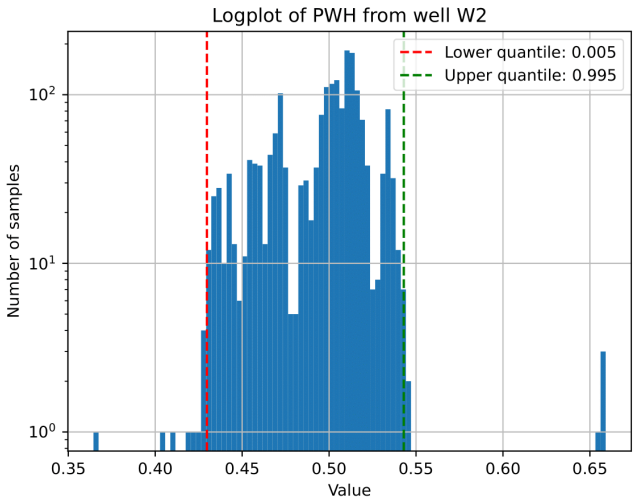


Figure 3.4.6: All wellhead pressure data points from well 2 plotted is a histogram with 100 bins. The plot has a logarithmic y-axis to increase the visibility of the bins that contained very few samples.

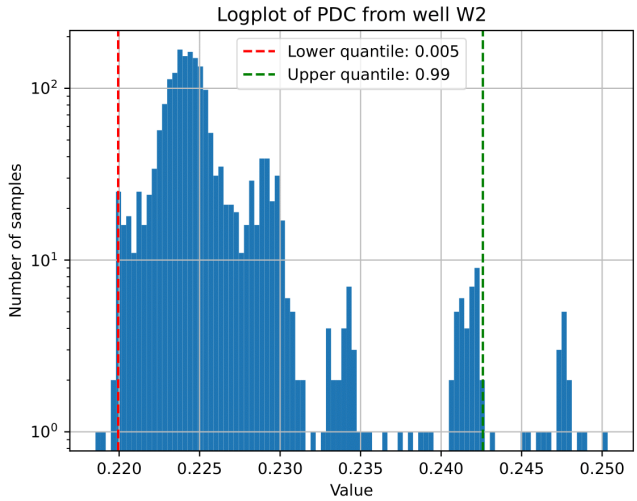


Figure 3.4.7: All downstream choke pressure data points from well 2 plotted is a histogram with 100 bins. The plot has a logarithmic y-axis to increase the visibility of the bins that contained very few samples.

For Figure 3.4.7, the large majority of the data points live between 0.220 and 0.230, but a significant number of data points also live between 0.230 and 0.250. Putting the limit at the green striped line as seen in Figure 3.4.7 will statistically result in equally many fake data points to living between 0.235 and 0.240 (where for this example it only is eight real data points). However, the limit for this specific example was still set to a 0.99 percentile to include the island of data points to the left of the green striped line. The case was similar for several other variables for all wells, where a similar assessment had to be made. The histograms for all variables for all wells can be seen in Appendix A.3.

When the variables were randomly drawn, the choke variable, the downstream choke pressure variable, the wellhead temperature variable, and the fraction of oil variable (η_O) were all drawn between the lower and upper percentiles. The wellhead pressure variable was drawn between the drawn downstream choke pressure value and the upper percentile, plus 0.05 to ensure that the upstream pressure was significantly above the downstream pressure. The fraction of gas variable η_G was drawn between 0 and $1 - \eta_O$, while the fraction of water variable η_W was calculated as seen in (2.3.4) ($\eta_W = 1 - \eta_O - \eta_G$). The area parameter A used in (2.3.1) was set to 0.003 (m^2) multiplied by the drawn choke variable. A table showing the percentiles used can be seen in Table 3.6.

A scatter plot of all the fake data from all wells drawn from a uniform distribution, plotted against the wellhead temperature variable, the choke opening variable, and the delta pressure ($p_{wh} - p_{dc}$) can be seen in Figure 3.4.8. All the fake data points appear as 3D boxes in the plot, since all variables were drawn from a uniform distribution. A scatter plot of the real data on top of the generated data for well 2 may be seen in Figure 3.4.9 as an example.

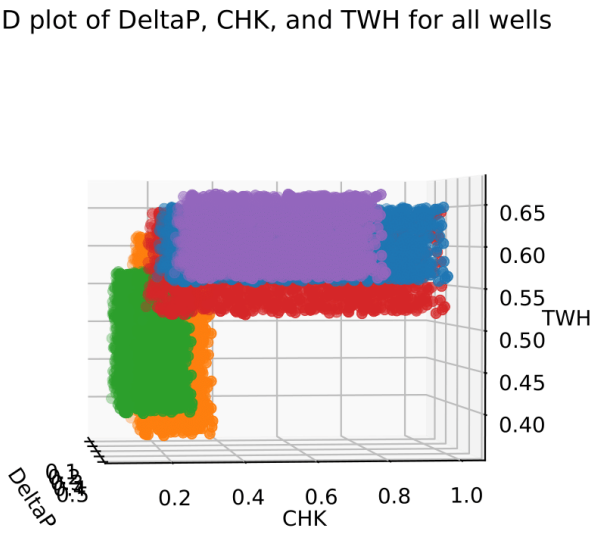


Figure 3.4.8: All generated data with variables drawn from uniform distribution gathered in the same plot.

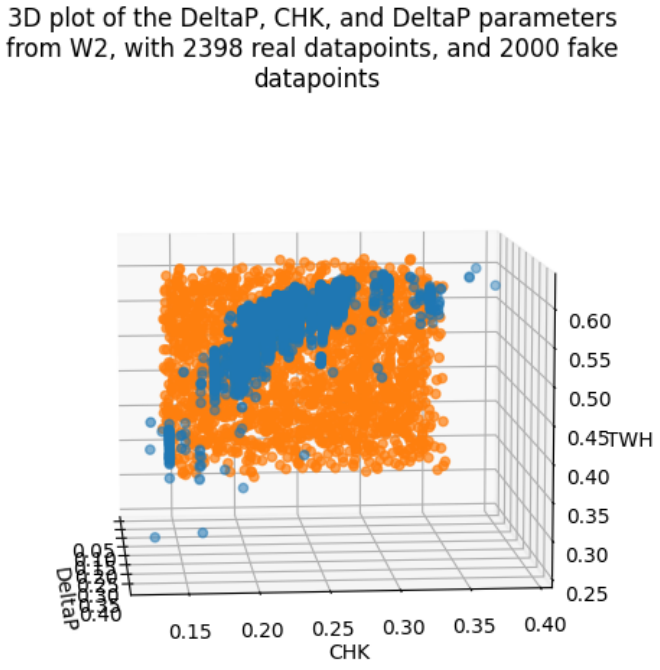


Figure 3.4.9: The real data from well 2 plotted in a scatter plot together with the generated data drawn from a uniform distribution. The real data appear as blue dots, while the generated data appear as orange dots.

Comparing Figure 3.4.3 and Figure 3.4.9 makes it easy to see that the extracted values from the two distributions are completely different.

Table 3.6: The percentiles that was used when drawing data from uniform distribution. The lower percentile for the wellhead pressure variable was not applicable as the drawn downstream choke pressure variable was used as the lower limit when drawing the wellhead pressure variable.

Variable	Percentile	W1	W2	W3	W4	W5
CHK	Lower	0.004	0.005	0.001	0.002	0.005
	Upper	1.000	0.995	1.000	1.000	1.000
PDC	Lower	0.003	0.005	0.004	0.005	0.003
	Upper	0.999	0.990	0.995	0.997	1.000
PWH	Lower	N/A	N/A	N/A	N/A	N/A
	Upper	0.998	0.995	0.996	0.997	0.997
TWH	Lower	0.010	0.005	0.020	0.005	0.005
	Upper	0.995	0.995	0.990	0.995	0.995
FOIL	Lower	0.005	0.002	0.006	0.004	0.006
	Upper	0.995	0.995	0.995	0.998	1.000

Figure 3.4.10 and Figure 3.4.11 visualize how different the distributions in the data points from the two distributions appear compared to the real data shown in blue in both plots.

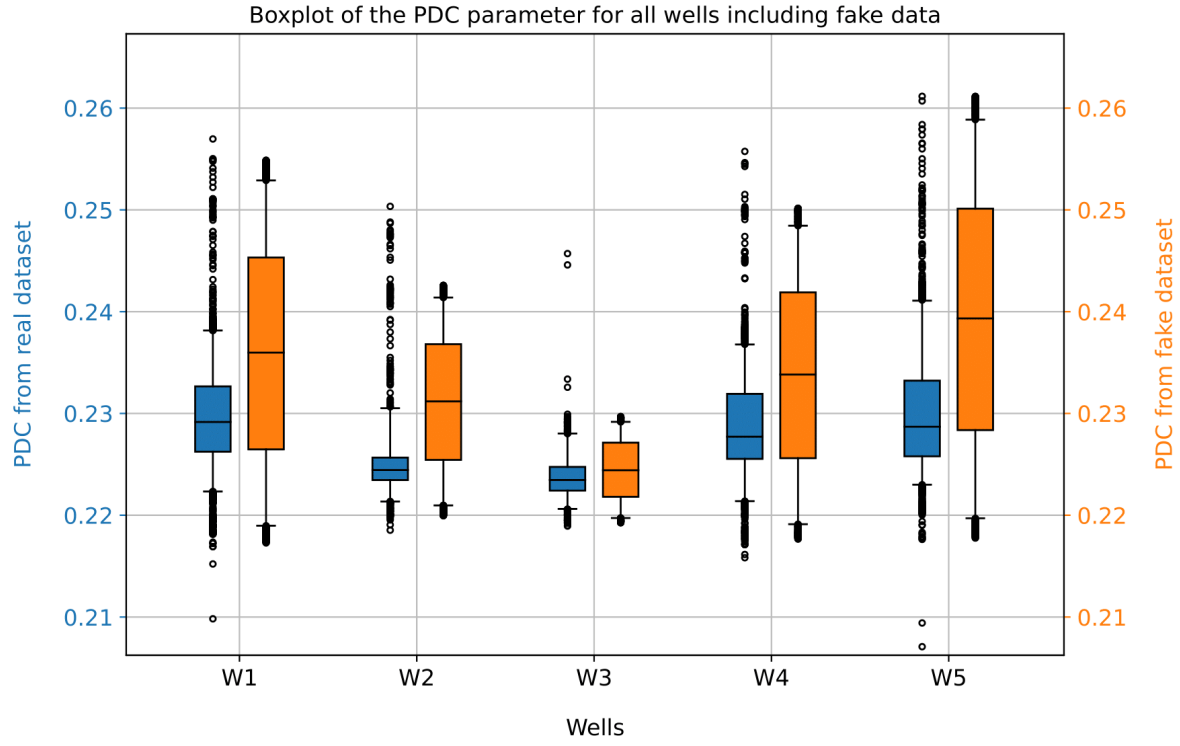


Figure 3.4.10: A boxplot of the downstream choke pressure variable drawn from uniform distribution compared with the real downstream choke pressure variable.

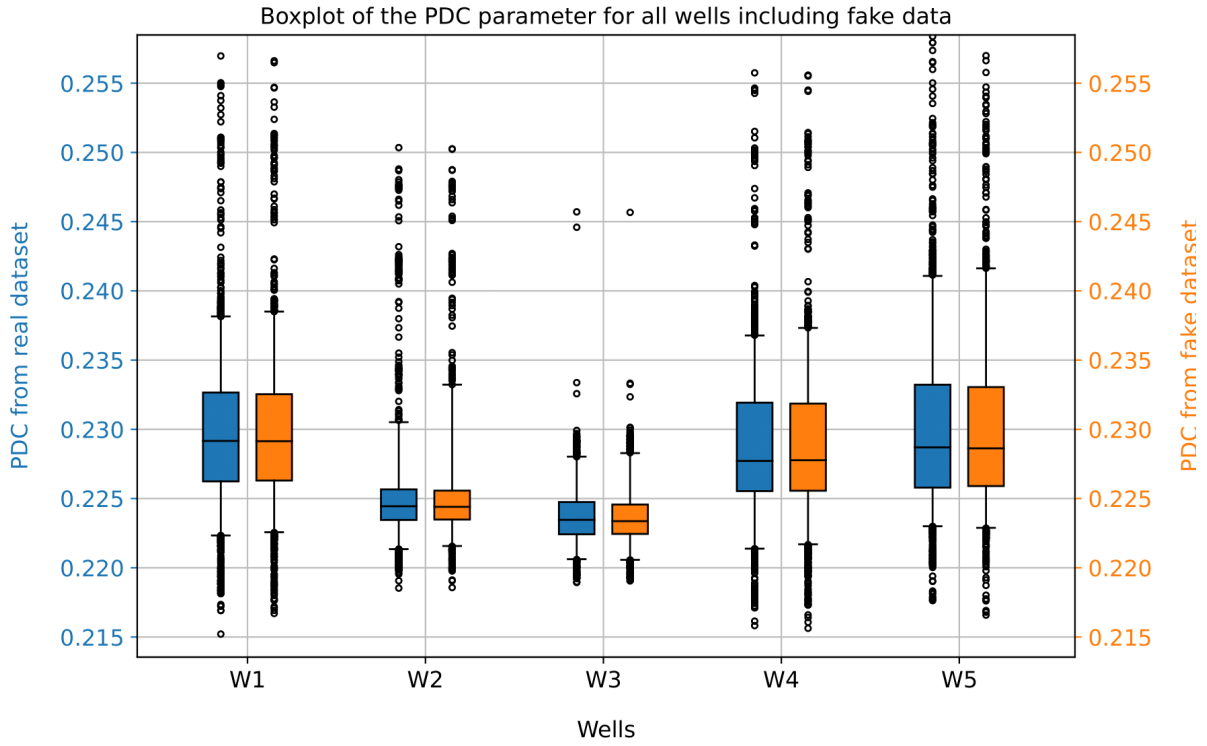


Figure 3.4.11: A boxplot of the downstream choke pressure variable drawn from well-specific distribution compared with the real downstream choke pressure variable.

3.5 Combination of pretraining and physics-informed loss function

The two approaches of hybrid flow modeling that are described in Section 3.4.1 and Section 3.4.2 are in theory independent of each other, and a combined version of the two was, for that reason, also tested to evaluate the performance of such a combination of hybrid flow modeling. No further development or adjustments to the algorithms were made. Therefore, two hybrid flow models consisting of a PILF with pretrained parameters $\boldsymbol{\theta}$ were developed. One was pretrained with generated fake data drawn from a uniform distribution, while the other were pretrained with generated fake data drawn from a well-specific distribution. The same hyperparameters as for the experiments in Section 3.4.1 and Section 3.4.2 were used for both experiments. The results of these experiments can be seen in section 4.7.

Chapter 4

Results

In this chapter, all the results produced in this thesis will be presented. The method used to produce them is described in the previous chapter (Chapter 3, Methodology). The results may be briefly discussed in this chapter, but the main part of the discussion can be found in the next chapter (Chapter 5, Discussion).

4.1 Standard data-driven flow model

The results of the experiment described in Section 3.3 where a standard data-driven flow model was tested on the test dataset and trained with a combination of the training dataset and the validation dataset can be seen in Table 4.1. The best results for each well produced in the project report ([7]) may be seen in Table 4.2. Feature engineering was not part of this thesis, but Table 4.1 shows that the results are approximately in the same range as for the standard feature combination seen in Table 4.2. The tables indicate that there may be even more potential for improved performance using feature engineering in combination with a hybrid flow model. The results presented in Table 4.1 will be used as reference in the other results presented in this chapter.

Table 4.1: The performance of the standard data-driven flow model with the standard feature combination (the choke opening, the wellhead temperature, the wellhead pressure, and the downstream choke pressure). The LR column shows the learning rate α, and the l_2-reg column shows the regularization factor λ. The Epochs column shows for how many epochs each model trained. The MAE, MSE, and MAPE columns shows the error on the test dataset as described in Section 3.3.

Well	LR	l_2-reg	Epochs	MAE	MSE	MAPE
W1	0.002	0.003	2021	0.9853	1.4584	2.5285%
W2	0.003	0.005	529	2.8849	12.9182	34.7340%
W3	0.002	0.003	7119	0.1886	0.0116	2.5634%
W4	0.003	0.005	1324	2.3778	6.3680	7.3569%
W5	0.002	0.003	2819	2.2337	7.8562	8.8449%

Table 4.2: The results from the project report [7] with the best performance of the data-driven flow model with the standard feature combination (the choke opening, the wellhead temperature, the wellhead pressure, and the downstream choke pressure) and the best feature engineering combination. FOIL is the fraction of oil, FGAS is the fraction of gas, Z is the precalculated gas compressibility factor, and DP2 is the squared delta pressure over the choke valve.

Well	Added features	l_2-reg	LR	Epochs	MAE	MSE	MAPE
W1	None	0.003	0.002	7500	1.046	1.606	2.725%
	FOIL	0.005	0.002	10000	0.714	0.849	1.883%
W2	None	0.005	0.003	12500	3.302	16.492	39.606%
	FOIL, FGAS, Z, DP2	0.003	0.003	7500	0.760	1.353	9.763%
W3	None	0.003	0.002	12500	0.145	0.035	2.183%
	Z, DP2	0.003	0.003	12500	0.144	0.035	2.166%
W4	None	0.005	0.003	12500	2.211	5.515	6.973%
	FOIL, FGAS, Z, DP2	0.003	0.003	12500	1.276	2.725	4.213%
W5	None	0.005	0.003	12500	3.009	14.412	11.440%
	FOIL, Z, DP2	0.005	0.003	5000	0.839	1.661	3.999%

4.2 Data-driven flow model with merged data

The results of the experiments described in Section 3.3 where training data from several wells were merged can be seen in Table 4.3. The first row for each well displays the same results as in Table 4.1 for reference (the standard data-driven flow model). The second row for each well displays the performance on the test data when the model was trained on a merged dataset with training data from all the wells and the respective wells validation data. The third row for each well displays the performance of the model on the test set when it was trained on merged training data from well 2 and 3 and validation data from the respective well. The fourth row for each well displays the performance of the model on the test set when the model was trained on training from well 1, 4, and 5 and validation data from the respective well. Only training data between the wells were merged, not validation data and test data.

For well 1 and especially well 3, the results show that these wells performed the best when data from the other wells not was used in the training. The results for well 2, 4, and 5 appear to be a little different. Well 2 performed the best when trained on data from well 1, 4, and 5, while well 4 performed best when trained on data from well 2 and 3. Well 5, on the other hand, performed best when trained on data from all the wells, but only marginally compared to when it only was trained on data from its own well. Also, notice that the best performance of well 5 required less than 25% of the training epochs compared to when it only was trained on its own data.

Both well 1, 4, and 5 only performed slightly worse when data from the other two wells in that set was added to the training data of the models. Well 2 also performed slightly worse when data from well 3 was added to the training data, while well 3 performed significantly worse when data from well 2 was introduced to the training data.

Table 4.3: The table shows the performance of the data-driven flow model when it was trained on merged data. The Training data column shows from what well(s) the training data consisted of. The best-performing configuration for each well is highlighted in bold.

Well	LR	l_2-Reg	Training data	Epochs	MAE	MSE	MAPE
W1	0.002	0.003	W1	**2021**	**0.9853**	**1.4584**	**2.5285**
			All	531	1.4010	2.7100	3.6383%
			W2 and W3	677	1.0985	2.3145	2.9778%
			W1, W4, and W5	1051	1.1521	1.6050	3.0658%
W2	0.003	0.005	W2	529	2.8849	12.9182	34.7340%
			All	518	1.6007	4.1852	19.5251%
			W2 and W3	513	3.0333	13.9884	36.4403%
			W1, W4, and W5	**790**	**1.2576**	**2.5700**	**16.1326%**
W3	0.002	0.003	W3	**7119**	**0.1886**	**0.0116**	**2.5634%**
			All	522	0.7292	1.2234	8.4338%
			W2 and W3	537	1.1082	2.2812	13.4199%
			W1, W4, and W5	630	1.7710	3.6280	28.5430%
W4	0.003	0.005	W4	1324	2.3778	6.3680	7.3569%
			All	646	2.9011	9.4846	9.2137%
			W2 and W3	**2098**	**2.2439**	**6.6101**	**7.0938%**
			W1, W4, and W5	783	2.4851	6.8971	7.6233%
W5	0.002	0.003	W5	2819	2.2337	7.8562	8.8449%
			All	**556**	**1.9490**	**5.9788**	**8.6083%**
			W2 and W3	565	6.8532	72.4180	21.5735%
			W1, W4, and W5	553	2.4028	8.0981	9.4064%

4.3 Data-driven flow model with noise

In this section, the same experiments as in Section 4.1 is presented, but with normally distributed noise added to the total flow rate label in the training dataset. The first row for each well displays the same results as in Table 4.1 for reference (the standard data-driven flow model). The second row displays the performance of the model on the test set when normally distributed noise with 0 mean and 0.5 standard deviations (std) was added to the label in the training and validation data. The third row displays the performance of the model on the test set when normally distributed noise with 0 mean and 1.0 standard deviation was added to the label in the training and validation data.

