Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

D Aqnan Marusaha Matthew

Proxy Modeling for CO_2-EOR Design Study : Water Alternating Gas and Storage

Master's thesis in Petroleum Engineering
Supervisor: Ashkan Jahanbani Ghahfarokhi
Co-supervisor: Alv-Arne Grimstad
June 2021

NTNU
Norwegian University of
Science and Technology

D Aqnan Marusaha Matthew

Proxy Modeling for CO_2-EOR Design Study : Water Alternating Gas and Storage

Master's thesis in Petroleum Engineering
Supervisor: Ashkan Jahanbani Ghahfarokhi
Co-supervisor: Alv-Arne Grimstad
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

This thesis itself is His Blessing. Hence, I dedicate this to my beloved family, friends, and those who never fail to put a grin on my face.

Abstract

Optimization study is an exhaustive study that requires many runs and vast amounts of space to find and store the results. Nevertheless, it is one of the most widely performed studies in petroleum engineering studies, such as production optimization and EOR assessment study. When talking about EOR, CO_2 is one of the most common methods employed. To assess the feasibility of the CO_2-EOR project, a reservoir design study must be conducted, where optimization will be performed. Some studies show that employing a proxy model to do this task saves a lot of space and time needed. However, no studies explicitly stated how this is done and what problems need to be tackled to build a proxy model.

In this study, we developed proxy models to solve a multi-objective optimization problem using NSGA-II on the reservoir models we have. The study was performed for CO_2-WAG reservoir assessment, where gas injection rate, water injection rate and half-cycle length are assessed to maximize the oil recovery and CO_2 stored in the reservoir. Two reservoir models were studied. One represents a simple geological model (Egg Model), while the other represents a complex model (Gullfaks Model). In this study, we described in details the process to build both proxy models from scratch. Following that, we found out that a higher amount of sampling is needed, and more proxy segmentations are needed to build a robust proxy model for a complex reservoir model. In alignment with that, we found that to reach the maximum oil recovery on CO_2-WAG, we need to have a maximum gas injection rate with a minimum water injection rate. However, this configuration will result in the reduction of the total CO_2 stored in the reservoir. All proxies have average error less than 2% and is concluded to be robust based on the blind test results.

Keywords: Proxy Model, CO_2-WAG, NSGA-II, Optimization Study

Preface

This thesis is written to fulfil the partial requirements for the MSc degree in Reservoir Engineering and Petrophysics at Department of Geoscience and Petroleum - Norwegian University of Science and Technology (NTNU). This research is a part of CEORS Gemini-Center (CO_2 Enhanced Oil Recovery and Storage), a strategic cooperation between NTNU and SINTEF (https://www.ntnu.edu/igp/ceors). This thesis is the continuation of the previous work during the previous semester on the specialization project (Matthew, 2020). This study started from June 2020, with topic proposed by my supervisor. Although six months are the allocated time to finish this thesis, a year has passed since I started studying about proxy model.

I would like to express my gratitude to my supervisor, Professor Ashkan Jahanbani Ghahfarokhi, to introduce the topic, sharing and giving excellent guidance from last year until the end of this study. I would like to thank Alv-Arne Grimstad as my co-supervisor for meaningful feedback on my study. In addition to that, my appreciation also goes to Menad Nait Amar and Cuthbert Shang Wui Ng for all meaningful feedback, suggestions and support throughout the study. I would like to thank Professor Curtis Hays Whitson for guidance in PVT modeling and Whitson AS for the Advanced PVT Course that helped the PVT modeling for this study.

For those who just started a study related to proxy building, especially its application for optimization study, I hope that this can be a good starting point for the research. And for other readers, I hope this can be a good insight for your knowledge.

Trondheim, June 2021

D Aqnan Marusaha Matthew

Table of Contents

List of Tables

List of Figures

Abbreviations

AE	=	Absolute Error
ANN	=	Artificial Neural Network
API	=	American Petroleum Institute
APRE	=	Average Percentage Relative Error
CCE	=	Constant Composition Expansion
CCUS	=	Carbon Capture, Utilization and Storage
CDF	=	Cumulative Distribution Function
CO_2	=	Carbon Dioxide
cp	=	Centipoise
DL	=	Differential Liberation
DNN	=	Deep Neural Network
EOR	=	Enhanced Oil Recovery
FCO_2PR	=	Field CO2 Production Rate
FCO_2PT	=	Field CO2 Production Total
FGIR	=	Field Gas Injection Rate
FOPR	=	Field Oil Production Rate
FOPT	=	Field Oil Production Total
ft	=	feet
GUI	=	Graphical User Interface
HC	=	Half-cycle
IOR	=	Improved Oil Recovery
K	=	Kelvin
KG	=	Kriging
KPI	=	Key Performance Indicator
LHS	=	Latin Hypercube Sampling
md	=	milidarcy
MMP	=	Minimum Miscibility Pressure
MSE	=	Mean Squared Error
NSDE-R	=	Non-dominated Sorting Differential Evolution based on Reference points
NSGA	=	Non-dominated Sorting Genetic Algorithm
NSGA-II	=	Non-dominated Sorting Genetic Algorithm-II
NSGA-III	=	Non-dominated Sorting Genetic Algorithm-III
NTG	=	Net-to-gross
PAES	=	Pareto Archived Evolution Strategy
P_{avg}	=	Average Pressure
PCA	=	Principal Component Analysis
PR	=	Polynomial Regression
P_{sat}	=	Saturation Pressure
PV	=	Pore Volume