Well 3 and 5 were the only two wells that performed the best when no noise was added to the label. Well 1 and 2 performed insignificantly better when noise was added to the label. In contrast, well 4 performed significantly better, even though the best performance trained for a lower number of epochs than the two other models for that well.

Table 4.4: The table shows the performance of the data-driven flow model with noise added to the total flow label. The Noise mean column shows the mean of the added noise on the total flow rate, while the Noise std column shows the standard deviation of the added noise. The best-performing configuration for each well is highlighted in bold.

Well	LR	l_2-reg	Noise mean	Noise std	Epochs	MAE	MSE	MAPE
			N/A	N/A	2021	0.9853	1.4584	2.5285%
W1	0.002	0.003	0	0.5	1698	1.1312	1.7385	2.9316%
			0	1.0	1648	**0.8958**	**1.1691**	**2.4210%**
			N/A	N/A	529	2.8849	12.9182	34.7340%
W2	0.003	0.005	0	0.5	522	**2.8542**	**12.5915**	**34.3556%**
			0	1.0	509	3.1945	14.7956	38.8108%
			N/A	N/A	7119	**0.1886**	**0.0116**	**2.5634%**
W3	0.002	0.003	0	0.5	4417	0.2743	0.1064	3.9576%
			0	1.0	7608	0.1871	0.0555	2.7822%
			N/A	N/A	1324	2.3778	6.3680	7.3569%
W4	0.003	0.005	0	0.5	880	**1.5423**	**2.9955**	**4.7472%**
			0	1.0	1352	2.3799	6.3373	7.2006%
			N/A	N/A	2819	**2.2337**	**7.8562**	**8.8449%**
W5	0.002	0.003	0	0.5	1780	2.7250	10.2136	10.6045%
			0	1.0	1202	3.1154	12.3768	12.5406%

4.4 Performance of the mechanistic flow model

The results presented in this section are not of substantial importance as mechanistic flow modeling was not a part of the hypothesis. However, the performance of the mechanistic flow model is included to give the reader a better intuition for evaluating the relationship between the data provided and the estimated total flow rate using the mechanistic flow rate model (2.3.1).

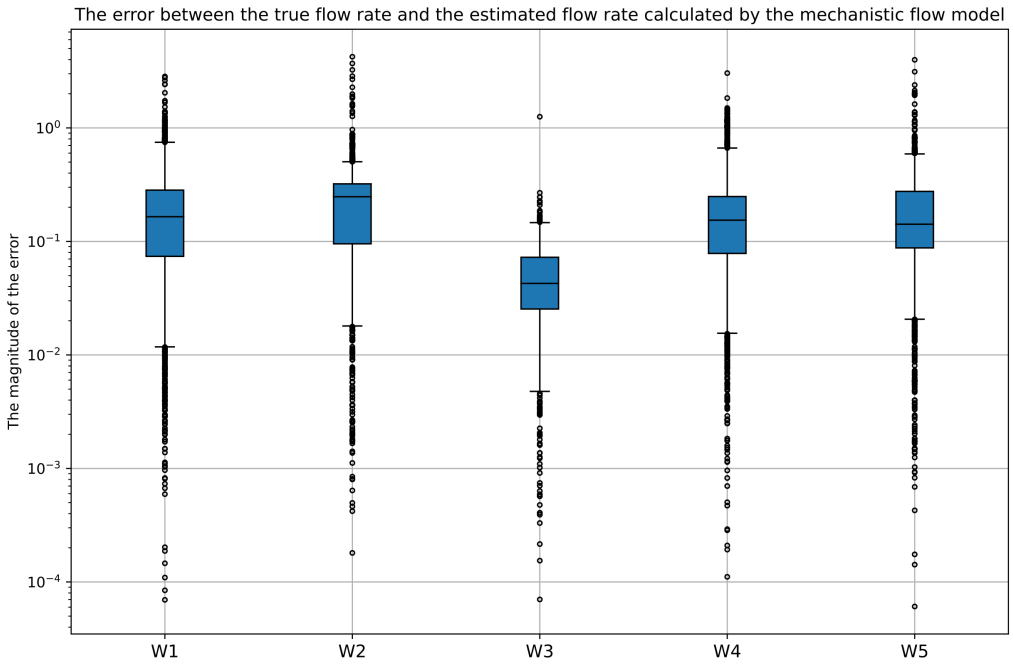


Figure 4.4.1: A boxplot of the absolute percentage error in each well between the mechanistic flow model and the provided data, plotted with a logarithmic scale. The 10^{0} line marks the line where a data point missed the true value by a magnitude of 1, or 100%. The 10^{-1} line marks the line where a data point missed the true magnitude of 1/10, or 10%. The true total flow rate values below 1 were discarded when calculating the error to avoid values exploding and painting an unrealistic error image.

Figure 4.4.1 shows that several data points from well 1, 2, 3, and 5 had an error of more than 200%. Even though it is difficult to determine in what range the true values lie, an error of more than 100% is substantial.

4.5 Hybrid flow model with pretraining

Table 4.5 shows the results of the hybrid flow model when it was pretrained with the generated fake data. The first row for each well in Table 4.5 is the same results as shown in Table 4.1, included here for reference (the standard data-driven flow model). The second and third row shows the performance of the hybrid flow model when fake data was generated by drawing variable values from a uniform and well-specific distribution, respectively, and label generation by calculating the total flow rate with the mechanistic flow rate model (2.3.1).

Well 1, 2, 3, and 4 performed the best in this experiment when it was pretrained with fake data generated from a well-specific distribution. Well 1 performed remarkably well when it was pretrained with well-specific data, even though it in total trained and pretrained for a relatively small number of epochs compared to the other wells. Well 1, together with well 2, 3, and 5 performed significantly worse when they were pretrained with data drawn from uniform distributions. Well 5 performed significantly worse when pretrained on data from both a uniform and a well-specific distribution.

Table 4.5: The table shows the performance of the hybrid flow model when it was pretrained with generated fake data. The Distribution column shows from what distribution the generated fake data was drawn from, while the Pre-epochs column shows for how many epochs the model pretrained on the generated fake data. The best-performing configuration for each well is highlighted in bold.

Well	LR	l_2-reg	Distribution	Pre-epochs	Epochs	MAE	MSE	MAPE
			N/A	N/A	2021	0.9853	1.4584	2.5285%
W1	0.002	0.003	Uniform	1449	1786	1.0951	1.6355	2.8599%
			Well-specific	1320	505	**0.6147**	**0.6675**	**1.6284%**
			N/A	N/A	529	2.8849	12.9182	34.7340%
W2	0.003	0.005	Uniform	547	500	3.1695	14.8367	38.3477%
			Well-specific	7395	500	**2.7692**	**12.3748**	**32.9247%**
			N/A	N/A	7119	0.1886	0.0116	2.5634%
W3	0.002	0.003	Uniform	569	503	0.5065	0.4512	6.2968%
			Well-specific	584	4797	**0.1532**	**0.0395**	**2.2758%**
			N/A	N/A	1324	2.3778	6.3680	7.3569%
W4	0.003	0.005	Uniform	1892	936	1.9340	4.4174	6.1655%
			Well-specific	2468	1847	**1.8825**	**4.1871**	**5.9419%**
			N/A	N/A	2819	**2.2337**	**7.8562**	**8.8449%**
W5	0.002	0.003	Uniform	1054	505	4.4696	24.1579	19.1662%
			Well-specific	635	503	4.1315	20.8879	17.3225%

4.6 Hybrid flow model with physics-informed loss function

This section visualize the results from the hybrid flow model trained with a PILF. The results are shown in Table 4.6. The first row for each well is the same results as shown in Table 4.1, included here for reference (the standard data-driven flow model). The second row for each well in Table 4.6 shows the results of the hybrid flow model where the loss function was modified to penalize deviations from the mechanistic flow model. The results of the hybrid flow model seem to vary significantly between the five wells. Well 1, 2, and 4 perform better than the data-driven flow model, while well 3 and 5 perform significantly worse.

Table 4.6: The table shows the performance of the hybrid flow model when the networks loss function was modified to penalize deviation from the models total flow rate estimate. The PILF column shows whether or not the cost function was modified to include penalty for deviations from the mechanistic flow model, and the Qdiff-reg column shows the regularization factor λ_H. The best-performing configuration for each well is highlighted in bold.

Well	LR	l_2-reg	PILF	Qdiff-reg	Epochs	MAE	MSE	MAPE
W1	0.002	0.003	No	N/A	2021	0.9853	1.4584	2.5285%
			Yes	0.001	2322	**1.3649**	**0.9410**	**2.4309%**
W2	0.003	0.005	No	N/A	529	2.8849	12.9182	34.7340%
			Yes	0.001	524	**2.6130**	**9.9352**	**32.0243%**
W3	0.002	0.003	No	N/A	7119	**0.1886**	**0.0116**	**2.5634%**
			Yes	0.001	3861	0.7037	0.5683	10.1391%
W4	0.003	0.005	No	N/A	1324	2.3778	6.3680	7.3569%
			Yes	0.001	924	**2.2124**	**5.5886**	**6.8813%**
W5	0.002	0.003	No	N/A	2819	**2.2337**	**7.8562**	**8.8449%**
			Yes	0.001	914	3.6890	18.8683	16.1341%

4.7 Hybrid flow model with physics-informed loss function and pretraining

The results presented in Table 4.7 show the performance of the hybrid flow model with it was pretraining with generated fake data drawn from a uniform and a well-specific distribution, including a PILF. The first row for each well is the same results as shown in Table 4.1, included here for reference (the standard data-driven flow model). The second row for each well shows the performance of the hybrid flow model when the neural network had a modified loss function as described in Section 3.4.1 (PILF) and was pretrained on 2000 generated fake data points drawn from a uniform distribution. The third row for each well shows the performance of the hybrid flow model when the neural network had a PILF and was pretrained on 2000 generated fake data points drawn from a well-specific distribution. The total flow rate in both distributions was calculated using the mechanistic flow rate model (2.3.1).

Well 1 shows a slightly improved performance when pretraining on well-specific generated data and using a PILF compared to the standard data-driven flow model. The same can be said for well 2, but well 2 does not appear to differ much between pretraining on uniform or well-specific drawn data in this experiment. Well 3 and 5 performed significantly better without the hybrid flow model in this experiment. At the same time had well 4 a slightly improved performance with pretraining on fake data drawn from a uniform distribution.

Table 4.7: The performance of two variants of hybrid flow modeling compared to the standard data-driven flow model. The Qdiff-reg column has been removed from the table due to horizontal space, but the regularization factor λ_H was set to 0.001 for all models. The best-performing model for each well is marked in bold.

Well	LR	l_2-reg	Distribution	PIFL	Pre-epochs	Epochs	MAE	MSE	MAPE
			N/A	No	N/A	2021	0.9853	1.4584	2.5285%
W1	0.002	0.003	Uniform	Yes	1449	1787	1.5384	1.0537	2.7585%
			Well-specific	Yes	1200	995	**1.1491**	**0.9321**	**2.4411**%
			N/A	No	N/A	529	2.8849	12.9182	34.7340%
W2	0.003	0.005	Uniform	Yes	547	502	9.4428	2.5183	30.7590%
			Well-specific	Yes	614	502	**9.0024**	**2.4925**	**30.5557**%
			N/A	No	N/A	7119	0.1886	0.0116	**2.5634%**
W3	0.002	0.003	Uniform	Yes	500	9249	0.4011	0.6062	8.9265%
			Well-specific	Yes	10000	2270	0.3978	0.6016	8.9662%
			N/A	No	N/A	1324	2.3778	6.3680	7.3569%
W4	0.003	0.005	Uniform	Yes	1158	1473	**5.5243**	**2.2121**	**6.8276**%
			Well-specific	Yes	2370	1184	6.5192	2.4257	7.4106%
			N/A	No	N/A	2819	**2.2337**	**7.8562**	**8.8449**%
W5	0.002	0.003	Uniform	Yes	546	2477	10.3116	2.6883	10.8747%
			Well-specific	Yes	661	502	3.7064	17.3806	15.6503%

Chapter 5

Discussion

In this chapter, the results presented in Chapter 4 will be presented. The discussion is divided into two, where the thesis methodology is discussed in Section 5.1, and the discussion of the results is discussed in Section 5.2

5.1 Discussion of methodology

5.1.1 Preprocessing of data

The preprocessing of the data was carried out as described in Section 3.2. The preprocessing of the dataset removed some data points with filters that obviously were not representative of the entire dataset (for each well). However, more effort should probably have been put into engineering the filters and possibly split them into different wells instead of treating the whole dataset (from all the wells) as one. A risk of removing too many data points is that data points that appear as a little bit different can be removed even though they might be valid. Filtering out data point is therefore an assessment that is challenging to tune perfectly.

Relatively few data points were removed in total (22). Therefore, not significant amounts of representative data were discarded, but especially filter 6 and 7 (seen in Table 3.3) should have been adapted to each individual well. From Figure 3.4.2 it becomes clear that almost all total flow rate variables for well 3 are below 10, which makes filter 6 quite useless for this well. However, there may be data points in well 3 that have unrealistically high total flow rates compared to choke opening, but filter 6 would not be able to catch that. The same applies to filter 7, which was not very well adapted to the total flow rate distribution of well 3 (and well 2).

Filter 2, 3, and 4 turned out to be excessive, but did not do any harm, regardless. The fact that filter 3 and 4 did not filter out any data points indicates that the provided data was realistic. Filter 2 could probably have been tuned a little bit better, as the wellhead pressure realistically needs to be a certain amount larger than the downstream choke pressure. In hindsight, decreasing the threshold for what data points filter 2 to the data

points with a wellhead pressure lower than 0.05 (i.e. filtering more data points) only discards 2 more data points, but does probably not have a significant impact on the final results. Filter 5 only filtered out one data points, and could therefore have been tweaking a little bit up and down to research whether the data just was of a high quality, or if the threshold for what data points to filter out should have been decreased.

The split between training data and validation data was not adjusted after the data was provided. Since the dataset for each well is relatively small (especially for well 3 and possibly for well 2), techniques such as k-fold validation could have been implemented to be able to train on a larger portion of the data during the initial experimentation [45].

5.1.2 Hyperparameters

As mentioned in Chapter 3, the hyperparameters were adopted in large part from the project report [7]. The objective of the thesis was not to find the best set of hyperparameters. Since hyperparameter optimization is a challenging problem and require an additional optimization problem on top of the standard parameter estimation problem as described, the research question would probably not have benefited much from spending more time on tweaking the hyperparameters. However, during the PILF experiments, 0.001 was used as the regularization factor λ_H for all wells. It is not given that this regularization factor works great for all wells and should, in hindsight, probably have been adjusted a little bit back and forth in case the factor was out of proportions for one of the wells.

5.1.3 The mechanistic flow model

The mechanistic flow model used as the physics-informed part of the hybrid flow models is one of the simplest flow models there is and it was assumed not to be a very good model for the choke valve. However, the model contains information about the pressure drop over the choke and may guide the data-driven flow model to learn appropriate correlations. As Guo *et al.* [4] describes, many flow models exist, and the Sachdeva model, as presented in (2.3.2) is probably better suited for accurately modeling a choke valve. It should be noted that usage of the Sachdeva flow model would include a broader set of assumptions, including the ones that was used in (2.3.1). However, it may be possible to make realistic assumptions and adjustments to utilize the Sachdeva model regardless. When that is said, the mechanistic part of the used hybrid flow models is not very dominant, and a flow model with a little bit lower accuracy (as the one used) may not be that impactful. Suppose a more complex and accurate flow model were to be used. The hybrid flow model may in that case have benefited from being more impacted by the mechanistic flow model, but that remains for another research question.

There is no perfect mechanistic flow model, and adjustments to a factor such as the valve discharge coefficient C_D must be made regardless of the model used. The tuning of C_D was found with trial and error in this thesis, as a more complex tuning approach was not assessed as necessary. A more analytical approach, such as summing the squared errors and calculating the mean over the dataset, could perfectly center the error of the graph produced by the mechanistic flow model around zero. However, it is challenging to know

to what extent this would provide the data-driven flow model with better information about the mechanistic correlations in the data.

5.1.4 The data-driven flow models

The data-driven flow model of this thesis was a fully connected feed-forward neural network trained with back-propagation. The ReLU activation function was used with the Adam optimizer. The data-driven flow model appeared to work very well during the initial testing phase and performed satisfactorily, as seen in Section 4.1, Section 4.2 and Section 4.3, although the results varied. There exists research, such as Li *et al.* [42], arguing that pretraining of a neural network does not work very well when using the ReLU activation function, hence more time should probably have been spent on researching the variations between different activation functions. When that is said, there are no direct indications of better results by changing neither the activation function nor the optimizer, and also the He initialization as suggested by He *et al.* [24] worked well. A more complex optimizer than a first-order gradient-based optimizer, such as a second-order gradient-based optimizer, could have been experimented with. However, a more complex optimizer is computationally heavier, and a first-order optimizer seems to work satisfactorily.

As mentioned in Chapter 2, there exist other data-driven models, but one of the most versatile is the neural network. However, other variants of neural networks than the one used here exist. The most interesting in this context is the Recurrent neural network (RNN) which may produce a data-driven model that performs even better than a traditional neural network when the data is time-dependent, which is the case for a hydrocarbon production well as mentioned in Section 3.2. A RNN requires more effort in implementation and training and is more computational heavy than a standard neural network. It is impossible to know whether a RNN would perform better without testing, but there is no apparent reason why hybrid flow models would have performed better with a RNN. Since the dataset for each well is not particularly large, and data-driven models are quite data-demanding, there might exists other data-driven methods that are better suited to adapt to the total flow rate through a choke valve.

5.1.5 The hybrid flow models

The model that included a PILF was developed simply by adding a term $H(\boldsymbol{\theta})$ to the existing loss function as described in Section 3.4.1. The $H(\boldsymbol{\theta})$ term got bigger if the model prediction deviated from the pre-calculated total flow rate using (2.3.1). It is trivial to see from Figure 3.3.2 that the mechanistic flow model does not perfectly matches the provided data.

During the generation of fake data, the distribution of each variable for each well was plotted as a histogram as a guide to define the upper and lower limit in which to draw random values from, as seen in Figure 3.4.7 and Figure 3.4.6. The results would have been the same, but it would probably have been easier to adjust the absolute numbers upper and lower limits instead of guessing the percentiles. In other words, with Figure 3.4.7 as

an example, it is easier to define the lower limit of 0.220 rather than to tune by guessing what percentile would hit the 0.220 line.

Figure 3.4.9 makes it easy to compare the generated data from a uniform distribution with the real data for well 2. It becomes clear from the plot that the percentile method worked and that all true data points except a few outliers were within the defined limits as intended, but it also becomes clear that most of the data are not in the same neighborhood as the real data. The point of pretraining with generated fake data from a uniform distribution was to prepare the model for real data it had not seen before, disjoint from the training dataset. However, it becomes very clear from the plot that correlations within the real data are not present in the generated fake data. It is evident that there is a correlation between DeltaP (the differential pressure between the downstream choke pressure and the wellhead pressure) and the choke opening, which is not present in the generated data. It is also safe to assume that there is some kind of time correlation within the data that is not mirrored in the generated data.

However, the generated fake data is much denser than the real data. A sparse dataset would result in a less robust model as it would not be able to learn how to behave outside of the spare data, but maybe data as dense as seen in Figure 3.4.9 was too dense.