PVT	=	Pressure, Volume, Temperature
QC	=	Quality Control
q_g	=	Gas injection rate
q_w	=	Water injection rate
RF	=	Recovery Factor
RMSE	=	Root Mean Square Error
RNN	=	Recurrent Neural Network
SAGD	=	Steam Assisted Gravity Drainage
SPEA	=	Strength Pareto Evolutionary Algorithm
SPEA-2	=	Strength Pareto Evolutionary Algorithm 2
SPM	=	Smart Proxy Model
SRM	=	Surrogate Reservoir Model
STOIP	=	Stock Tank Oil in Place
STOOIP	=	Stock Tank Original Oil in Place
T_{res}	=	Reservoir Temperature
TSP	=	Thin-plate Spilines
WAG	=	Water Alternating Gas

Chapter 1

Introduction

1.1 Background

Most studies in petroleum industry involve optimization. The most common objective function is to improve oil production. For example, to find the best location to add additional wells, waterflood injection design, and EOR injection design. This is performed more intensively nowadays as oil production decline occurs in existing fields, yet the energy demand constantly rises. Discoveries of new oil fields do not follow this problem. To tackle the problem, more countries have started to study the feasibility of EOR applications in their mature fields, such as CO_2-EOR.

The study can be complex and hard to solve when discussing the optimization study for CO_2-EOR. Designing preferable injection rates and injection conditions is one of the reservoir engineering design techniques for CO_2 EOR. Deciding whether it is profitable is such an exhaustive process, where one of the components to be done is to optimize the CO_2 injection EOR based on selected optimization parameters. This study usually is performed using a reservoir simulator first before moving to the small coverage area (pilot test), followed by field-scale application. The current limitation of this study is its excessive time and space needed to solve the optimization problem and store the result, especially for the complex reservoir model.

For the past ten years, the idea to tackle this problem has been by employing a proxy model, commonly called Smart Proxy Model (SPM) or Surrogate Reservoir Model (SRM). The previous study (Matthew, 2020) shows that the proxy model is able to reduce the total running time from 4.5 minutes to less than 10 seconds. The proxy model learned the decline and plateau oil rate behaviour of our reservoir model (Egg Model). This proxy was used to perform an optimization study to optimize total oil produced with gas injection rate, start time of injection, total slug size injected and composition of the injected gas as the parameters to be studied in the CO_2 flooding project.

Most studies related to proxy modelling, such as CO_2-related proxy model that performed by Gholami (2014), Amini (2015), Nait Amar et al. (2018), and Nait Amar et al. (2020) show that proxy model can be a substitute for our reservoir model and can be used for optimization study. In our previous study (Matthew, 2020), we constructed a simple guideline for developing a proxy model from scratch and applied the workflow to the CO_2 flooding optimization study. As the previous study continues, the current idea is to develop proxies for a simple model and a complex field-like model to solve optimization study. From both of them, we can see the complexity that might be encountered when building a proxy model for a complex model and tackle the problems encountered while working with it.

1.2 Objective

The primary research objective is developing a new strategy that allows a significant reduction of runtime and storage associated with the commercial simulators without sacrificing accuracy. This study will be performed on CO_2-WAG as one of the most common Enhanced Oil Recovery (EOR) methods. The proxy model will be made as our reservoir model substitute, where we will maximize total oil produced and CO_2 stored as the objective function of our optimization problem.

This study will be performed on two geological models, where one acts as a simple model while the other represents the complexity we usually have in a real field model. Several points that are studied in this research are:

1. Formulating a multi-objective optimization problem.
2. Building a proxy model for a simple and complex reservoir model that involves sampling using experiments, proxy building, and proxy robustness assessment.
3. Analyzing the problem and complexity encountered when building a proxy model in a complex reservoir model by comparing the results with the simple reservoir proxy model.
4. Solving the optimization problem with the generated proxy, both simple and complex reservoir model using an optimization algorithm.

1.3 Structure of the Report

The thesis is arranged in seven chapters. Chapter one introduces the background and the research objective. Chapter two summarizes the basic theory of CO_2-EOR, proxy modeling and optimization. Chapter three describes the methodology and the problem, followed with the details of the models used in the study.

Chapter four focuses on the details how proxy model is built to solve the optimization problem. The fifth chapter describes the details of how each proxy model is constructed for each reservoir model we have. Chapter six starts with the comparative study between each proxy models, evaluations, improvements, and the truth found out during building a proxy model. Finally, conclusion are presented in chapter seven.

Chapter 2

Literature Review

A complete review of theories and concepts used are explained in this chapter.

2.1 CO_2 Enhanced Oil Recovery

Starting 1901, when Spindletop discovered, an oil boom was started, making the oil consumption economically feasible. Throughout the decades, thousands of oil fields are discovered and extracted to fulfil the energy needs. Almost all known major fields in the world are discovered by now. Even though still producing, most fields already passed their peaks and now decline over time.

Primary recovery might be not economical after several years of field production. Improved Oil Recovery (IOR) then will be employed to keep the production surpass the economic limit. With the exponential growth in technology and knowledge in the industry, Enhanced Oil Recovery (EOR) now soars as an option. Tabulated in **Table 2.1**, EOR types are categorized as thermal and nonthermal (chemical, miscible, and other EOR types).

Table 2.1: List of EOR methods.

Group	EOR Methods[1]	Principle[2]
Thermal	SAGD, Cyclic steam injection, hot water flood, steam flooding	Sweep and displacement efficiency improvement
Chemical	Polymer, micellar-polymer, emulsion, alkaline, surfactant,	Sweep or displacement efficiency improvement
Miscible	CO_2 flooding, water alternating gas (WAG) N_2 flooding, vaporizing gas drive	Displacement efficiency improvement

[1]Satter and Iqbal (2016)

[2]Carcoana (1992)

When we try to observe EOR projects from 1971-2017, as shown in **Figure 2.1**, we can see an increase of interest in CO_2 as an EOR method to be applied. This method surpassed thermal EOR from 2006 as the most employed EOR method. The increase in other gas injection methods also can be seen, starting 2001. Chemical, as one of most studied EOR methods in the eighties, declines rapidly after CO_2 started to be applied in some fields in the US.

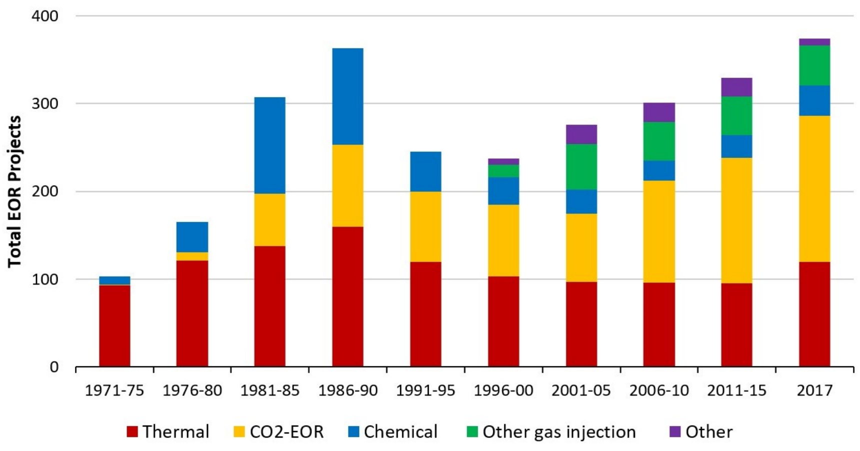

Figure 2.1: Summary of EOR projects globally (IEA, 2017).

The surge of greenhouse gas emission becomes a vital player of this method. Several policies were made to control the emission, reflecting the prediction of global warming scenarios. Carbon Capture, Utilization and Storage (CCUS), one of the main strategies for controlling gas emissions, noted that CO_2-EOR is more lucrative than direct storing. The benefits are tremendous, where the company can produce more hydrocarbon and reduce the CO_2 tax. CO_2 storage might not economical unless the CO_2 tax surpasses the injection cost. Again, this depends on the field location, regulations and CO_2 availability.

Other than that, the interaction between CO_2 with hydrocarbon also becomes the main reason for applying this method. As stated before, CO_2-EOR mainly affects the displacement efficiency of reservoir. CO_2 become miscible with reservoir hydrocarbon at lower pressure compared to other gas injection methods such as nitrogen, methane and methane-ethane mixture (Hawthorne et al., 2017). Richer gas injection may mix at pressure lower than CO_2, but its availability is lower than CO_2.

When mixed, CO_2 will reduce oil viscosity. As mobility ratio is a function of viscosity, this reduction decreases the mobility ratio, resulting in better displacement efficiency. The mixing process itself will increase oil volume, so-called oil swelling and pushes more oil to the producer. Other than that, as water is present in the system, CO_2 will also mix with it. Water expansion will happen. Hence, the gravity segregation effect will be reduced due to smaller density differences (Gholami, 2014). Carbonic acid formed due to water and CO_2 interaction will dissolve carbonates. It will increase reservoir permeability and injectivity.

Miscibility condition needs to be focused in the CO_2-EOR study, as it differs based on the system's pressure. Illustrated in **Figure 2.2**, we can see three stages of mixing. It is worth knowing that miscibility develops slowly for every pressure increase. It is not a "switch" where it only mixes when the minimum miscibility pressure (MMP) was reached. Higher pressure enables CO_2 to mix with heavier hydrocarbon to reach the first contact miscibility condition, but it requires higher pressure, even way higher than our reservoir pressure.

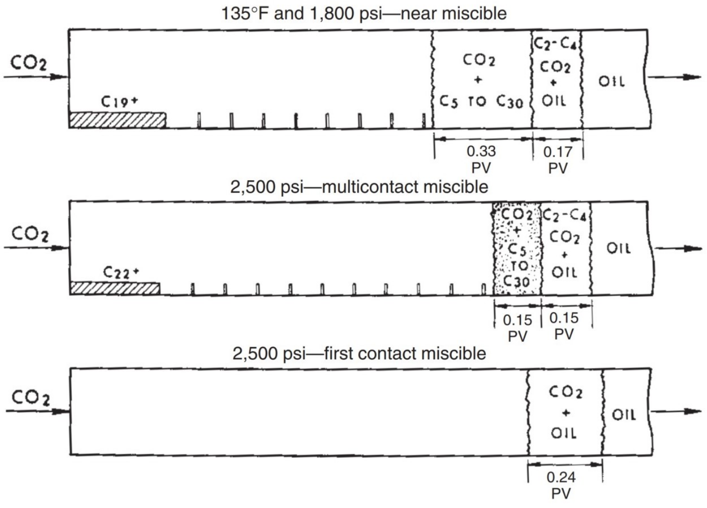

Figure 2.2: CO_2 displacement at miscible and near-miscible conditions (Whitson et al., 2000).