The regularization term λ during pretraining was set to 0.0005, and the learning rate α was set to 0.005. As with the standard data-driven flow model, these hyperparameters were not dedicated too much effort but rather decided after a couple of test runs (testing on validation data) where they performed satisfactorily. The learning rate increased slightly compared to the standard data-driven flow models, as the pretraining was assessed to take no harm of faster learning and hence convergence. The regularization during the pretraining was lowered quite a bit compared to the standard data-driven flow models, as overfitting to the generated data was not assessed as a problem due to the training that would take place after the pretraining. In hindsight, potential overfitting to the uniformly distributed data may have contributed to the model performing worse than its potential if that data turns out to not very well reflect the correlations in the real data.

The situation is not quite the same when the model was pretrained on data that were generated from a well-specific distribution. Figure 3.4.5, Figure 3.4.4, and Figure 3.4.3 makes it clear that the generated data is distributed more closely to the real data, and that the correlation between the variables is higher than compared with the uniform distribution in Figure 3.4.9.

Suppose the data in the test dataset has approximately the same distribution as the training data, and the generated data is of the same distribution as the training data. In that case, the pretrained model may perform better than a model that has been pretrained on data drawn from a uniform distribution. If the test data, however, is denser than the training data, a model that has been pretrained on generated data drawn from a uniform distribution may improve the models performance.

For well 5, a scatter plot of the test data and the generated fake data may be seen in Figure 5.1.1 and Figure 5.1.2. For well 3, a scatter plot of the test data and the generated fake data may be seen in Figure 5.1.3 and Figure 5.1.4. The similar plots for well 1, 2, and 4 will not be discussed but may be seen in Appendix A.7. These plots were generated

after the final experiments.

The test data of well 5 is for both well-specific and uniform distribution within the boundaries of the generated fake data. It is easy to see that the fake data points generated from a well-specific distribution seen in Figure 5.1.1 are more centered around the test data than the generated fake data points from a uniform distribution seen in Figure 5.1.2. However, Table 4.5 shows that pretraining generally performed poorly. The model pretrained on generated fake data from a well-specific distribution performed slightly better than the model that was pretrained on generated data from a uniform distribution, which is in conformity with the observations from Figure 5.1.1 and Figure 5.1.2.

In contrast to well 5, the results from well 3 is a little bit more challenging to interpret. The test data appear to partly be outside the test data boundaries for both models that were pretrained on generated fake data as seen in Figure 5.1.3 and Figure 5.1.4. However, Table 4.5 shows that the model that was pretrained on well-specific data performed much better than the model that was pretrained with data from a uniform distribution. When that is said is it still difficult to interpret the two scatter plots and determine which of the fake data distributions are the least unfavorable regarding the test data. The pattern of the test data points that is outside of the generated fake data boundaries seen in Figure 5.1.4 may be due to slowly changing well conditions.

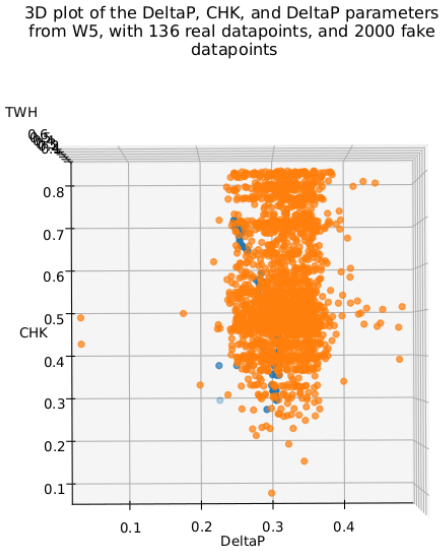


Figure 5.1.1: The plot shows the generated fake data drawn from a well-specific distribution fitted to well 5 in orange, and the test data from well 5 in blue. The plots shows that all the test data is within the boundaries of the generated fake data.

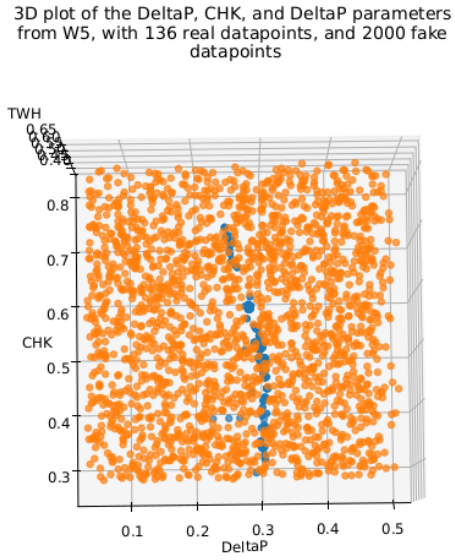


Figure 5.1.2: The plot shows the generated fake data drawn from a uniform distribution for well 5 in orange, and the test data from well 5 in blue. The plots shows that all the test data is within the boundaries of the generated fake data, but that the generated fake data is much sparser than the test data.

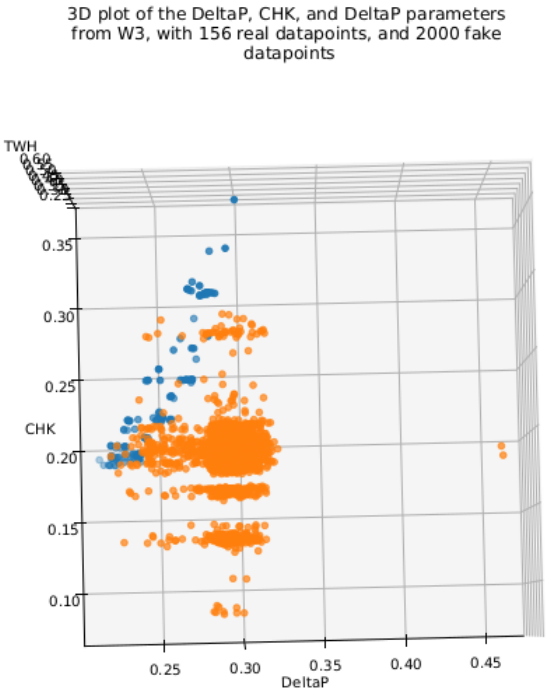


Figure 5.1.3: The plot shows the generated fake data drawn from a well-specific distribution fitted to well 3 in orange, and the test data from well 3 in blue. The plots shows that the similarities between the two data sets is questionable, as it is not clear if the fake data mirrors the distribution of the test data in a satisfactory manner.

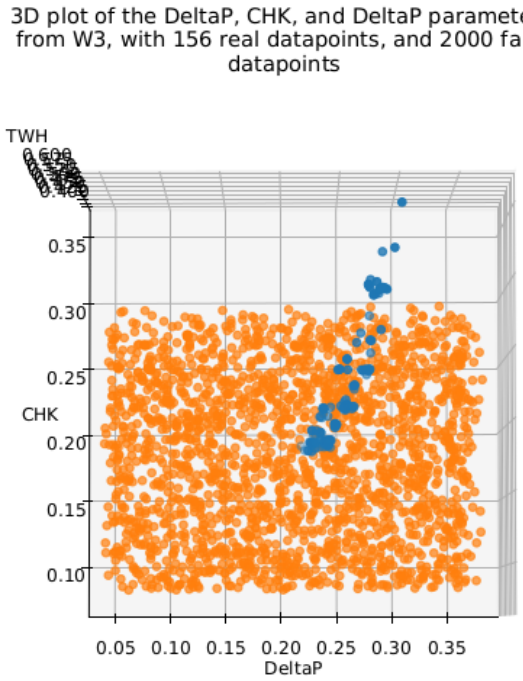


Figure 5.1.4: The plot shows the generated fake data drawn from a uniform distribution for well 3 in orange, and the test data from well 3 in blue. The plots shows that not all the test data is within the boundaries of the generated fake data.

Table 4.5 also shows that both models on well 5 trained for a concise amount. The model from well 3 was pretrained on well-specific data trained on the real training data for 4797 epochs, while the model that was pretrained with uniform data only trained on real training data for 503 epochs. It becomes clear that it is difficult to draw any conclusions only based on these plots. However, a small indication might be seen that pretraining on generated fake data from a well-specific distribution is favorable if pretraining is a given to be used.

5.2 Discussion of results

5.2.1 Results from the data-driven flow models

The results seen in Table 4.1 show that the data-driven flow models for well 1 and 3 performed very well without any modifications. The model performed relatively well for well 4 and 5, while well 2 performed relatively poorly. It is therefore clear that there is room for improvement.

The performance of a data-driven flow model trained on a merged dataset as described in Section 3.3 can be seen in Table 4.3. The results appear at first sight to be a little bit counterintuitive. When a merged dataset from well 2 and 3 was used as training data, well 2 and 3 were expected to perform better. The table clarifies that both wells performed

worse, which also applies for wells 1 and 5. Well 4 surprisingly performed a little better when trained on data from well 2 and well 3, but not significantly.

When a merged dataset from well 1, 4, and 5 was used as training data, those wells were expected to perform better, but as in the previous case, all of these wells performed worse than before. Well 3 performed much worse, but well 2 surprisingly performed much better when trained on that dataset. Although a MAPE of 16.1326% overall is not a very impressive performance, the model performed more than twice as accurately compared to before.

While well 2 and 5 performed better than before, well 1, 3, and 4 performed worse when merged data from all the wells were used as the training dataset. It should be noted that the merged data from all wells resulted in the best performance out of all experiments for well 5 when merged data from all wells was used as training data. It should also be noted that most wells trained for a relatively short number of epochs when training on a merged dataset, with one exception for well 4 when it trained on merged data from well 2 and 3, and one exception for well 1 when it trained on merged data from well 1, 4, and 5. A reason for the low number of epochs may be that the validation (and test) data has a different distribution than for the respective wells and that the data from the wells that were merged was not as equal as first assumed.

The performance of the data-driven flow model when noise was added to the label is presented in Section 4.3. Adding normally distributed noise to the label improved the performance of three of five wells even though there was already noise from the measurement equipment in the data. The improved performance of well 1 and 2 appears to be very marginal and does not indicate a significant performance boost. However, for well 4 the accuracy is improved by 35%, which is a very good performance boost. It would have been interesting to experiment further with the improved model for well 4, but such experiments were not prioritized.

5.2.2 Results from the mechanistic flow model

A plot showing the deviation between the total flow rate calculated by the mechanistic flow model (2.3.1) and the total flow rate provided in the dataset can be seen in Figure 4.4.1. The plot is scaled with a logarithmic scale to make it easier to interpret, since the magnitude of the error ranges over a large scale.

More than 50% of the data points for all the wells except well 3 had an error of more than 10%, as the median line in the boxplot lies above the 10^{-1} line. Many data points had an error of less than 1% (the 10^{-2} line), but some also missed more than 100% (the 10^{0} line). It is difficult to interpret in what range the proper total flow rate value lies, but as the mechanistic flow model was expected not to fit the true data very well, the errors seen in Figure 4.4.1 might not be too bad.

5.2.3 Results from the hybrid flow models

The results of the hybrid flow model with pretraining can be seen in Section 4.5. Well 1, which has performed exceptionally well throughout every experiment, performed even better when the model was pretrained using generated data from a well-specific distribution. The improvement was greater than 35%. Also, notice that the model for well 1, when pretraining on generated data drawn from a uniform distribution, made the model perform slightly worse. Well 2 performed in the same pattern as well 1, with a slight improvement when the model was pretrained on generated data drawn from a well-specific distribution and worse when it was pretrained on uniformly distributed generated data. There is also a considerable difference in the number of epochs for well 2. Both pretraining models early stopped as soon as possible as the patience in the early stopping algorithm was set to 500 for all experiments.

Well 3 also performed in the same pattern as well 1 and 2, with better performance than the standard data-driven flow model when the model was pretrained on data generated from a well-specific distribution and worse when it was pretrained on data drawn from a uniform distribution. Also, notice that the hybrid flow model trained on fake data from a uniform distribution did not pretrain for very long, and in total trained for about one-seventh as much as for the standard data-driven flow model. Well 4 broke the pattern for the three previous wells and performed better than the data-driven flow model for both distributions. It performed the best when pretrained on data from a well-specific distribution and trained for the highest number of epochs compared with the other experiments for well 4.

As four out of five wells performed the best when the models were pretrained on data from a well-specific distribution, it is easy to conclude with great potential for improved performance when using this approach. Well 5 performed much worse when the model was pretrained with both distributions, which is surprising as the other four wells performed the best when they were pretrained with generated data from the well-specific distribution. However, as mentioned in Chapter 3 did each experiment run once, so the results might appear better than they are if the experiments had been conducted several times with different random seeds. Especially the excellent result for well 1 might be too good to be true, which also will be discussed when discussing the results from the hybrid flow model with PILF and pretraining.

The hybrid flow models that contained a PILF may be seen in Section 4.6. All models were trained with a regularization factor λ_H value of 0.001 multiplied with the hybrid term $H(\boldsymbol{\theta})$. The value of 0.001 worked satisfactorily for all wells, but probably could have been better adjusted to fit the individual wells. Regardless, three out of five wells turned out to perform better with PILF compared to the data-driven flow model. The hybrid flow model with PILF performed a little better in well 1 compared to the performance of the data-driven flow models. The same goes for well 2 and 4, which performed the best with the PILF, but the results are not unprecedented. Well 3 and 5 performed much worse when the model included the PILF.

It should be questioned how well the mechanistic flow model that is embedded in the PILF is suited for guiding the neural network to better performance. The results presented in Section 4.4 make it clear that the mechanistic flow model seen in (2.3.1) does not match the data very well, and it is not apparent how much the mechanistic flow model should

be trusted. However, if every data point calculated by the mechanistic flow model had had an error very close to 0%, the $H(\boldsymbol{\theta})$ term would contribute with very little additional information. The term would penalize just like the mean square error term $R(\boldsymbol{\theta})$. When that is said, more than half of the data points for well 1, 2, 4, and 5 deviated by more than 10%, so it is difficult not to question the accuracy of the mechanistic flow model. It may also be the case that the flow meter that collected the data provided had an error bias, but it challenging to conclude upon.

The results of the hybrid flow model with pretraining and PILF can be seen in Section 4.7. Table 4.7 shows that the results of the hybrid flow model vary greatly between the five wells. The performance of the hybrid flow model on well 1 had a marginal improvement compared with the data-driven flow model when the hybrid flow model was pretrained on data from the well-specific distribution. As well 1 already has a very good performance without pretraining, a large improvement is not very likely to occur. However, for well 1, out of the three hybrid flow models the one with pretraining and PILF performed the worst out of those three. There was only a marginal improvement in the hybrid flow model with pretraining and PILF compared with the hybrid flow model with PILF, which makes the hybrid flow model that was pretrained on well-specific generated data without PILF look suspiciously good. The hybrid flow model tested on well 1 with pretraining on uniform data and containing a PILF performed worse than the data-driven flow model but slightly better than the hybrid flow model that only was pretrained on uniformly distributed data.

Well 2 performed better with every hybrid flow model introduced in the thesis (except one) compared with the data-driven flow model. For well 2, out of the best performing hybrid flow models for each experiment, the hybrid flow model with pretraining and PILF performed the best. The model tested on well 2 seems to work equally well when pretraining on uniformly distributed data and well-specific data. Only regarding the hybrid flow model tested in this section, well 2 had the shortest number of training epochs out of all the wells. One should be careful to conclude, but it seems like there is a correlation between a low number of training epochs and performance, even though the hybrid flow model with pretraining on well-specific data is an exception for well 2. Even though the hybrid flow models appear to perform best no matter what for well 2, none of the hybrid flow models are close to the performance of the data-driven flow model when it was trained on merged data from well 1, 4, and 5 as seen in Table 4.3.

The hybrid flow model with pretraining and PILF tested on well 3 did not perform exceptionally well. Both hybrid flow models in Section 4.7 performed worse than the data-driven flow model, and the model does not seem to favor either of the two hybrid flow models over the other. However, when tested on well 3, the hybrid flow model with pretraining on data from both distributions and PILF performed better than the hybrid flow model that only had a PILF.

The performance of the hybrid flow model with pretraining and PILF tested on well 4 was slightly better when the pretraining was with data from a uniform distribution. However, the performance was not better than the performance of the data-driven flow model when noise was included, as can be seen in Section 4.3. This experiment was the only experiment during the whole thesis where the model performed best with a uniform distribution, which appears a little bit random.

All hybrid flow models tested on well 5 performed worse than the data-driven flow model did, and as seen in Table 4.3 was the data-driven trained with merged data from all wells the only model that performed better than the standard data-driven flow model trained with data from well 5. The performance of the hybrid flow model with pretraining on well-specific data and PILF performed particularly poorly, which is not very surprising when compared to the two hybrid flow models with only pretraining and only PILF. However, there is no apparent reason for the hybrid flow models to perform that much worse when tested on well 5 compared to when they were tested on the other wells.

A table showing the best results for each well with both hybrid flow models and data-driven flow models can be seen in Table 5.1.

Table 5.1: The best hybrid flow model and data-driven flow model for each well out of all the models.

Well	Model type	Specification	MAPE
W1	Hybrid	pretraining, well-specific distribution	1.6284%
	Data-driven	Trained with data from W1 with noise (1 std)	2.4210%
W2	Hybrid	pretraining and PILF, well-specific distribution	30.5557%
	Data-driven	Trained with data from W1, W4 and W5	16.1326%
W3	Hybrid	pretraining, well-specific distribution	2.2758%
	Data-driven	Trained with data from W3	2.5634%
W4	Hybrid	pretraining, well-specific distribution	5.9419%
	Data-driven	Trained with data from W4 with noise (0.5 std)	4.7472%
W5	Hybrid	pretraining and PILF, uniform distribution	10.8747%
	Data-driven	Trained on data from all wells	8.6083%

The results from Jacobsen [7] reprinted in Table 4.2 show that there are even better results to be produced with a data-driven approach that utilizes feature engineering. However, the hybrid flow model did, as seen above, produce improved results with some of the tested configurations compared to the standard data-driven flow model.

Chapter 6

Conclusion and future work

6.1 Conclusion

As discussed in Chapter 5, the results presented in Chapter 4 show that several of the hybrid flow model experiments showed improved performance compared to a standard data-driven flow model. However, it is not straightforward to conclude by interpreting the results. Using a hybrid flow model achieved better performance on four out of five wells compared to the standard data-driven flow model without modifications. However, as seen in Table 5.1, other data-driven flow models outperformed hybrid flow models for three out of five wells. For one of the wells did no modified data-driven flow model (with noise or merged data), and no hybrid flow model outperformed the standard data-driven flow model. This may indicate that there is no generalizable method for achieving excellent performance. The results presented in Jacobsen [7] show that, applying feature engineering, four of five wells performed better than hybrid flow modeling. However, utilizing a hybrid flow model did improve performance in some situations.

As mentioned in the discussion, there exists research that indicated that pretraining is not optimal when using the ReLU activation function. However, the dataset in this thesis is not a standard dataset, and generalizing from other datasets to a dataset from five specific hydrocarbon wells should not be done without further ado. Since the data used to train the neural network in this thesis are time series data, there may be potential to utilize a RNN instead of a standard neural network. A weakness of this thesis is that only one random seed was used in the experiments. Therefore, the results may be slightly affected by randomness. Even though there is no reason to suspect much worse or much better results with different random seeds, a more robust result would have been produced if the experiments had been reproduced with several random seeds.

The discussion makes it clear that the hypothesis presented in Chapter 1 is not trivial to answer. Although there was only one well where the hybrid flow model outperformed the data-driven flow models, the potential for improved performance by utilizing hybrid flow modeling is clear. Therefore, the hypothesis is partly confirmed, but to make further and more robust conclusions, as well before utilizing the technology in the industry, more research should be conducted.

6.2 Future Work

The thesis partly proves that there is potential for improved performance by using a hybrid flow model. However, there are several directions to take in further development of the work presented. First, several wells should be included in the research, and more data should be collected to make the learning process of the neural network more robust. The experiments should also be reproduced with several random seeds.

Although the hyperparameters in this thesis made the several neural networks perform satisfactorily, more research on how to best pick hyperparameters in combination with hybrid flow modeling should be conducted to make sure that the potential behind the hybrid flow models is not restricted by adopting hyperparameters for a hybrid flow model from a data-driven flow model. Generating fake data has proven to be an interesting direction, and designing distributions to better prepare for training on real data seems to have potential that should be utilized. There may even be potential to only train a network only on generated fake data, which will be an interesting path forward in the research of VFMs.