CO_2-EOR can be employed in many injection designs. The most well-known injection designs are flooding, Water Alternating Gas (CO_2-WAG), and huff-and-puff. Most designs are performed until they reach the predetermined volume of total CO_2 injected. One of the methods, huff-and-puff, is performed using the same well to produce and inject the gas. It involves injection, soak and production period as one cycle. Here, the other two injection designs will be described deeper based on Verma (2015).

2.1.1 CO_2 Flooding

This design, commonly called continuous CO_2 injection, requires CO_2 throughout the injection period. Pure CO_2 is usually preferred as the injection fluid but may be mixed with other gas as impurities, such as nitrogen and methane, although this may increase the MMP. This design is often applied to reservoirs with light to medium oil gravity, strongly water-wet reservoirs, or reservoirs sensitive to water flooding. This method may be implemented right after primary recovery if needed.

After the injection volume target is reached, water flooding may be performed to sweep the oil left in the lower part of the reservoir due to the gravity segregation. It helps to increase the recovery in low permeability or homogeneous reservoirs. Other types of gas, such as nitrogen, can maximize gravity segregation, although this might not be preferred.

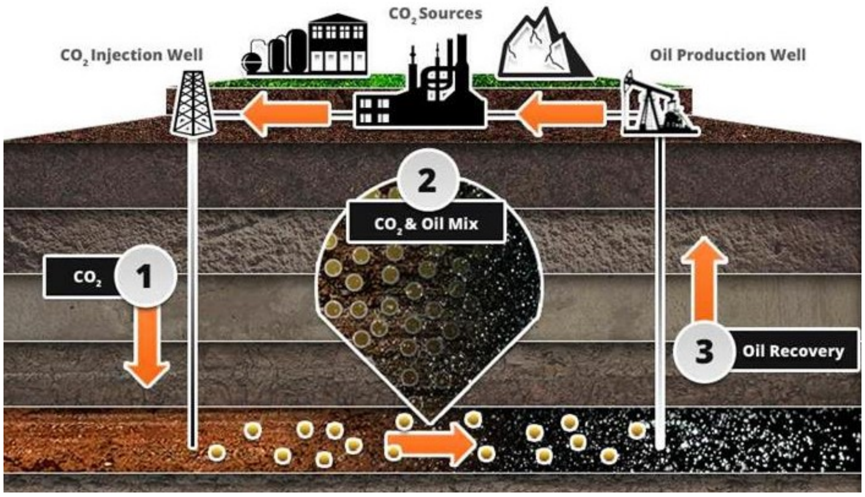

Figure 2.3: CO_2 continuous injection illustration (LLC, 2020).

Figure 2.3 shows the injection cycle of this design. Using this design, as we only have one injection fluid type (compared to WAG), the work can be considered almost the same as water flooding, except the tools should be more corrosive-resistant. Injection pattern is critical, but in some cases, this can not be prioritized as adding or converting wells will cost extra expenses. It reflects the capital cost that will be high, especially from surface facilities, as most fields are not prepared for CO_2 injection initially.

The main problem here is CO_2 availability. It is a common problem for other CO_2-EOR design, but CO_2 in bulk for this design is usually the main reason this method is not employed. Channelling, which causes early breakthrough, is also one of the concerns. As the gas density and or viscosity is lower than the fluid system we have in reservoir, it will go through available thief zones, hence reduce the amount of oil that can be recovered from the project. These two problems can be tackled by employing WAG design.

2.1.2 CO_2 Water Alternating Gas (WAG)

In this method, water and gas are injected in cycles until the targeted gas injection volume is reached. The cycle changes can be determined either by the total volume of fluid injected or the total injection time for each injection phase. Conventionally, every cycle consists of the total amount of gas and water injected, but tapered WAG can also be employed as technology advances. Tapered WAG is usually employed to reduce the amount of CO_2 needed, prevent an early breakthrough, and improve the oil recovery.

After the targeted slug volume is injected, water flooding can be performed. It is suitable for layered reservoirs with different permeability in each layer. Gas flooding can be another option, as it is usually cheaper gas such as air or nitrogen. Later, CO_2 inside the reservoir will migrate to the upper layer of the reservoir.

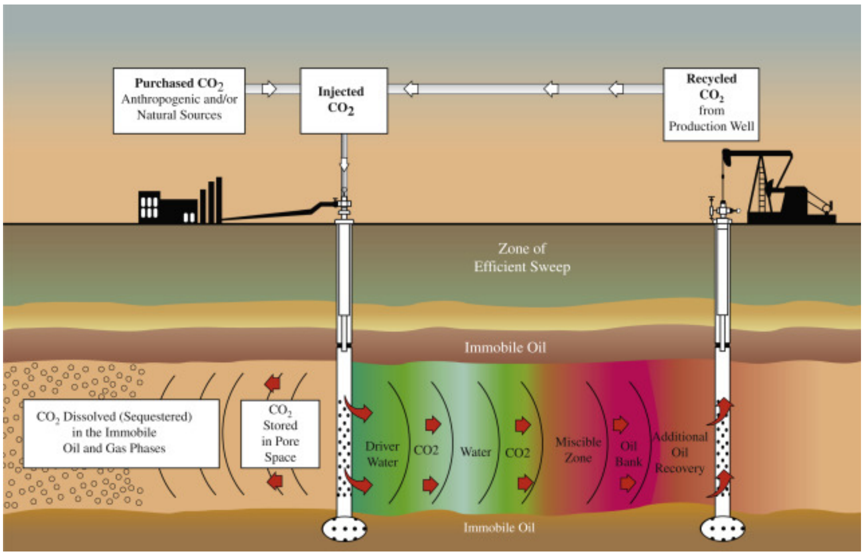
Purchased CO_2
Anthropogenic and/or Natural Sources
Injected CO_2
Recycled CO_2 from Production Well
Zone of Efficient Sweep
Immobile Oil
CO_2 Dissolved (Sequestered) in the Immobile Oil and Gas Phases
CO_2 Stored in Pore Space
Driver Water
CO2
Water
CO2
Miscible Zone
Oil Bank
Additional Oil Recovery
Immobile Oil

Figure 2.4: CO_2 WAG illustration (Lake et al., 2019).

Figure 2.4 shows the illustration of CO_2-WAG design. Water alternating CO_2 injection can tackle the gas override and channelling, which is one of the main problems with the previous design. Although the amount of CO_2 needed may be the same, the need in bulks throughout the project lifetime will be less burdensome than before. A new problem, storing recycled CO_2 during the water injection phase, also needs to be considered. This design can be adaptive, as the cycle can be modified based on the real-time reservoir response.

Although these problems can be tackled, new problem await. As we have CO_2 and water, regular workover to change the injection fluid are needed. In addition, CO_2 and water mixture can form carbonic acid. This corrosive solution can corrode pipelines, so extra preparations are needed to reduce the effect of this acid.

2.1.3 CO_2 EOR Reservoir Engineering Design

A comprehensive study needs to be performed before conducting CO_2-EOR. It is noted that not all reservoirs are suitable for this method. Many technical screening criteria are available, which are gathered from successful projects for every EOR method. One of them, written by Al Adasani and Bai (2011a), summarized EOR Projects based on oil properties (oil gravity and viscosity) and reservoir characteristics (porosity, permeability, depth, and other properties). Complementing this, rough economic screening usually is performed for testing the project profitability.

CO_2-EOR is usually applied in medium to light oils, which varies between 28 to 45°API, with viscosity averaging in 2.1 cp (Al Adasani and Bai, 2011a). When a reservoir passed the screening criteria, then reservoir engineering design can be started. Jarrell et al. (2002)

mentioned four points that need to be performed to design an EOR project. These points collect valid input data, history matching, predicting the EOR performance, and determining the optimum flow design.

The first step, collecting data, involves tests (mainly laboratory tests) that explain the behavior of reservoir when CO_2 is injected. Related tests are slim-tube, multi-contact, and swelling test. Then, data upscaling will be performed to the history-matched reservoir model. Here, the behavior will be observed from a field scale. The reservoir model then can be used to make predictions, which reflects the field performance.

The last step, which is highly iterative and time-consuming, is determining the flow design. Here, the field constraints will be included, such as CO_2 availability, pressure limit, pipeline design, and surface facilities condition. Based on this and other parameters that influence the EOR performance, optimization is performed. This step is repetitive, as we want to reach the optimum condition of the field performance. A complex reservoir model with a high number of grids, a high number of study parameters, and a complex fluid model will increase the time needed to perform this study.

2.2 Proxy Model

Performing exhaustive study takes much running time and memory when performed with current reservoir model technology. To tackle this problem, proxy model studies are performed and yield a promising solution. Proxy model is a mathematically or statistically defined function that replicates the simulation model output for selected input parameters (Zubarev et al., 2009). One of the proxy model study results, performed to mimic the oil saturation of the reservoir model, is attached in **Figure 2.5**. The ability of a proxy model to learn complex reservoir model behavior is proven by recent studies, both synthetic and field model.

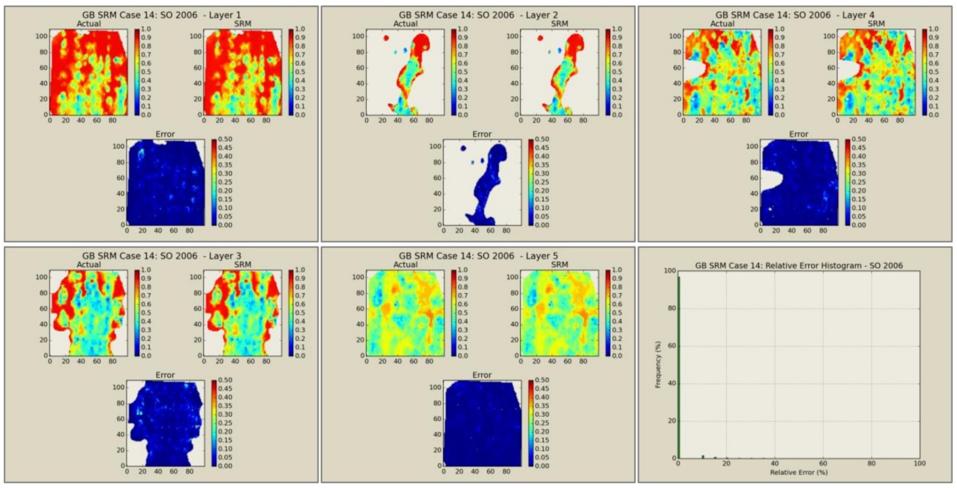

Figure 2.5: Study results of proxy modeling for CO_2-WAG study (Gholami, 2014)