It would also be interesting to explore further what causes the different datasets to perform the way they do when using hybrid flow models, i.e., by conducting a sensitivity analysis. Regarding the quality of the training data, the workings of Brunton *et al.* [46] on obtaining governing equations would be unprecedented if applicable to the VFM research field. Finally, combining feature engineering with hybrid flow modeling may be the most apparent direction forward from the work presented, as the results presented in Table 4.2 would be a good starting point for further improvement. Improved performance from that point would result in very satisfactory model performances.

Bibliography

[1] BASF, "Petrochemicals BASF," May 2022. eprint: \\https://chemicals.basf.com/global/en/Petrochemicals.

[2] N. petroleum, *Exports of oil and gas*, https://www.norskpetroleum.no/en/production-and-exports/exports-of-oil-and-gas/, [Online; accessed 25-May-2022], May 2022.

[3] N. petroleum, *The service and supply industry*, https://www.norskpetroleum.no/en/developments-and-operations/service-and-supply-industry/, [Online; accessed 25-May-2022], May 2022.

[4] B. Guo, X. L. Liu, and X. Tan, *Petroleum Production Engineering*. Elsevier, 2017.

[5] G. Falcone, G. F. Hewitt, C. Alimonti, and B. Harrison, "Multiphase flow metering: current trends and future developments," *Journal of Petroleum Technology*, 2002.

[6] G. Falcone, G. F. Hewitt, and C. Alimonti, *Multiphase flow metering, principles and Applications*. Elsevier Science & Technology Books, 2010.

[7] P. Jacobsen, "TTK4550 project report," 2021.

[8] M. Hotvedt, B. Grimstad, and L. Imsland, "Developing a hybrid data-driven, mechanistic virtual flow meter-a case study," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 692–11 697, 2020.

[9] J. Henriksson, L. Wollebæk, and Z. Yang, "Hybrid Modeling for Multiphase Flow Simulations," in *Offshore Technology Conference*, OnePetro, 2022.

[10] G. Staff, G. Zarruk, J. Hatleskog, S. Stavland, H. McNulty, R. Ibarra, N. Calen, O. Hagemann, C. Lawrence, T. Selanger, *et al.*, "Physics guided machine learning significantly improves outcomes for data-based production optimization," in *Abu Dhabi International Petroleum Exhibition & Conference*, OnePetro, 2020.

[11] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, vol. 1, no. 1, pp. 1–34, 2020.

[12] T. A. AL-Qutami, R. Ibrahim, I. Ismail, and M. A. Ishak, "Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing," *Expert Systems with Applications*, vol. 93, pp. 72–85, 2018.

[13] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[14] M. Hotvedt, B. Grimstad, D. Ljungquist, and L. Imsland, "On gray-box modeling for virtual flow metering," *Control Engineering Practice*, vol. 118, 2022.

[15] "www.solutionseeker.no/," 2022.

[16] B. Guo, W. C. Lyons, and A. Ghalambor, *Petroleum Production Engineering, a computer-assisted approach*. Elsevier Science & Technology Books, 2007.

[17] G. Falcone and C. Alimonti, "The challenges of multiphase flow metering: today and beyond," International Conference on Offshore Mechanics and Arctic Engineering, 2007.

[18] *Ventilteknikk*. Norsk olje og gass, 2020.

[19] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*. Citeseer, 2007, vol. 2.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[21] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *arXiv:1811.03378v1*, 2018.

[22] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.

[23] S. K. Kumar, "On weight initialization in deep neural networks," *CoRR*, vol. 1704.08863, 2017. arXiv: 1704.08863.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," Proceedings of the IEEE international conference on computer vision, 2015.

[25] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, ser. Proceedings of Machine Learning Research, JMLR Workshop and Conference Proceedings, vol. 9, PMLR, 2010, pp. 249–256.

[26] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, Prediction*. Springer, 2008.

[28] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[29] N. Jorge and J. W. Stephen, *Numerical optimization*. Spinger, 2006.

[30] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[31] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," 2015.

[32] S. Baik, M. Choi, J. Choi, H. Kim, and K. M. Lee, "Meta-learning with adaptive hyperparameters," *arXiv preprint arXiv:2011.00209*, 2020.

[33] L. Prechelt, "Early stopping-but when?" In *Neural Networks: Tricks of the trade*, Springer, 1998, pp. 55–69.

[34] T. Bikmukhametov and J. Jäschke, "Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models," *Computers & Chemical Engineering*, vol. 138, p. 106 834, 2020.

[35] V. Cherkassky and F. M. Mulier, *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[36] A. Ansari, N. Sylvester, C. Sarica, O. Shoham, and J. Brill, "A comprehensive mechanistic model for upward two-phase flow in wellbores," *SPE Production & Facilities*, vol. 9, no. 02, pp. 143–151, 1994.

[37] J. G. Webster and H. Eren, *Measurement, Instrumentation, and Sensors Handbook: Two-Volume Set*. CRC press, 2018. [Online]. Available: http://dsp-book.narod.ru/MISH/CH28.PDF.

[38] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Marine Cybernetics AS, 2002.

[39] *Two-Phase Flow Through Chokes*, vol. All Days, SPE Annual Technical Conference and Exhibition, SPE-15657-MS, Oct. 1986. DOI: 10.2118/15657-MS. eprint: https://onepetro.org/SPEATCE/proceedings-pdf/86SPE/All-86SPE/SPE-15657-MS/2030021/spe-15657-ms.pdf. [Online]. Available: https://doi.org/10.2118/15657-MS.

[40] D. Solle, B. Hitzmann, C. Herwig, M. Pereira Remelhe, S. Ulonska, L. Wuerth, A. Prata, and T. Steckenreiter, "Between the poles of data-driven and mechanistic modeling for process operation," *Chemie Ingenieur Technik*, vol. 89, no. 5, pp. 542–561, 2017.

[41] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[42] J. Li, T. Zhang, W. Luo, J. Yang, X.-T. Yuan, and J. Zhang, "Sparseness Analysis in the Pretraining of Deep Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1425–1438, 2017. DOI: 10.1109/TNNLS.2016.2541681.

[43] B. Grimstad, V. Gunnerud, A. Sandnes, S. Shamlou, I. S. Skrondal, V. Uglane, S. Ursin-Holm, and B. Foss, "A Simple Data-Driven Approach to Production Estimation and Optimization," *In: SPE Intelligent Energy International Conference and Exhibition*, Sep. 2016.

[44] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016.

[45] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation.," *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[46] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

Appendix A

Appendix

A.1 The adam optimizer algorithm

---

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), $amsgrad$
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

---

for $t = 1$ **to** $\ldots$ **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / \big(1-\beta_1^t\big)$
  $\widehat{v_t} \leftarrow v_t / \big(1-\beta_2^t\big)$
  if $amsgrad$
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$

---

return θ_t

A.2 Flow model unit derivation

A unit analysis of (2.3.1) (reprinted in (A.2.2) to save the reader from scrolling) is derived in (A.2.1). Where $[Pa] = [\frac{kg}{ms^2}]$

$$[Q] = 1 * m^2 \sqrt{\frac{1}{\frac{kg}{m^3}}(Pa - Pa)} = m^2 \sqrt{\frac{m^3}{kg}\frac{kg}{ms^2}} = m^2 \sqrt{\frac{m^2}{s^2}} = \frac{m^3}{s} \tag{A.2.1}$$

$$Q = C_d A \sqrt{\frac{2}{\rho}(P_1 - P_2)} \tag{A.2.2}$$

A unit analysis of (2.3.2) (reprinted in (A.2.4) to save the reader from scrolling) is derived in (A.2.3). Where the adiabatic gas expansion coefficient κ is dimensionless, and $[Pa] = [\frac{kg}{ms^2}]$

$$[\dot{m}] = [1] * m^2 \left([2] \left(\frac{kg}{m^3}\right)^2 * Pa \left(\frac{[1]}{[1]-[1]} * [1] \left(\frac{[1]}{\frac{kg}{m^3}} - \frac{[1]}{\frac{kg}{m^3}} \right) + \left(\frac{[1]}{\frac{kg}{m^3}} + \frac{[1]}{\frac{kg}{m^3}} \right) ([1]-[1]) \right) \right)^{1/2}$$

$$[\dot{m}] = m^2 \left(\left(\frac{kg}{m^3}\right)^2 * Pa \left(\frac{[1]}{\frac{kg}{m^3}} \right) \right)^{1/2} = m^2 \left(\left(\frac{kg}{m^3}\right)^2 * \frac{kg}{ms^2} \left(\frac{m^3}{kg} \right) \right)^{1/2}$$

$$[\dot{m}] = m^2 \sqrt{\frac{kg^2}{m^6}\frac{kg}{ms^2} * \frac{m^3}{kg}} = m^2 \sqrt{\frac{kg^2}{m^4 s^2}} = m^2 \frac{kg}{m^2 s} = \frac{kg}{s} \tag{A.2.3}$$

$$\dot{m} = C_d A \left(2\rho_{dc}^2 p_{wh} \left(\frac{\kappa}{\kappa - 1} \eta_G \left(\frac{1}{\rho_{G,wh}} - \frac{p_r}{\rho_{G,dc}} \right) + \left(\frac{\eta_O}{\rho_O} + \frac{\eta_W}{\rho_W} \right) (1 - p_r) \right) \right)^{1/2} \tag{A.2.4}$$

A.3 Histograms of all variables for each well with percentiles

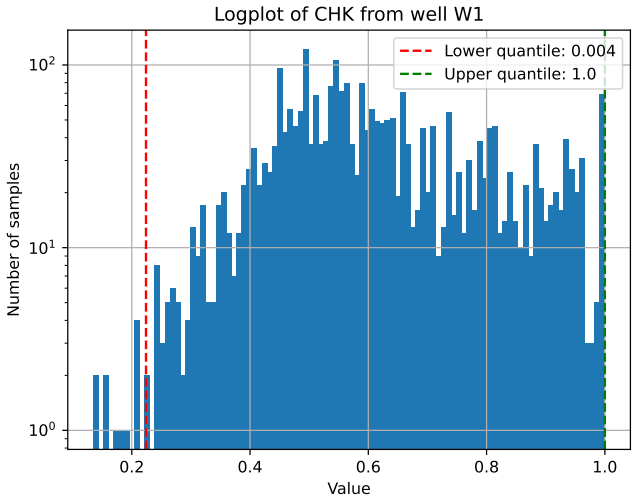


Figure A.3.1: The plot shows a histogram of the choke variable from the training data from well 1 with percentiles used when drawing fake data from a uniform distribution.

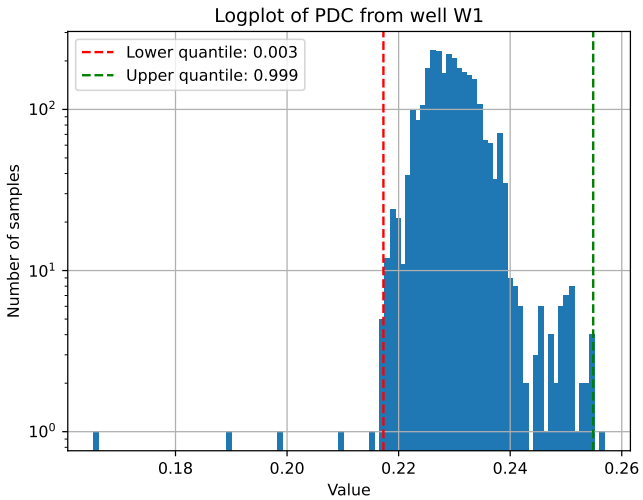


Figure A.3.2: The plot shows a histogram of the downstream choke pressure variable from the training data from well 1 with percentiles used when drawing fake data from a uniform distribution.

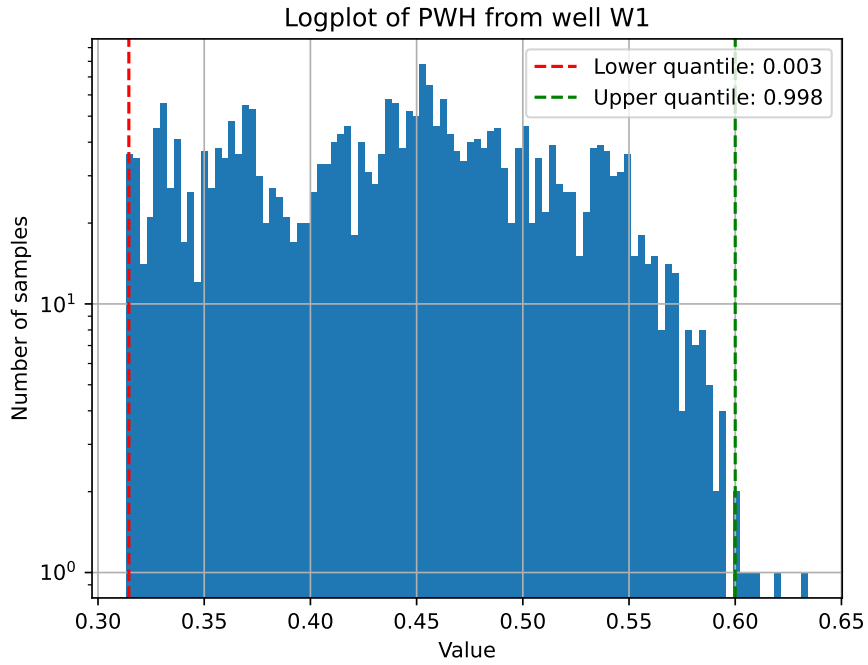


Figure A.3.3: The plot shows a histogram of the wellhead pressure variable from the training data from well 1 with percentiles used when drawing fake data from a uniform distribution.

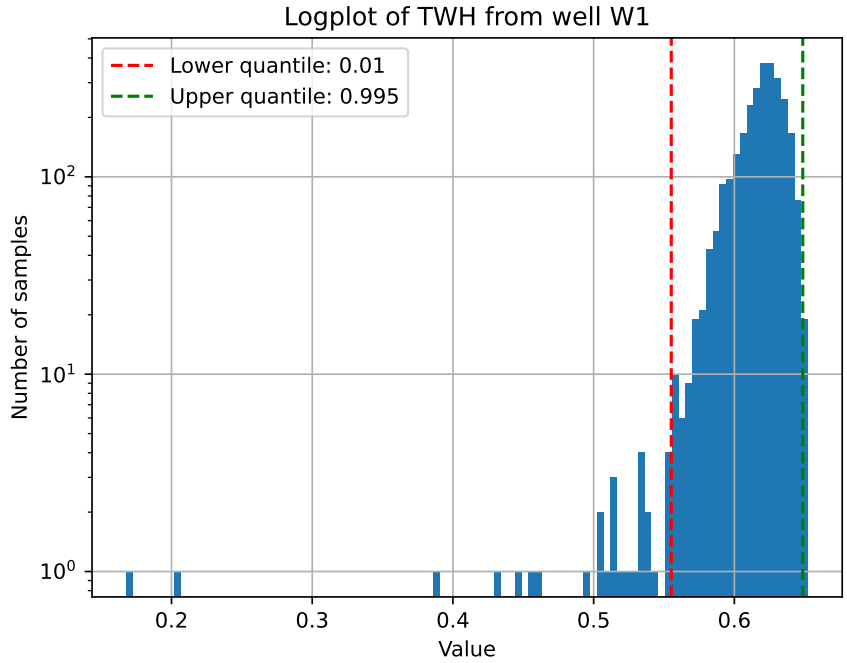


Figure A.3.4: The plot shows a histogram of the temperature wellhead variable from the training data from well 1 with percentiles used when drawing fake data from a uniform distribution.

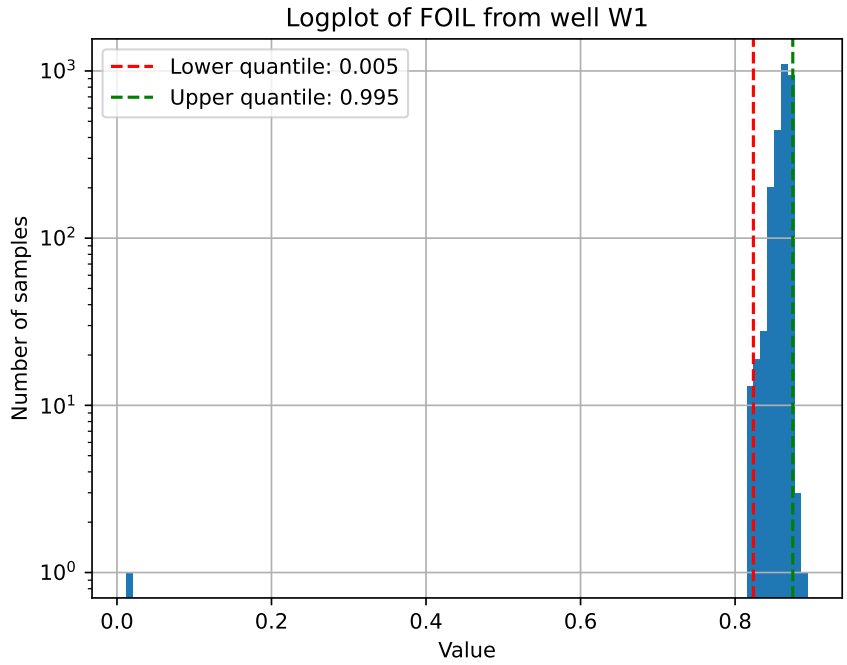


Figure A.3.5: The plot shows a histogram of the fraction of oil variable from the training data from well 1 with percentiles used when drawing fake data from a uniform distribution.

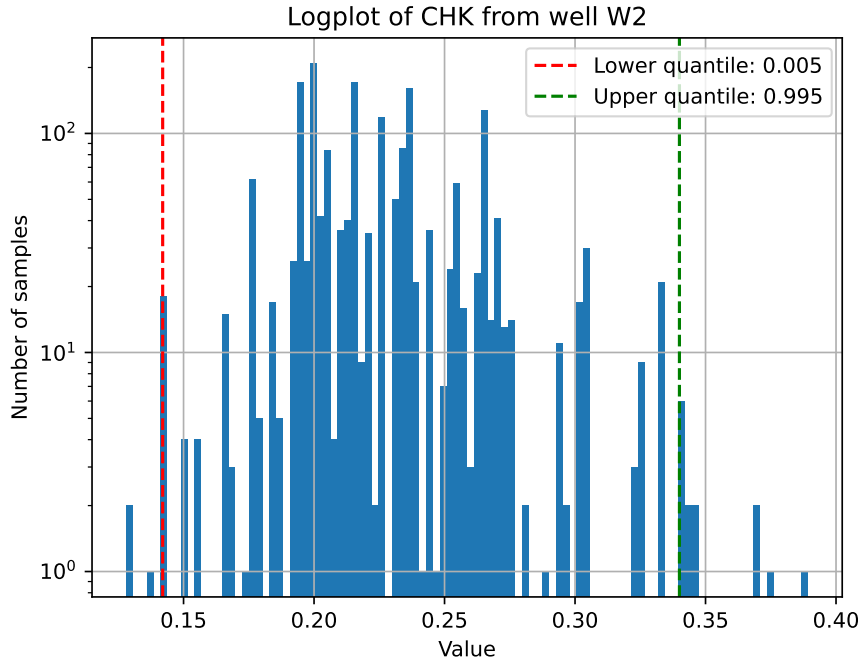


Figure A.3.6: The plot shows a histogram of the choke variable from the training data from well 2 with percentiles used when drawing fake data from a uniform distribution.

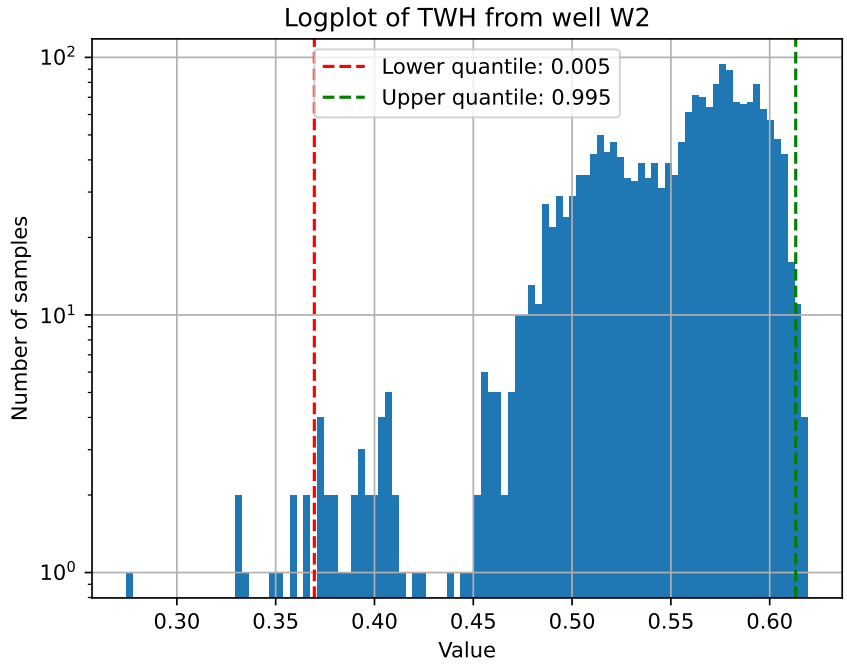


Figure A.3.7: The plot shows a histogram of the temperature wellhead variable from the training data from well 2 with percentiles used when drawing fake data from a uniform distribution.