A proxy model learns from the given training and validation data generated from the response surface or simulation models output. However, a proxy model is not response surfaces nor statistical representations of the simulation model. It is an engineering tool that honors the physics of the problem we have. This powerful tool is constructed based on data managed, clustered and filtered into information, later learned as knowledge. The proxy model has different naming in other studies, such as Smart Proxy Model (SPM), Surrogate Reservoir Model (SRM), Dynamic Proxy Model. Nevertheless, all have the same meaning as the proxy model we mentioned here.

Problems such as sensitivity analysis, optimization study and history matching are some studies that can be solved with the help of proxy models (Zubarev et al., 2009). These problems almost have the same way to solve, by iterative evaluation. As a proxy model has higher computational efficiency than reservoir model, exhaustive sampling can be performed with this method. How to build it, the growth of this study and room for improvements based on the previous studies will be described in this sub-chapter.

2.2.1 Building Proxy Model

There are no direct guidelines that state the steps to build the proxy model. However, most studies are performed in the same way. Here we summarized (in steps) a guideline for developing a proxy model.

1. Determining the study objective
   As proxy model only learns from the given sets of information, this creates a limitation for the model, which is case-specific. Different proxies need to be built for different study objectives, leading to sampling, proxy input-output combination, study limitations, and algorithm to solve the problem. However, the most important thing, the objectives will determine the scale and the complexity of our proxy model. There are different scales of proxy models based on the size of their elemental volume, summarized in **Figure 2.6**. Again, determining which scale to be used is based on the needs of the proxy to be built.

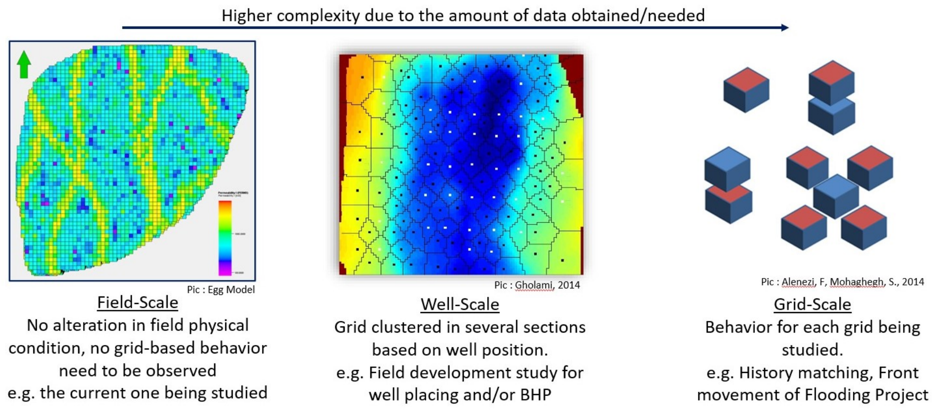
Higher complexity due to the amount of data obtained/needed
Pic : Egg Model
Field-Scale
No alteration in field physical condition, no grid-based behavior need to be observed
e.g. the current one being studied
Pic : Gholami, 2014
Well-Scale
Grid clustered in several sections based on well position.
e.g. Field development study for well placing and/or BHP
Pic : Alenezi, F, Mohaghegh, S., 2014
Grid-Scale
Behavior for each grid being studied.
e.g. History matching, Front movement of Flooding Project

Figure 2.6: Summary of proxy model scale and application.

Grid-based, the smallest, can track the change of pressure and saturation at the grid block level. This scale can monitor the pressure and rate changes near the injection and production wells, followed by a well-based proxy. The last one is field-based proxy, where a segment of a field (or the whole field) is being observed. The different scale yields different input needed, **Table 2.2** can be used as reference.

Table 2.2: Needed input for each type of proxy model.

Data	Grid-based*		Well-based*		Field-based	
	Property	Domain	Property	Domain	Property	Domain
Static	Grid Type	Grid	Drainage Area	Well		
	Location (i, j, k, Long, Lat)	Grid/Tier	Location (i, j, k, Long, Lat)	Well		
	Thickness	Grid	Thickness	Tier	No input needed here	
	Porosity	Grid	Porosity	Tier	(constant geological/	
	Permeability (x,y,z)	Grid	Permeability	Tier	static condition)	
	Grid top	Grid/Tier	Grid Top	Tier		
	Distance to boundary	Grid/Tier	Distance to boundary	Well		
Dynamic	Time		Time		Time	
	Pressure	Grid/Tier	Pressure	Tier		
	Saturation	Grid/Tier	Saturation	Tier		
	CO_2 Mole Fraction	Grid/Tier	CO_2 Mole Fraction	Tier		
	COW BHP	Well	COW BHP	Well		
	COW Amount of Prod/Inj	Well	COW Amount of Prod/Inj	Well		
	Amount of Prod/Inj	Field	Amount of Prod/Inj	Field	Amount of Prod/Inj	Field

*Gholami (2014)

2. Data Sampling
After determining the study objectives, proxy scale and input data for the proxy, and data sampling can be performed. Data sampling is usually performed by running the reservoir model. The results to be learned are then being sampled for proxy learning dataset. How to perform the data sampling is approached from available statistical sampling. In this study, Latin Hypercube Sampling (LHS) will be used as the problem is to solve the optimization study.