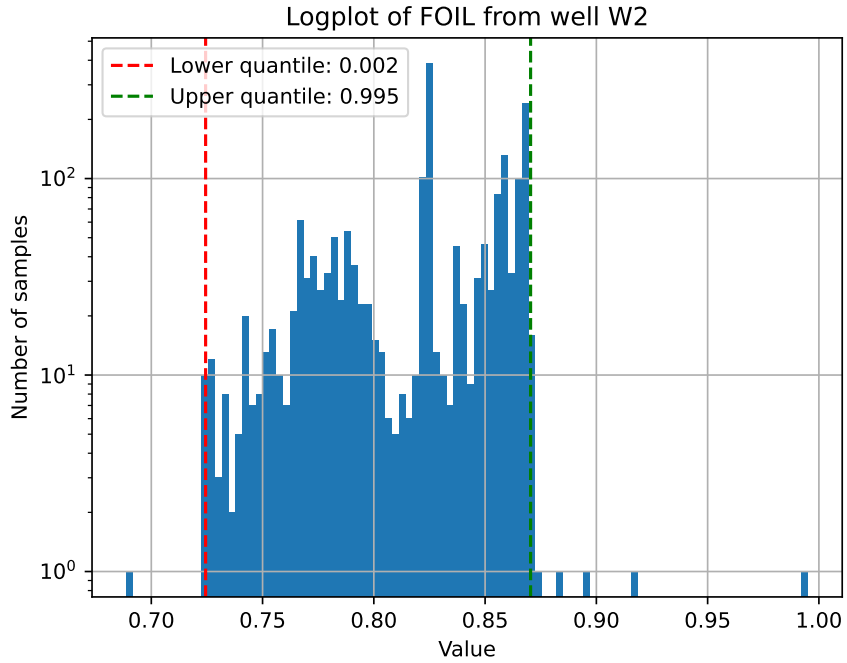


Figure A.3.8: The plot shows a histogram of the fraction of oil variable from the training data from well 2 with percentiles used when drawing fake data from a uniform distribution.

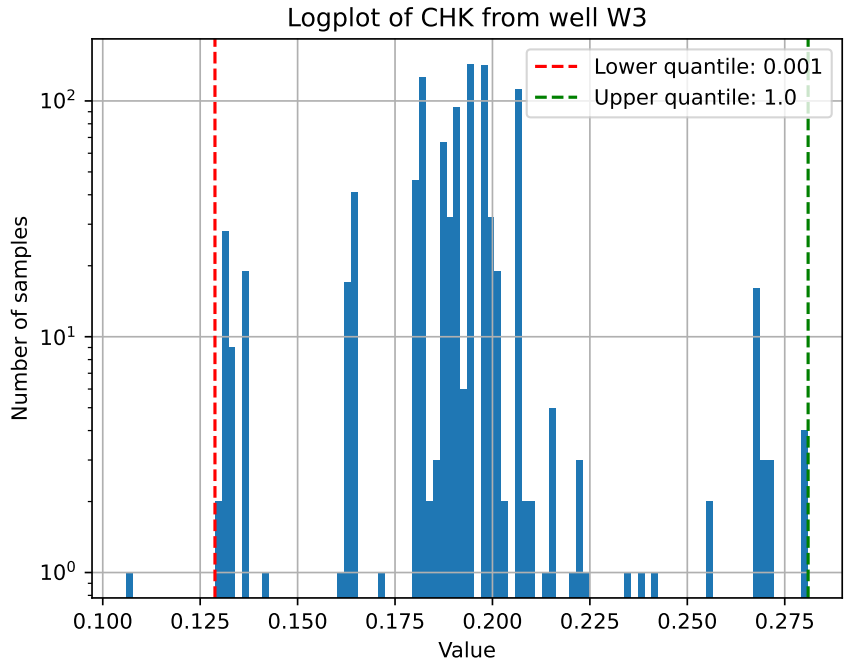


Figure A.3.9: The plot shows a histogram of the choke variable from the training data from well 3 with percentiles used when drawing fake data from a uniform distribution.

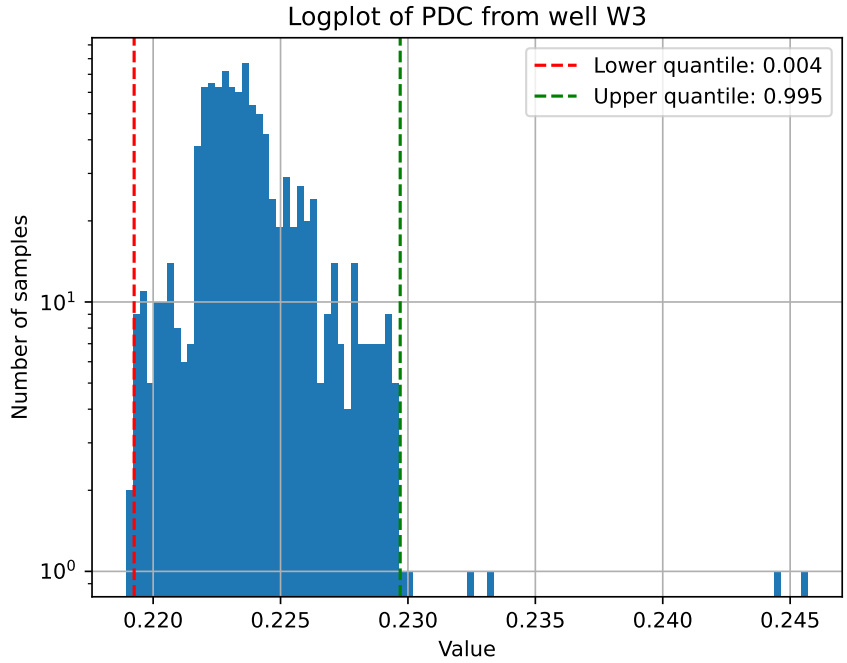


Figure A.3.10: The plot shows a histogram of the downstream choke pressure variable from the training data from well 3 with percentiles used when drawing fake data from a uniform distribution.

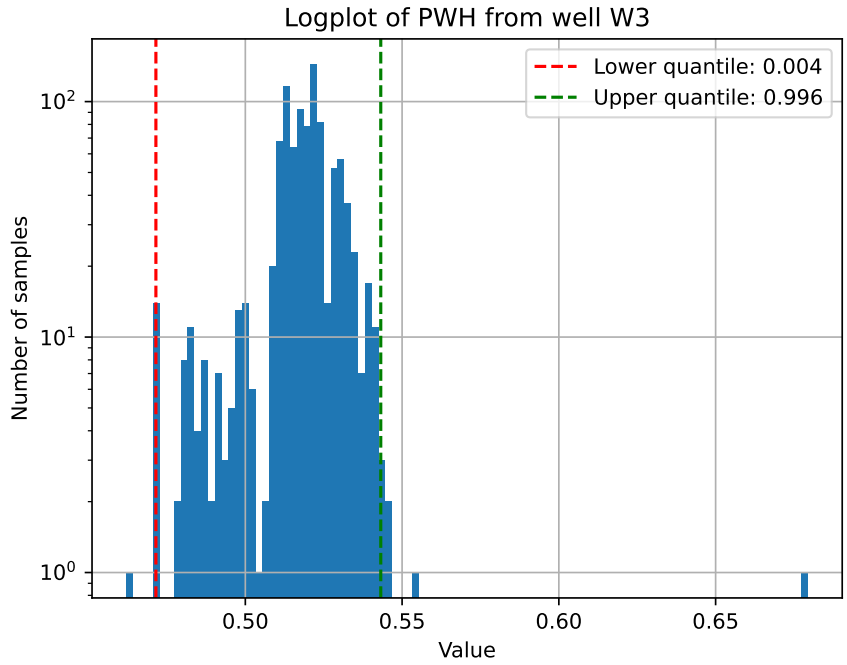


Figure A.3.11: The plot shows a histogram of the wellhead pressure variable from the training data from well 3 with percentiles used when drawing fake data from a uniform distribution.

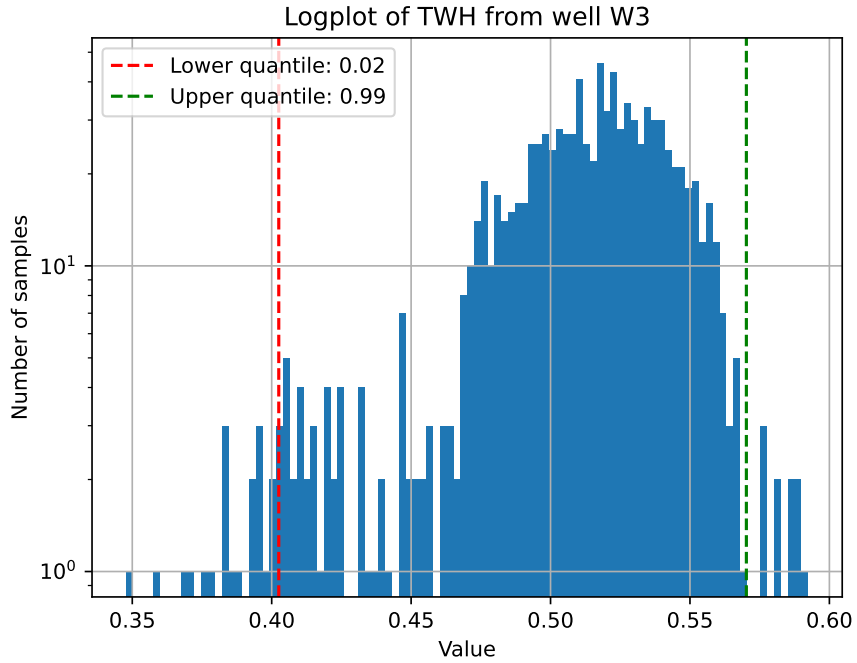


Figure A.3.12: The plot shows a histogram of the temperature wellhead variable from the training data from well 3 with percentiles used when drawing fake data from a uniform distribution.

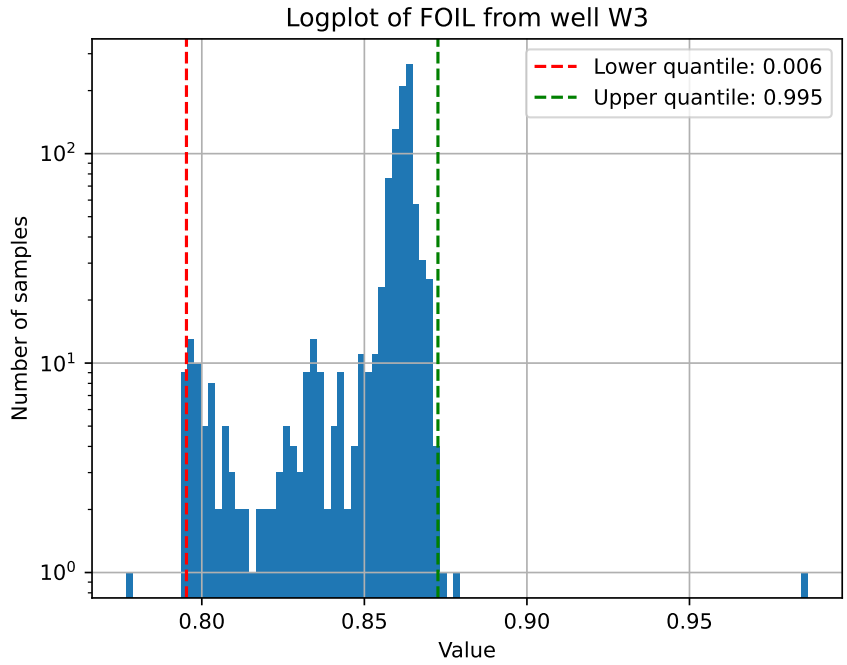


Figure A.3.13: The plot shows a histogram of the fraction of oil variable from the training data from well 3 with percentiles used when drawing fake data from a uniform distribution.

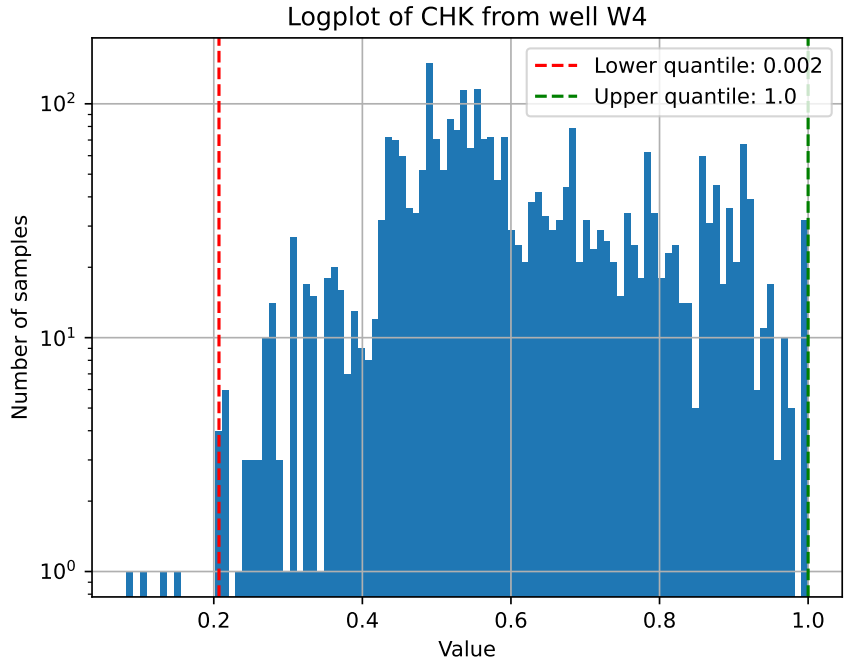


Figure A.3.14: The plot shows a histogram of the choke variable from the training data from well 4 with percentiles used when drawing fake data from a uniform distribution.

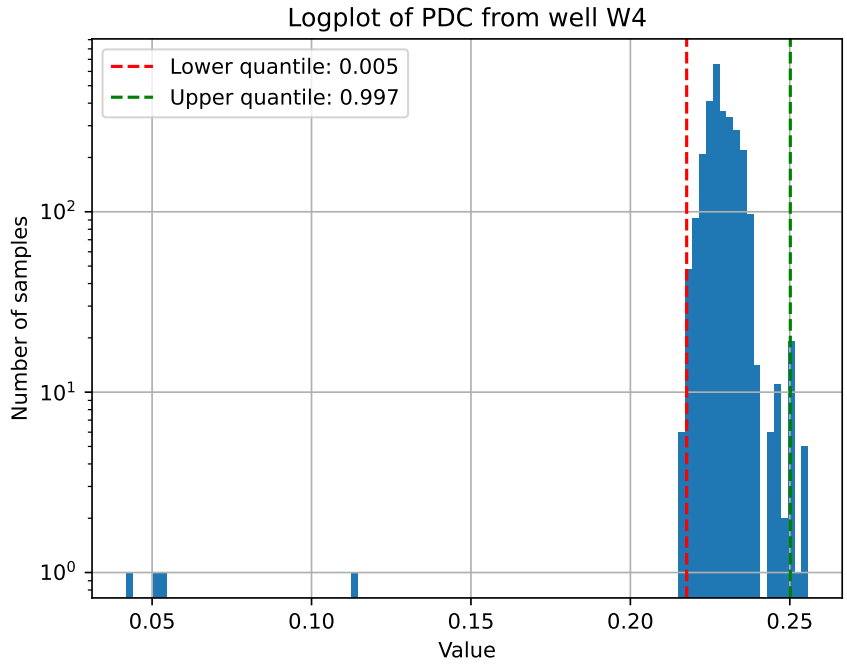


Figure A.3.15: The plot shows a histogram of the downstream choke pressure variable from the training data from well 4 with percentiles used when drawing fake data from a uniform distribution.

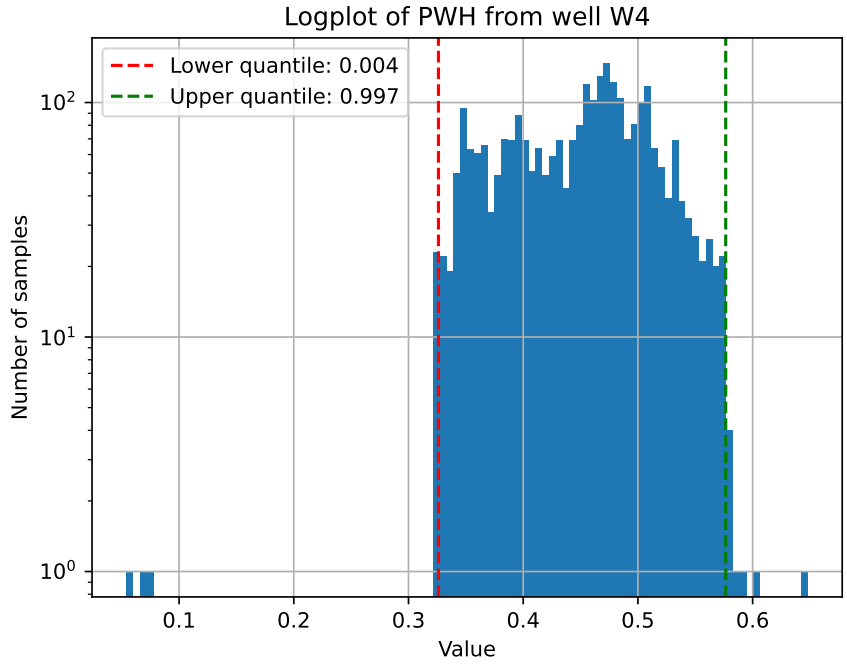


Figure A.3.16: The plot shows a histogram of the wellhead pressure variable from the training data from well 4 with percentiles used when drawing fake data from a uniform distribution.

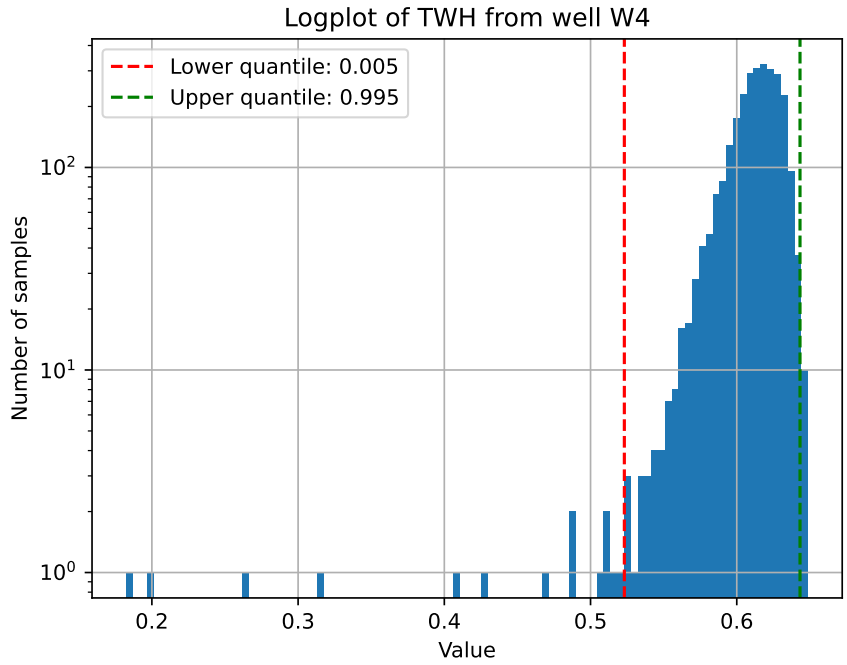


Figure A.3.17: The plot shows a histogram of the temperature wellhead variable from the training data from well 4 with percentiles used when drawing fake data from a uniform distribution.

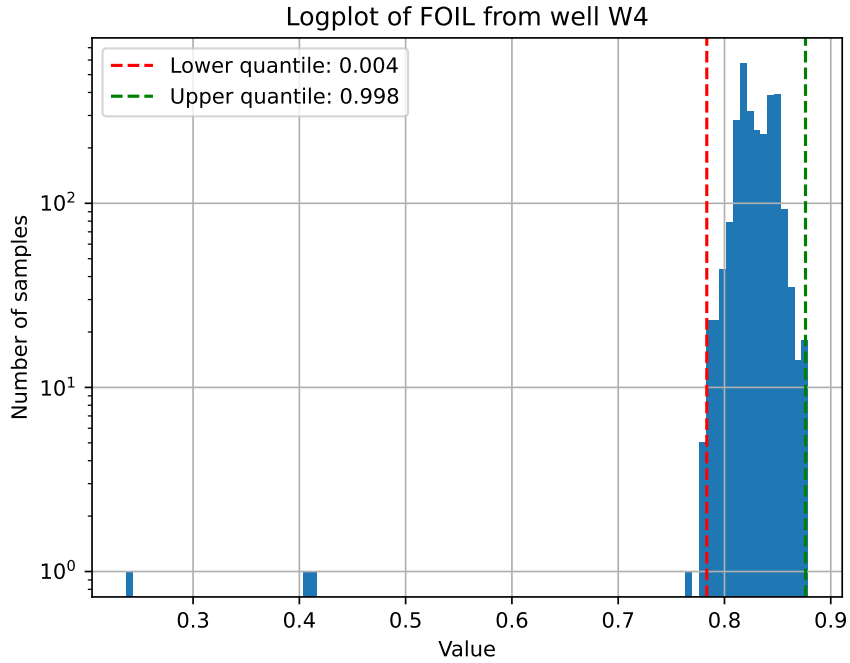


Figure A.3.18: The plot shows a histogram of the fraction of oil variable from the training data from well 4 with percentiles used when drawing fake data from a uniform distribution.

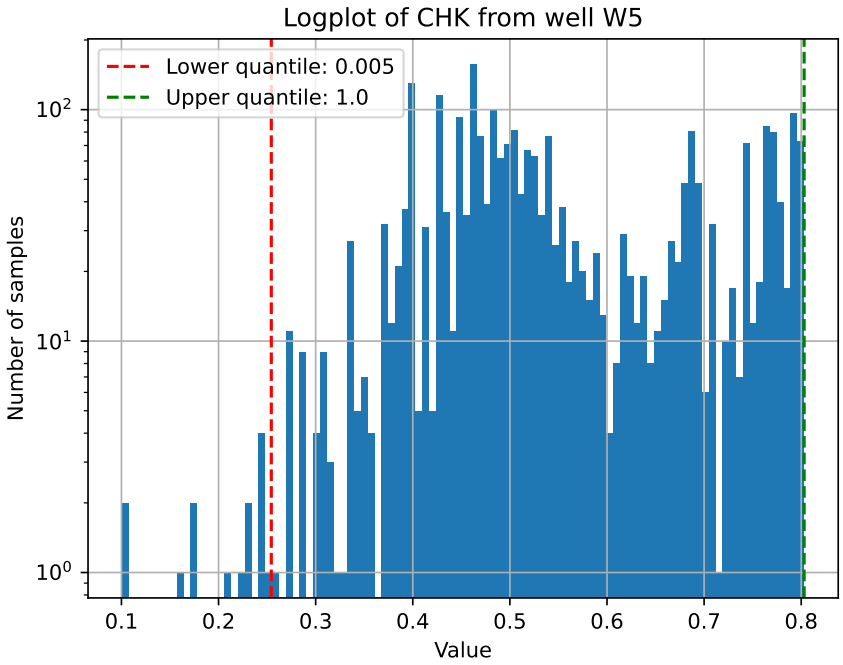


Figure A.3.19: The plot shows a histogram of the choke variable from the training data from well 5 with percentiles used when drawing fake data from a uniform distribution.

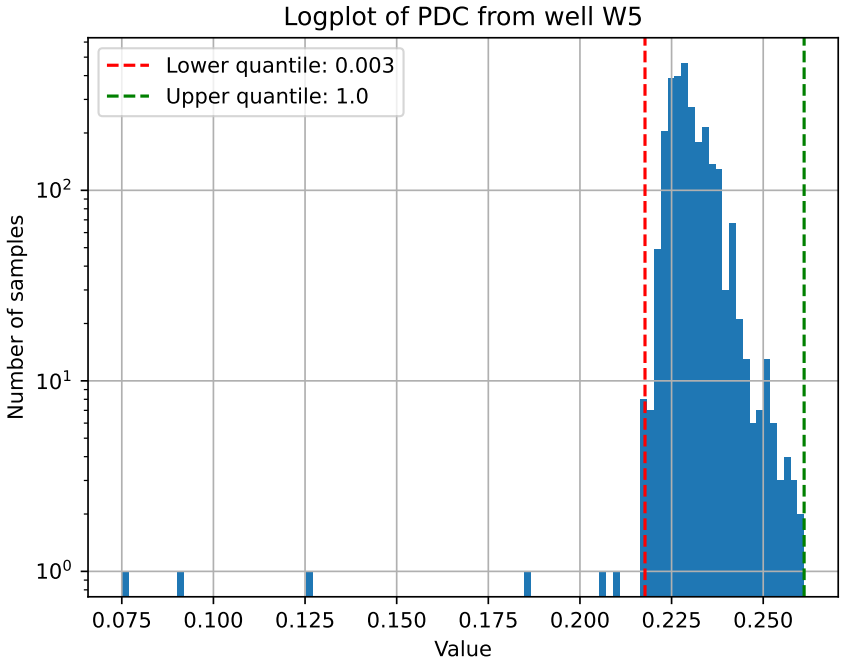


Figure A.3.20: The plot shows a histogram of the downstream choke pressure variable from the training data from well 5 with percentiles used when drawing fake data from a uniform distribution.

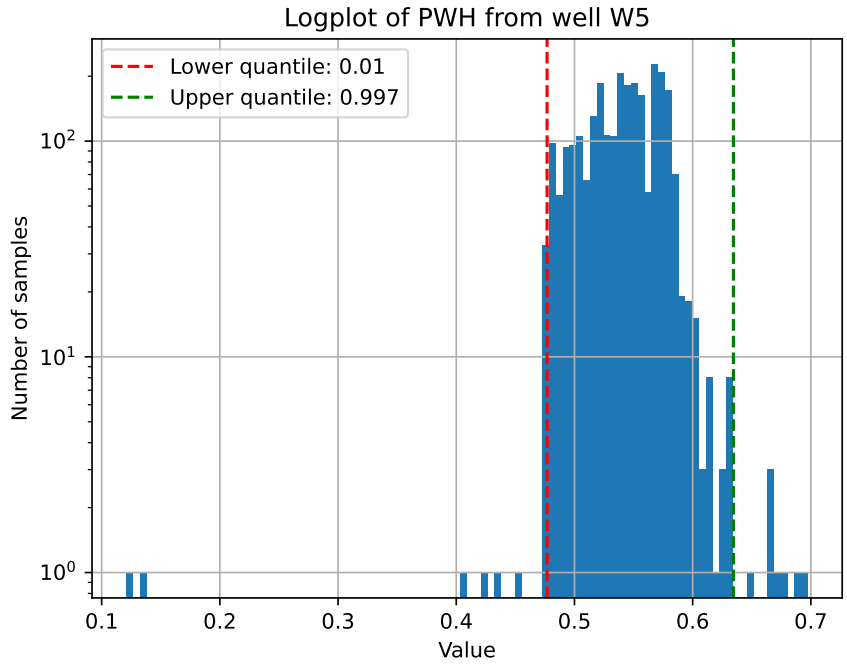


Figure A.3.21: The plot shows a histogram of the wellhead pressure variable from the training data from well 5 with percentiles used when drawing fake data from a uniform distribution.

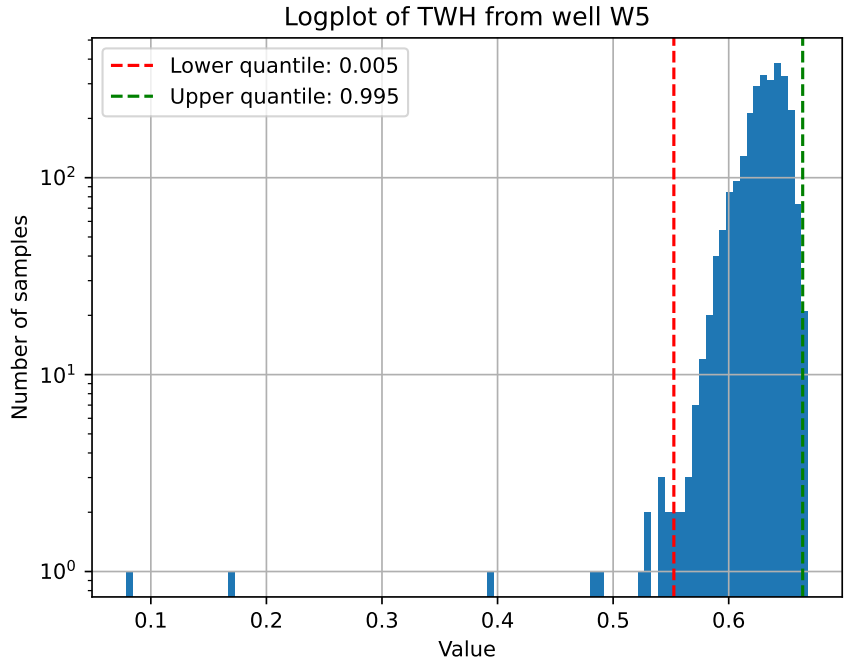


Figure A.3.22: The plot shows a histogram of the temperature wellhead variable from the training data from well 5 with percentiles used when drawing fake data from a uniform distribution.

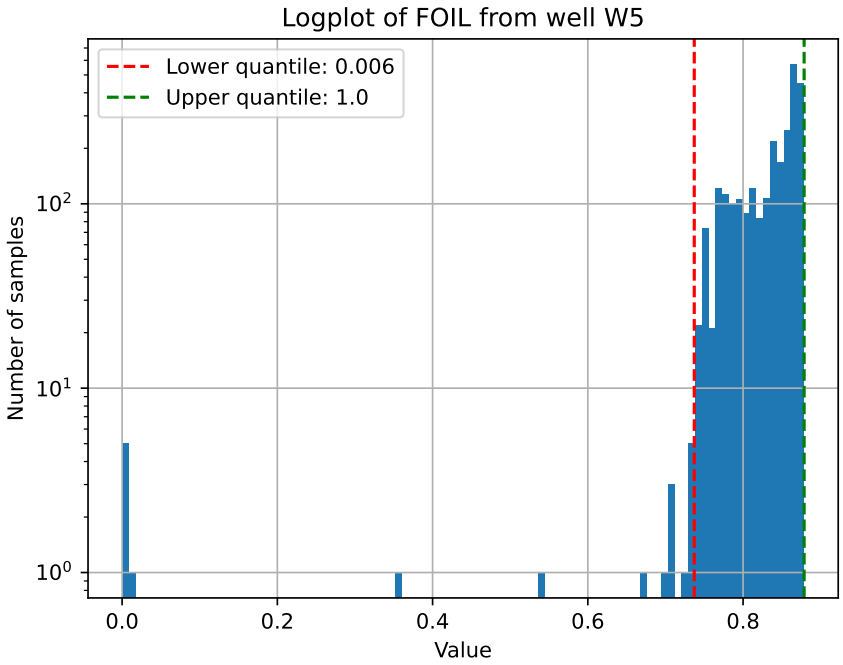


Figure A.3.23: The plot shows a histogram of the fraction of oil variable from the training data from well 5 with percentiles used when drawing fake data from a uniform distribution.

A.4 Flow rates of all wells plotted against time with error in mechanistic flow model

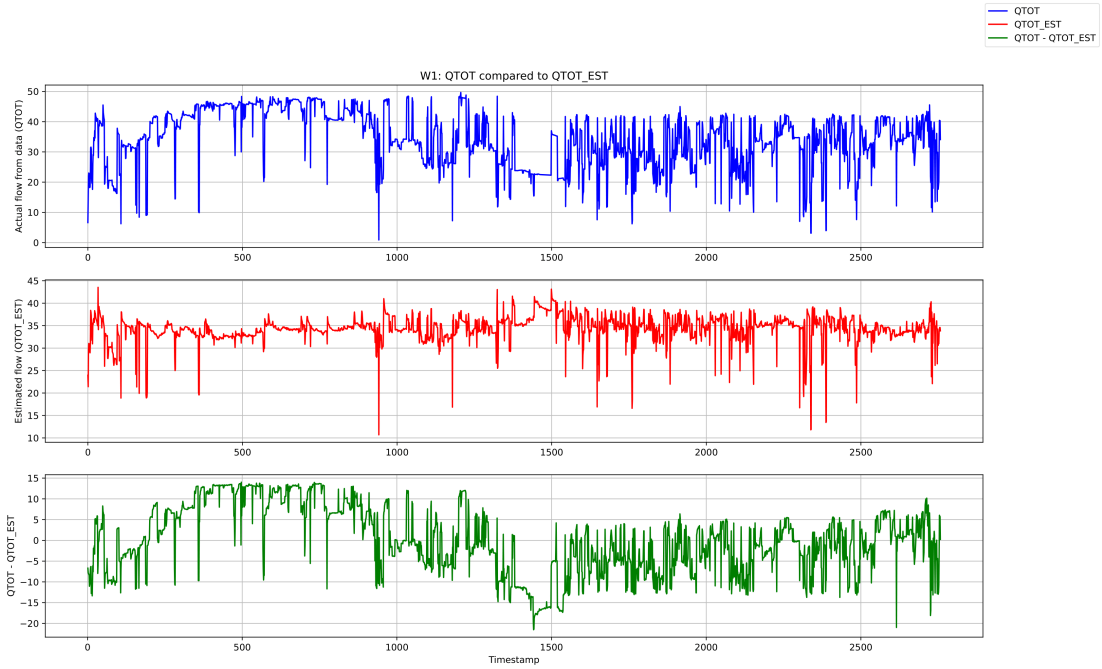


Figure A.4.1: The true total flow rate of well 1 (in blue) compared to the estimated total flow rate using (2.3.1) (in red), and the difference between them (in green).

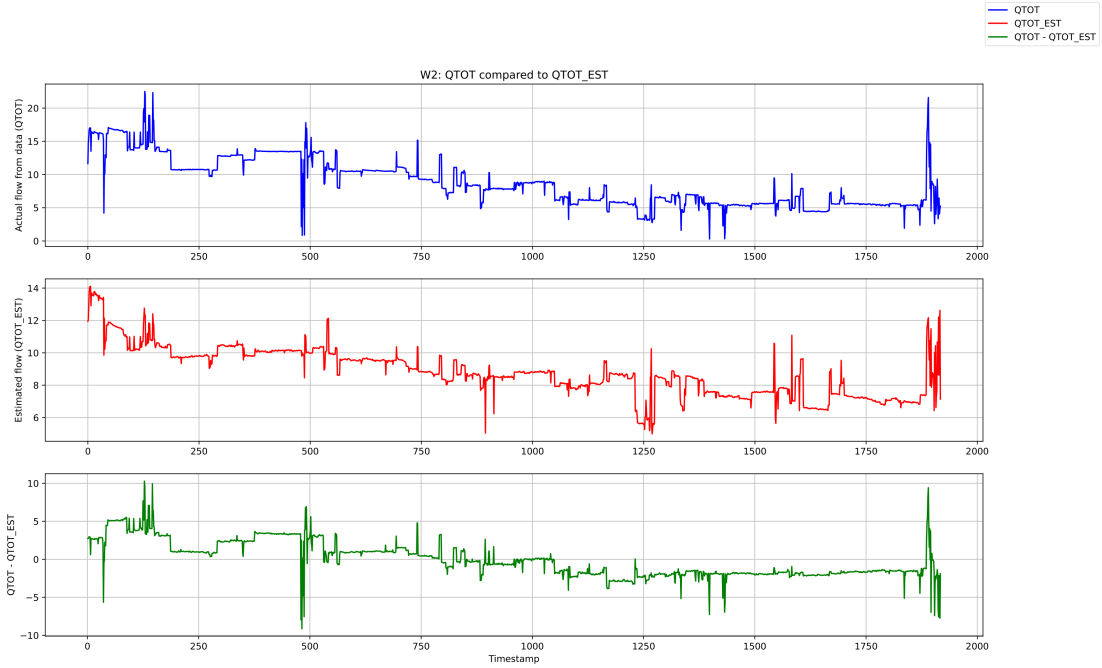


Figure A.4.2: The true total flow rate of well 2 (in blue) compared to the estimated total flow rate using (2.3.1) (in red), and the difference between them (in green).

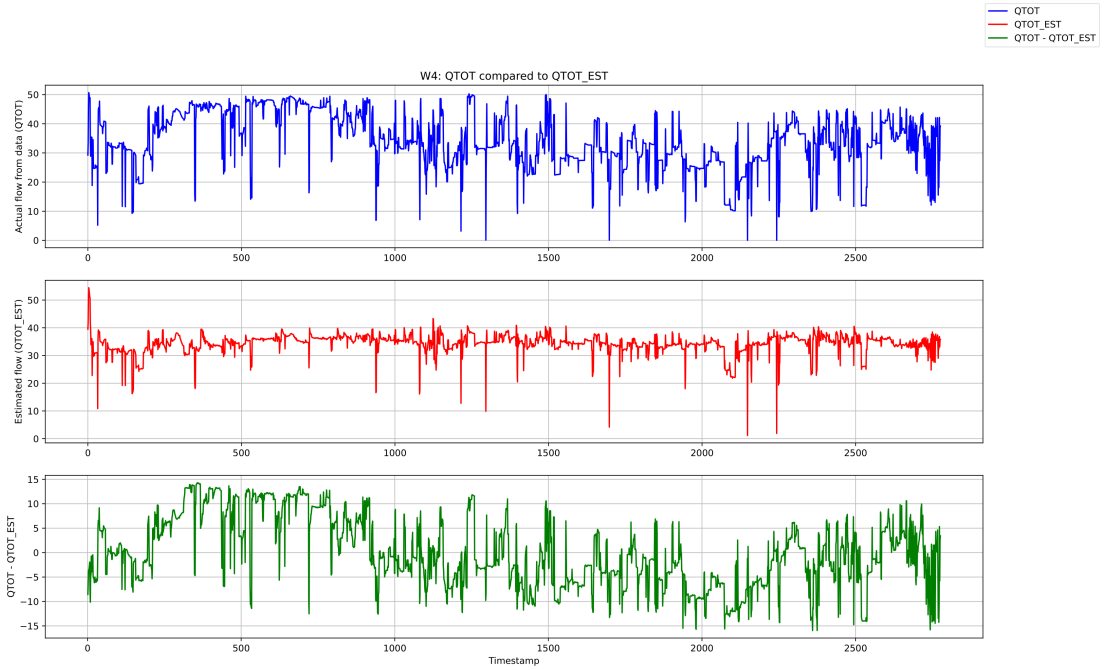


Figure A.4.3: The true total flow rate of well 4 (in blue) compared to the estimated total flow rate using (2.3.1) (in red), and the difference between them (in green).