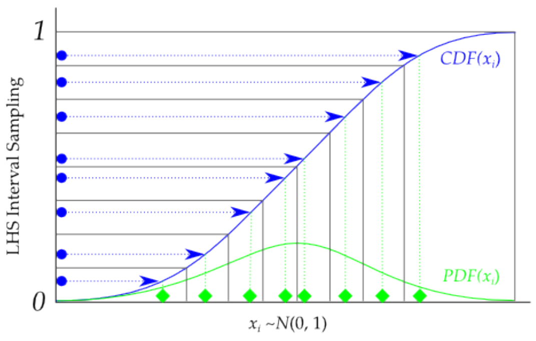

Figure 2.7: Latin Hypercube Sampling.

Rather than random sampling, LHS performs stratified sampling to improve the coverage of the solution space. Shown in **Figure 2.7**, sampling was performed by segmenting cumulative distribution function (CDF) into n equal, non-overlapping intervals. These intervals will make equiprobable intervals in our horizontal axis.

After segmentation, exactly one random value in between each interval will be selected. When inverted into the horizontal axis, precisely one value will be sampled for each equiprobable intervals. A random process must be used to ensure the randomness of each segment. Performing with LHS ensures that the entire range is completely covered without one variable dominating the others. It makes LHS more efficient than random sampling for a study with ample solution space. LHS is usually performed in safety assessment, computer modeling, and petroleum industry, particularly in optimization schemes (Iman, 1999).

3. Data Management
   After obtaining the sampling plan, reservoir model runs will be performed. Many data points can be obtained, yet not all of them will be used for our proxy model training. By reviewing the study objective, understanding the reservoir model physical behavior (**Table 2.2** as guideline), data can be filtered. All the needed data will then be organized as input and output combination for the proxy to learn.

4. Designing and building the proxy model
   Using available machine learning or deep learning models, a proxy model can be built. The proxy model will approximate the numerical reservoir model. It should mimic the nonlinearity in responses from the model. The complexity of the proxy model itself reflects the complexity of the reservoir model.

   Four common forms of a proxy model are polynomial regression model (PR), multivariate kriging model (KG), thin-plate splines model (TSP) and artificial neural network (ANN) (Zubarev et al., 2009). Regardless of the models, the typical workflow to build a proxy is shown in **Figure 2.8**. The first three steps are the ones performed during defining the study objective and doing data sampling.

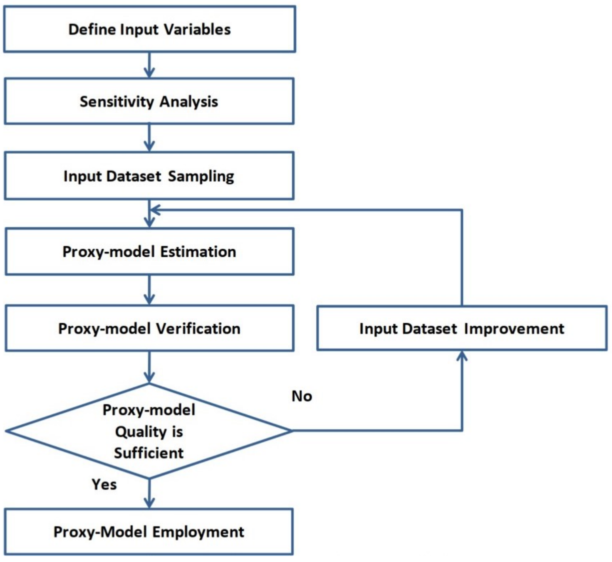

Figure 2.8: Proxy modeling workflow (Zubarev et al., 2009).

In this study, ANN will be used to build the proxy. ANN is a computational model inspired by the biological behavior of human neurological system. As shown in **Figure 2.9**, ANN consists of inputs, weights, bias, and activation function. This structure, formed in several layers, each containing several nodes before reaching the output, is called topology.

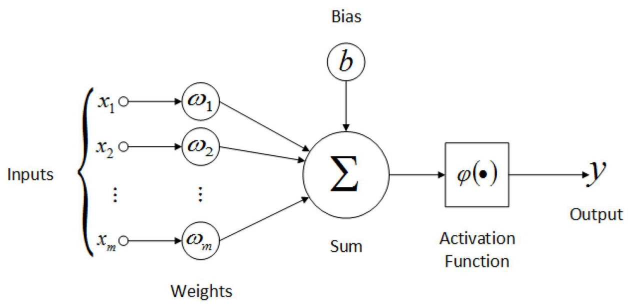

Figure 2.9: Main ANN structure.

The activation function gives ANN the ability to model the nonlinearity of reservoir model. ANN then learns from forward and backpropagation, where the weights will be updated during each process to minimize the error. During the forward propagation, the network moves from the input layer to the output layer. Passing through each node and transformed by the activation function, the results will reach the output. It will be noted as the neural network prediction.