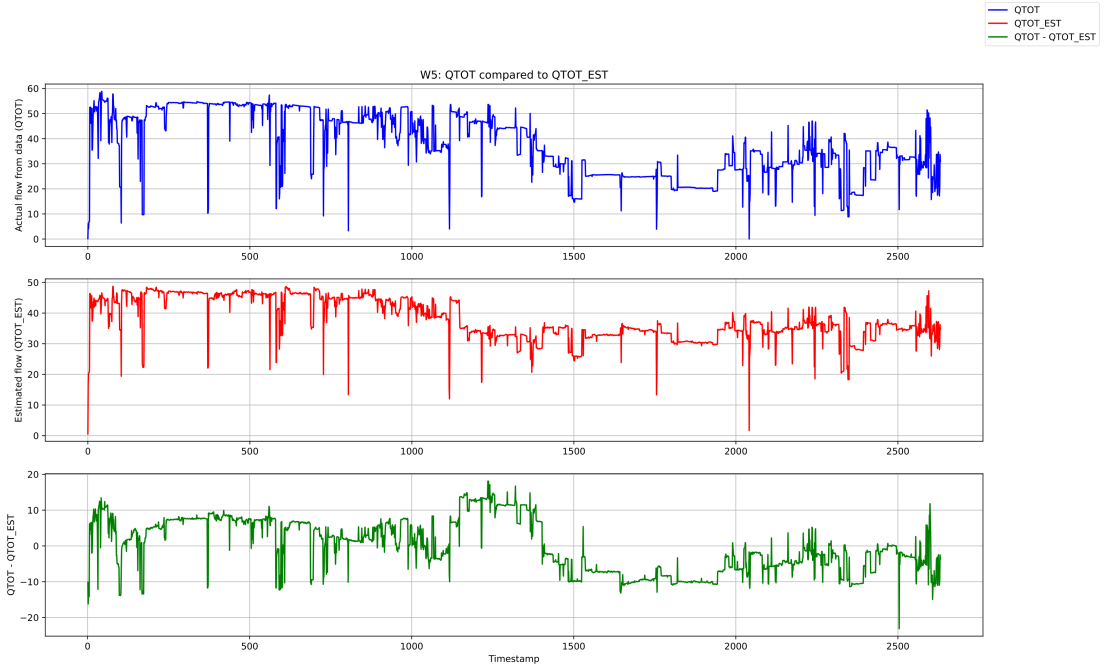


Figure A.4.4: The true total flow rate of well 5 (in blue) compared to the estimated total flow rate using (2.3.1) (in red), and the difference between them (in green).

A.5 Boxplots of all variables including fake data from both uniform and well-specific distributions

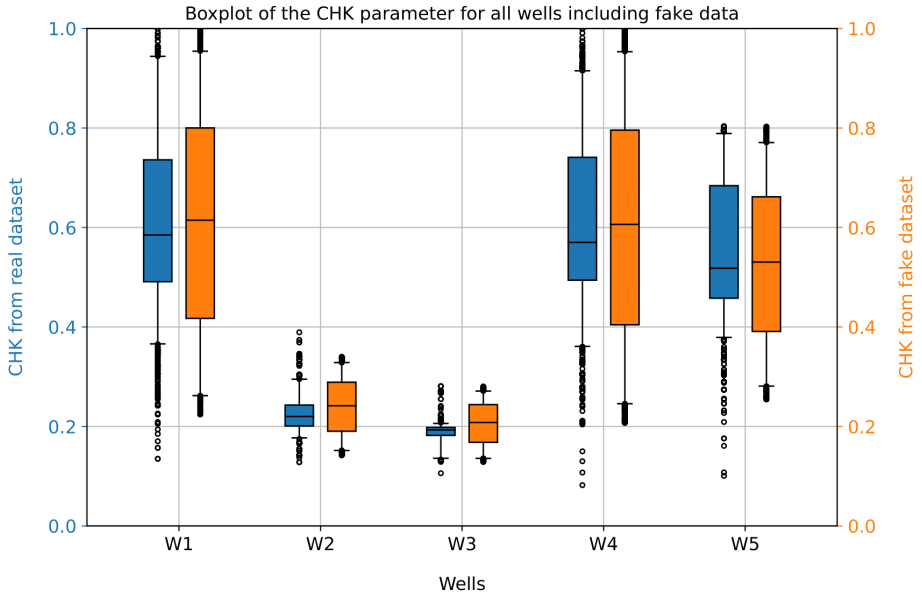


Figure A.5.1: A boxplot of the choke variable drawn from a uniform distribution for all wells compared to the true choke variable for all wells.

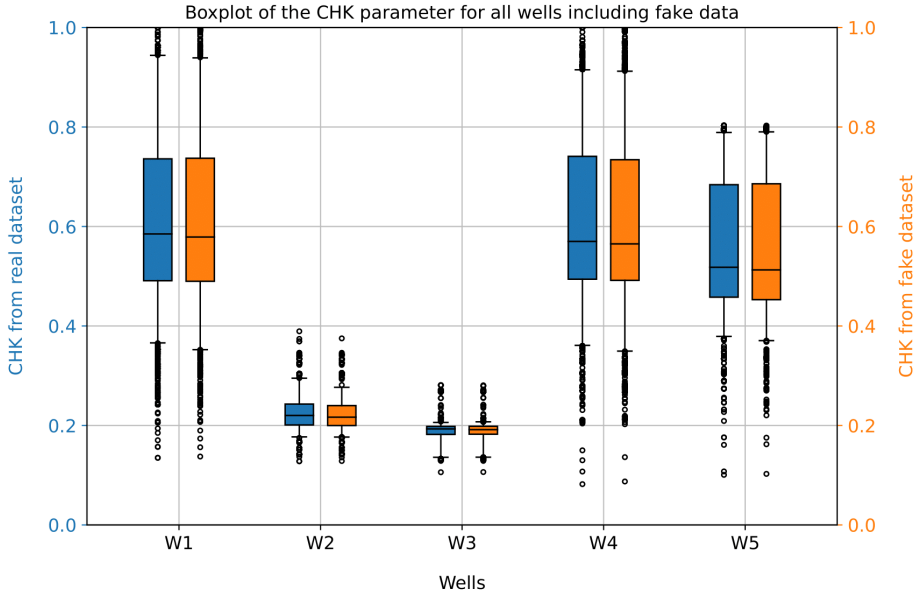


Figure A.5.2: A boxplot of the choke variable drawn from a well-specific distribution for all wells compared to the true choke variable for all wells.

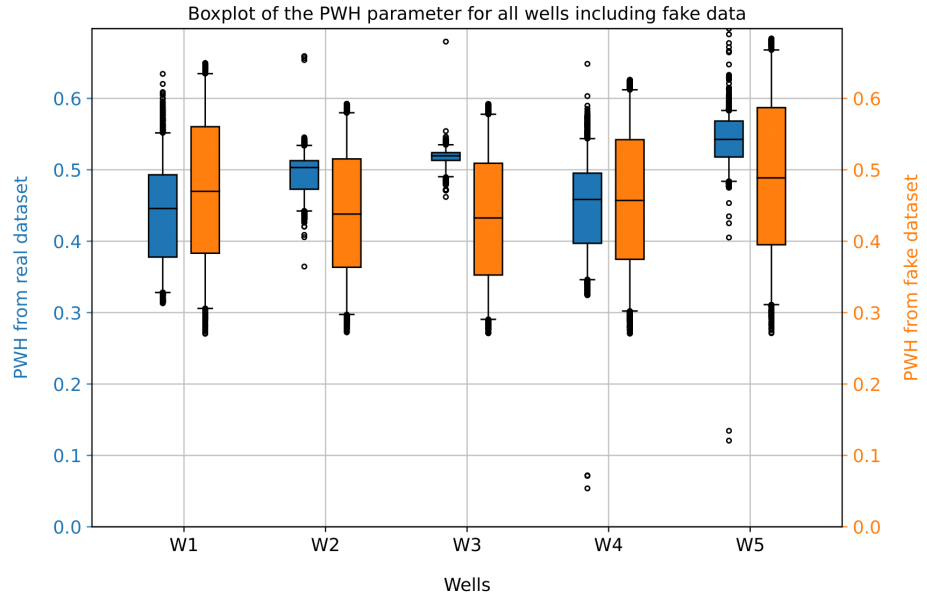


Figure A.5.3: A boxplot of the wellhead pressure variable drawn from a uniform distribution for all wells compared to the true wellhead pressure variable for all wells.

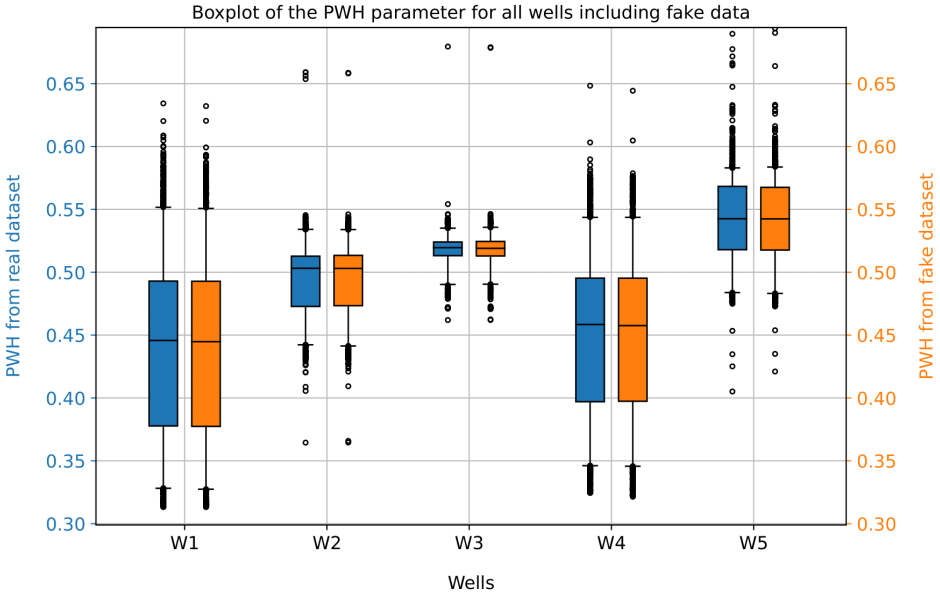


Figure A.5.4: A boxplot of the wellhead pressure variable drawn from a well-specific distribution for all wells compared to the true wellhead pressure variable for all wells.

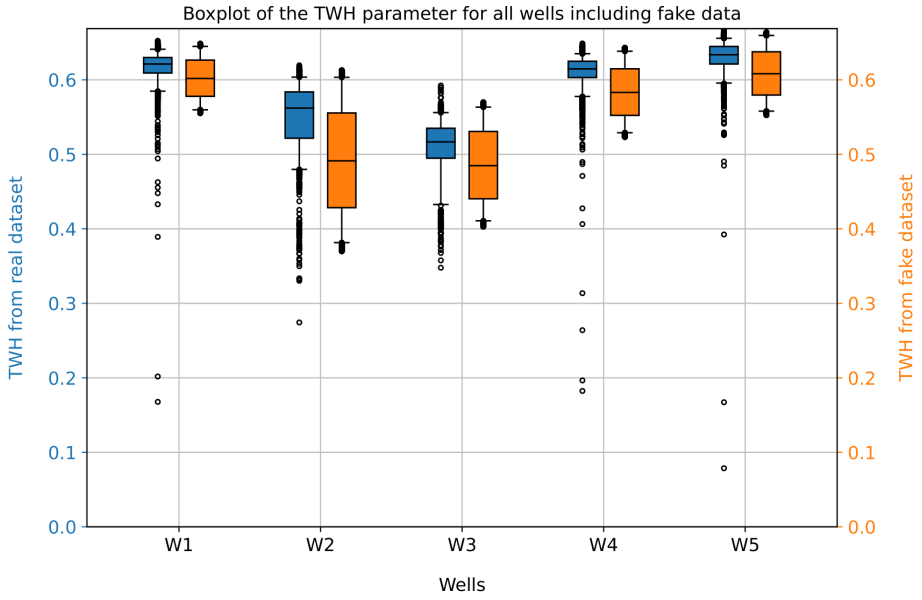


Figure A.5.5: A boxplot of the temperature wellhead variable drawn from a uniform distribution for all wells compared to the true temperature wellhead variable for all wells.

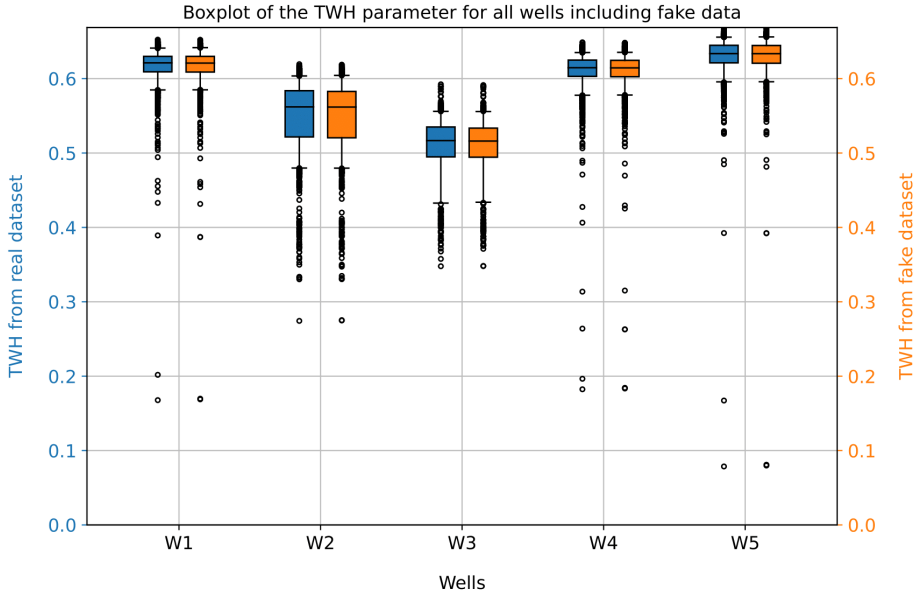


Figure A.5.6: A boxplot of the temperature wellhead variable drawn from a well-specific distribution for all wells compared to the true temperature wellhead variable for all wells.

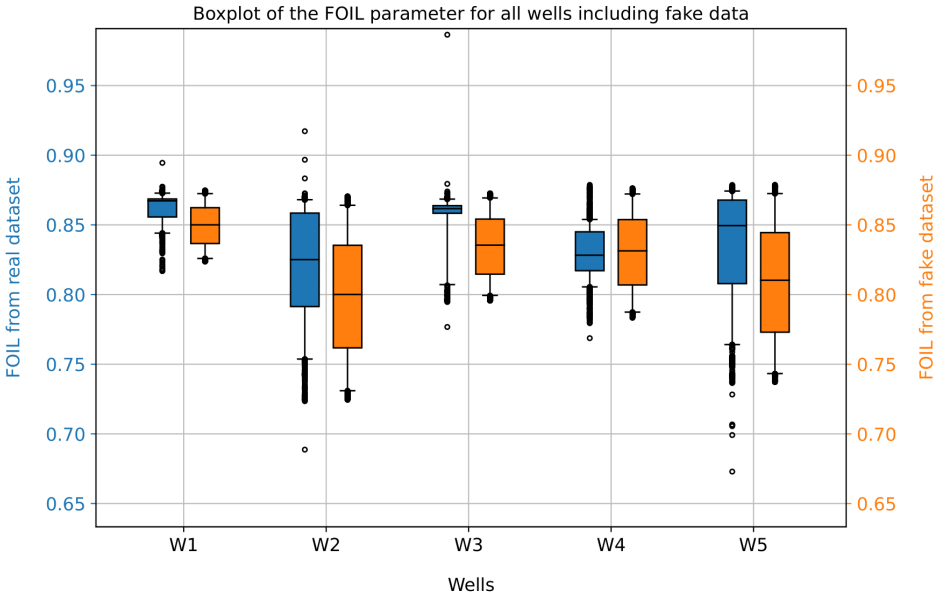


Figure A.5.7: A boxplot of the fraction of oil variable drawn from a uniform distribution for all wells compared to the true fraction of oil variable for all wells.

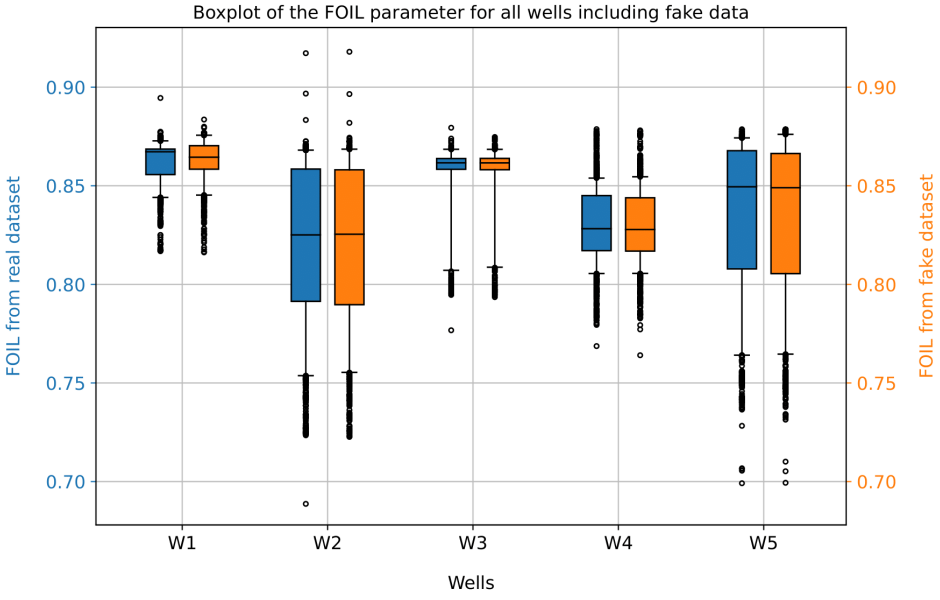


Figure A.5.8: A boxplot of the fraction of oil variable drawn from a well-specific distribution for all wells compared to the true fraction of oil variable for all wells.

A.6 Scatter plots of generated data on top of real data for each well for both uniform and well-specific distributions

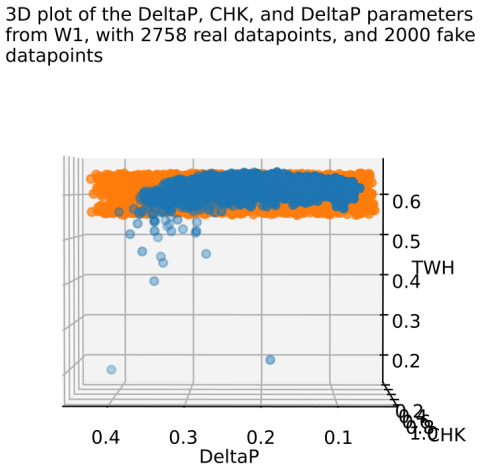


Figure A.6.1: A scatter plot of generated data on top of real data for well 1, where the generated data was drawn from a uniform distribution.

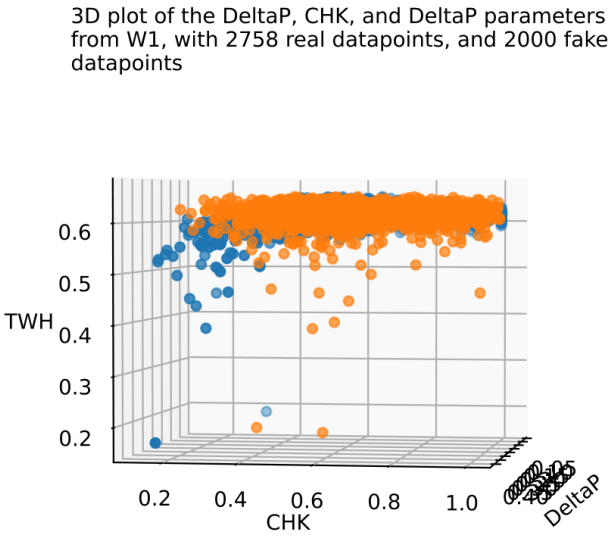


Figure A.6.2: A scatter plot of generated data on top of real data for well 1, where the generated data was drawn from a well-specific distribution.

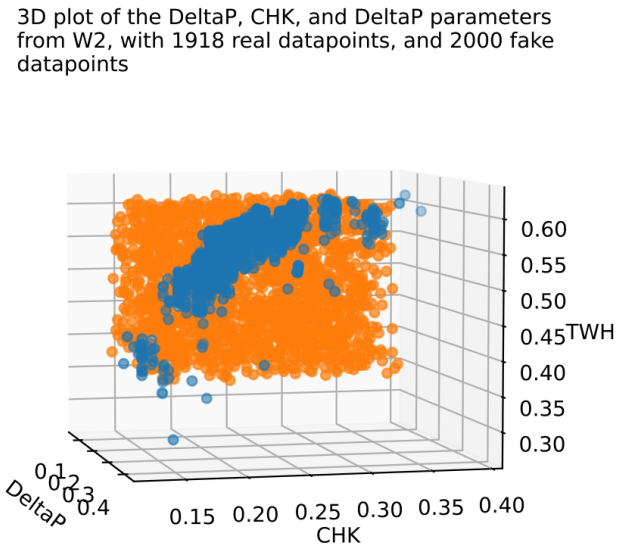


Figure A.6.3: A scatter plot of generated data on top of real data for well 3, where the generated data was drawn from a uniform distribution.

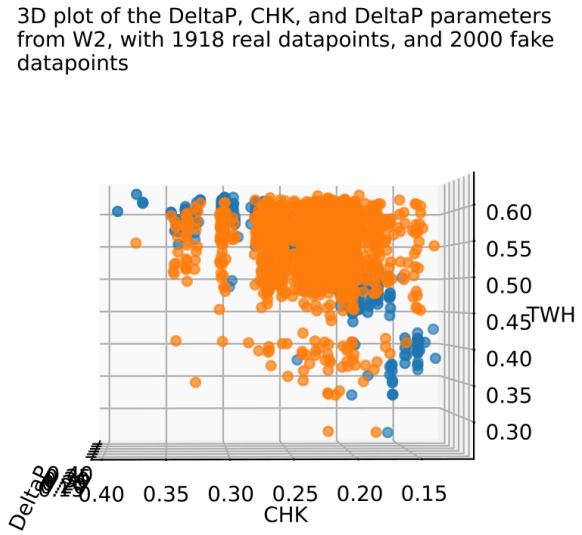


Figure A.6.4: A scatter plot of generated data on top of real data for well 3, where the generated data was drawn from a well-specific distribution.

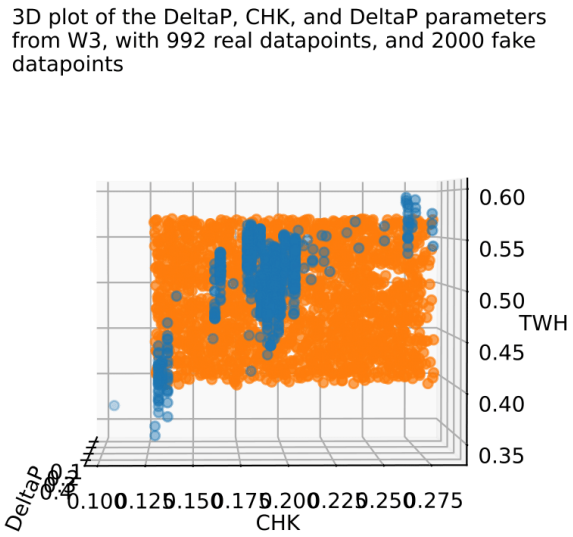


Figure A.6.5: A scatter plot of generated data on top of real data for well 3, where the generated data was drawn from a uniform distribution.

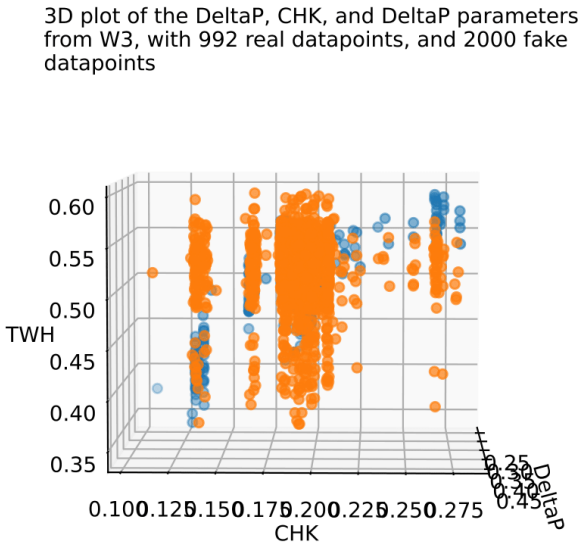


Figure A.6.6: A scatter plot of generated data on top of real data for well 3, where the generated data was drawn from a well-specific distribution.

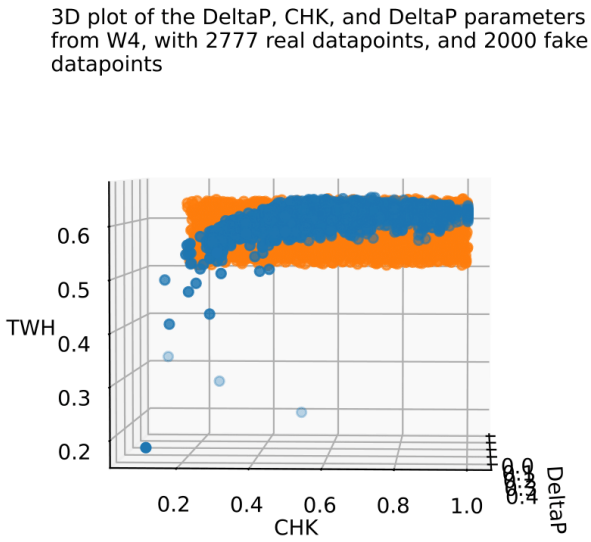


Figure A.6.7: A scatter plot of generated data on top of real data for well 4, where the generated data was drawn from a uniform distribution.

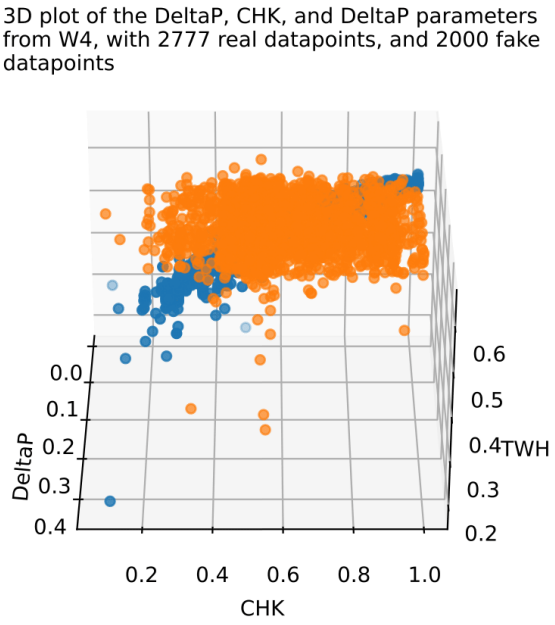


Figure A.6.8: A scatter plot of generated data on top of real data for well 4, where the generated data was drawn from a well-specific distribution.

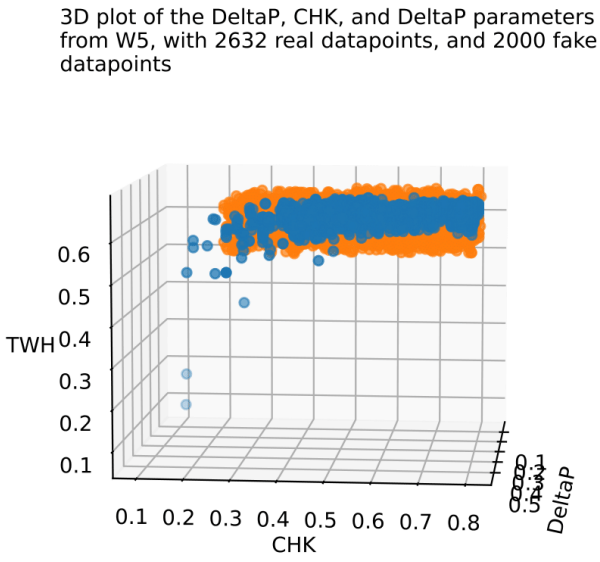


Figure A.6.9: A scatter plot of generated data on top of real data for well 5, where the generated data was drawn from a uniform distribution.

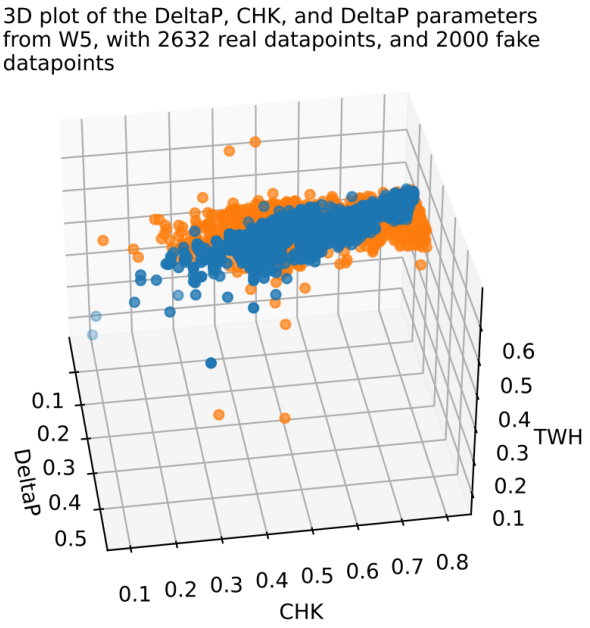


Figure A.6.10: A scatter plot of generated data on top of real data for well 5, where the generated data was drawn from a well-specific distribution.

A.7 Scatter plots of test data an generated fake data

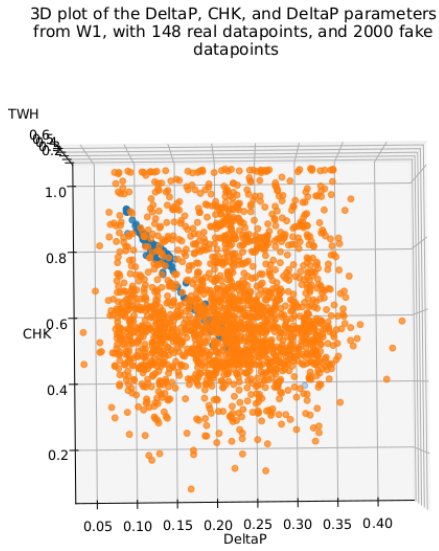


Figure A.7.1: A scatter plot of test data and fake data generated from well-specific distribution for well 1.

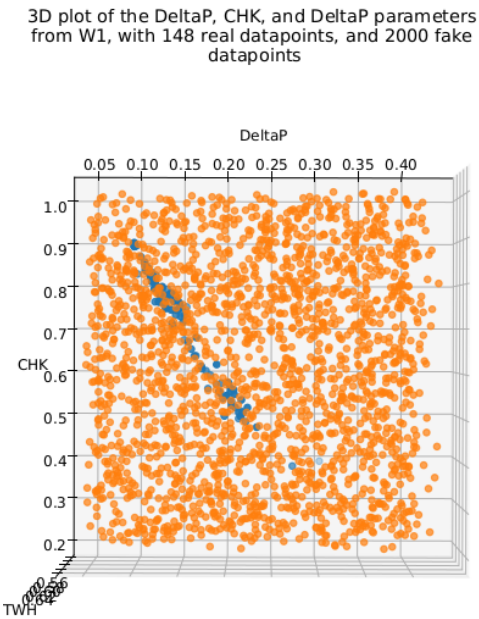


Figure A.7.2: A scatter plot of test data and fake data generated from uniform distribution for well 1.

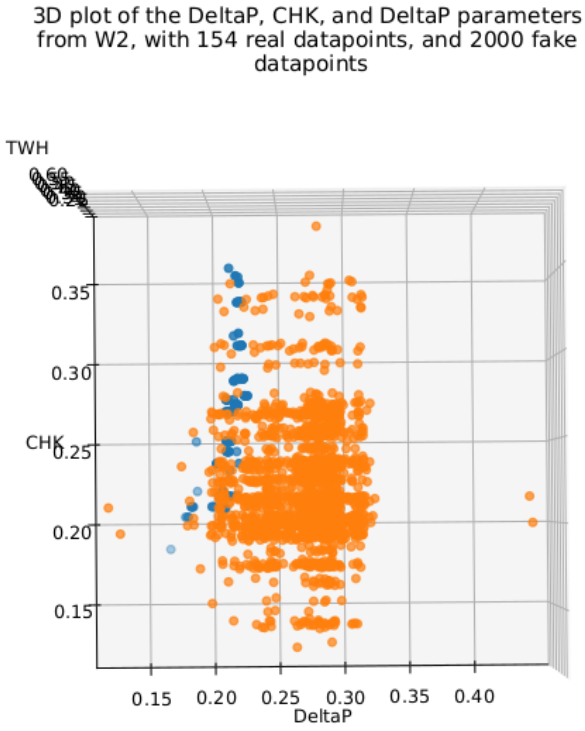


Figure A.7.3: A scatter plot of test data and fake data generated from well-specific distribution for well 2.

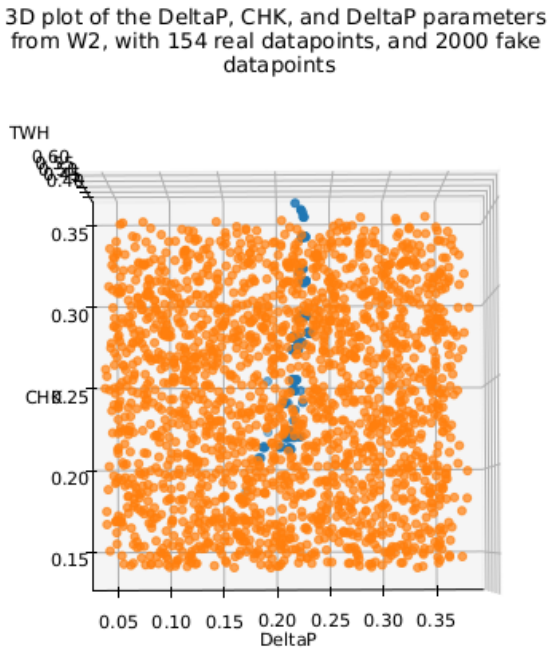


Figure A.7.4: A scatter plot of test data and fake data generated from uniform distribution for well 2.

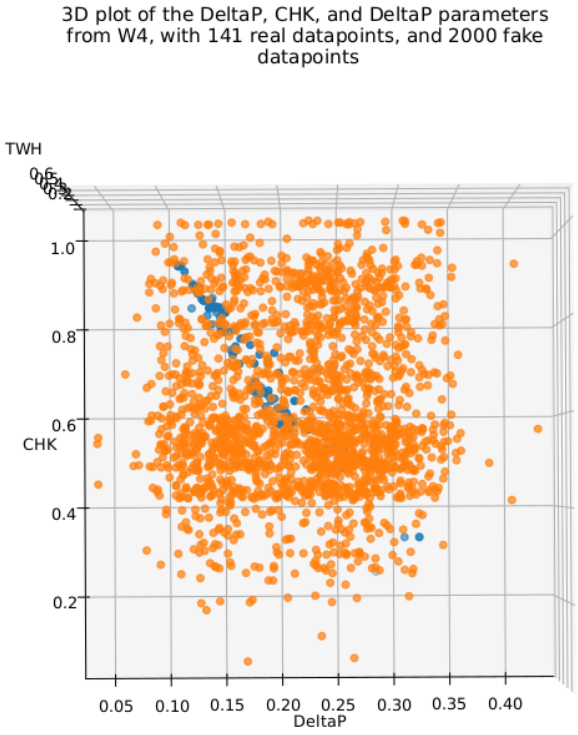


Figure A.7.5: A scatter plot of test data and fake data generated from well-specific distribution for well 4.

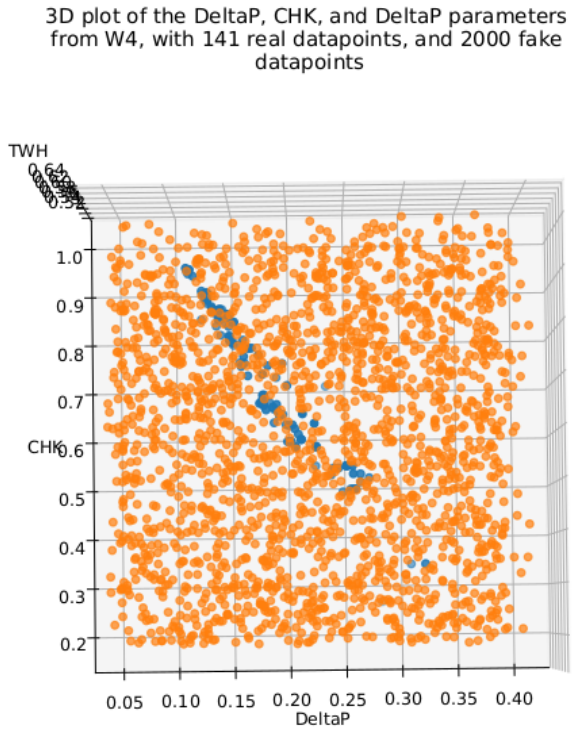
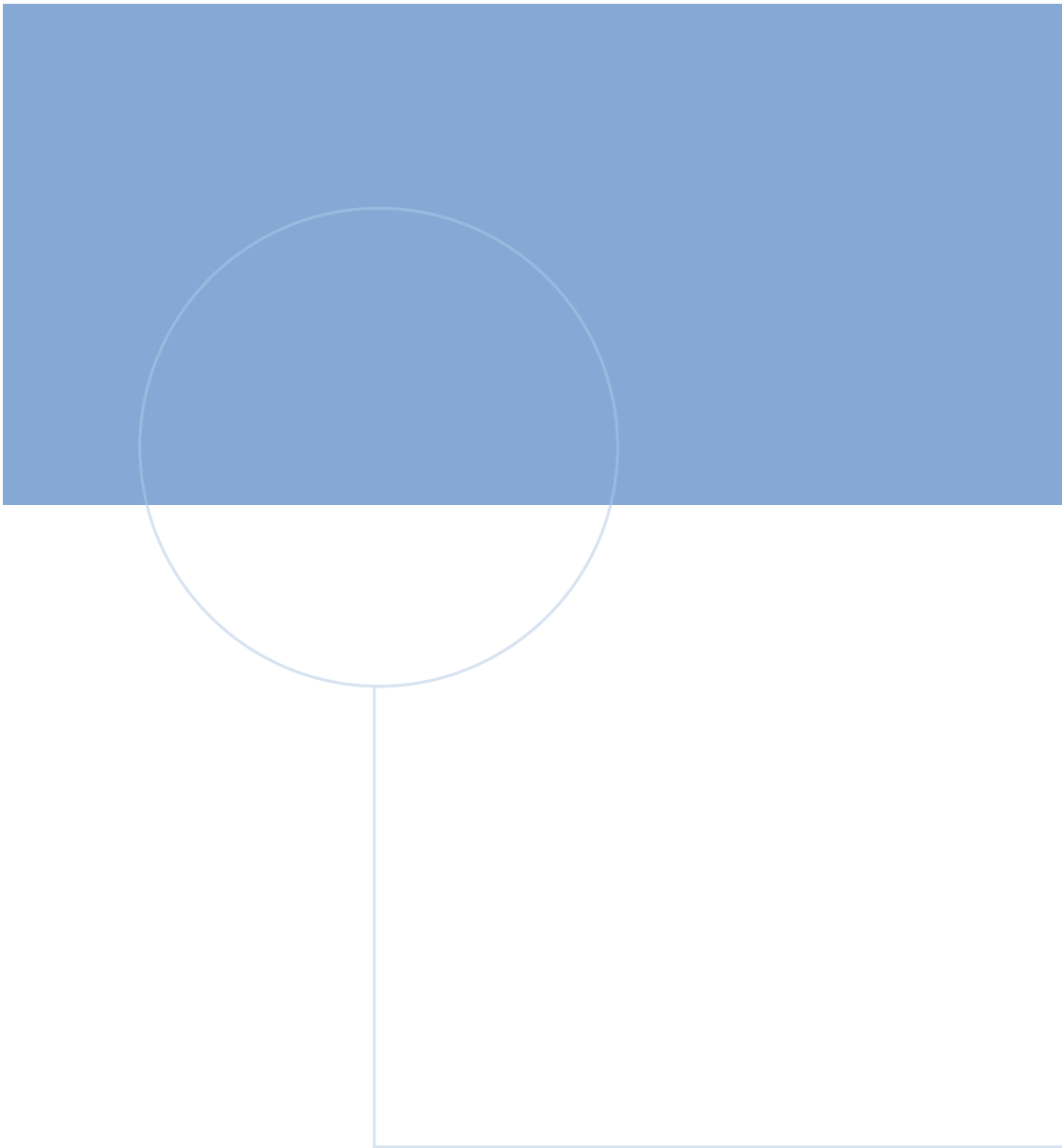


Figure A.7.6: A scatter plot of test data and fake data generated from uniform distribution for well 4.



NTNU

Kunnskap for en bedre verden