The learning process happens as we have the output dataset. The differences between the predicted and the actual output are then calculated using the loss function, such as mean squared error (MSE), root mean square error (RMSE), absolute error (AE), and any other loss function. Backpropagation will then be performed to minimize the loss function. The loss will be sent back to the input layer as a fraction of the total signal of the loss. These two processes will be performed until error satisfies the limit or the number of iterations (epoch multiplied by batch size).

Underfitting may happen if the network is not adequately trained. Overfitting, where ANN learns the noise instead of the signal, may also happen during this process. To avoid this, the database needs to be separated into training/validation datasets so that the validation data loss function will prevent overfitting. To make an ANN, topology needs to be defined, consisting of the number of hidden layers, nodes in each hidden layer and the activation function. Then, the node weights can be estimated by a supervised learning algorithm.

Hyperparameter is a parameter that control the learning process. Activation function, number of hidden layers, and number of nodes count as hyperparameters in ANN. Other than that, we have other hyperparameters such as learning rate. optimization function/optimizer, batch size, dropout and number of epoch. Learning rate is the step size for each iterations and optimizer is the optimization function to minimize the loss function. Dropout is the probability where nodes are randomly disconnected during training and batch size is the number of data points that pass through neural network every step. Epoch is the amount of times to go through our training data.

5. Testing the robustness of the proxy model
   The robustness of an ANN can be tested by performing a blind test. A blind test dataset is a dataset that is not used to train the network. This dataset can be used as an indicator of whether an ANN, which will be our proxy model, represents the reservoir model behavior or not. We can see whether our proxy has the prediction ability, which will be necessary for the study.

6. Performing the desired study
   After the robustness of our proxy is confirmed, the preferred study such as optimization study, history matching or sensitivity analysis can be performed using the available computational algorithm to solve the problem.

2.2.2 Previous studies on proxy modeling

The proxy model study was initiated in the 2000s. Most of the studies were performed to solve exhaustive problems as mentioned before. Zubarev et al. (2009) reviewed the growth of proxy model study from 1998 until 2008. They found that all proxy-modeling techniques showed dependence on the complexity of the reservoir model, solution space dimensions, and dataset quality. History matching, sensitivity analysis, and optimization studies were already performed using a proxy model as a substitute.

Based on their study of history matching, proxy models are able to calculate the objective function values. However, they fail to predict the global minimum compared to reservoir simulation that could locate it. In sensitivity analysis, proxy models can predict the field performance uncertainties, where good results are reported using a different type of proxy models. This result is obtained after applying a space-filling design (LHS) rather than other traditional designs to form the dataset. Proxy model is often used to assist optimization study. Past studies show that it can locate the local minimum, yet not all of them can. From Zubarev et al. (2009) study, solution space based on their sampling is shown in **Figure 2.10**. It shows that 100 samples were able to locate the known optimum of one in-fill optimization study. However, even 200 samples were not enough to locate the optimum for two in-fill optimization studies. In one of their case study, the proxy cannot locate the optimum solution even though the global optimum is used as a training case.

Gholami (2014) interpreted this as one of the typical examples where the technology is misused and misjudged. The neural network used was treated merely as a regression tool, which sets the study as a failure. Deploying an ANN needs to be taken care of as an attempt to observe, learn and generalize. Thus, particular comprehension of machine learning and deep learning are needed before applying them.

Gholami (2014) applied proxy model, so-called SRM, in her study to a more complex problem, where smaller elemental volume was studied. The study was applied to CO_2-WAG to mimic the grid behaviors (pressure and saturation), followed by a larger scale, well-based behaviors (production rates of oil, gas and water) where both of them are con-

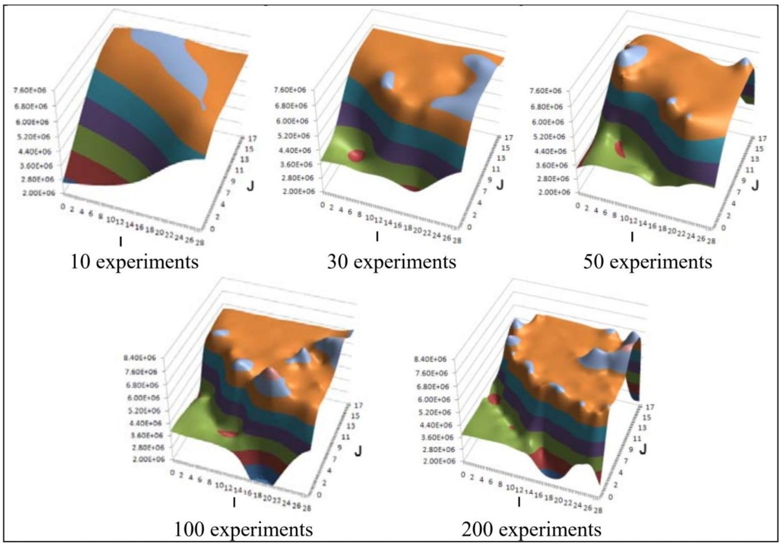

Figure 2.10: Evolution of the solution space for single in-fill optimization (Zubarev et al., 2009).

nected. Grid-based results are shown in **Figure 2.5**. Based on the study results, the constructed proxy model learned the preferred pressure, saturation and rate behavior, where one year was used as the timestep interval for reporting frequency.

Amini (2015) performed a detailed study for a grid-based proxy model following Gholami's results. The cascading effect and comparison between fine-grid and coarse-grid reservoir model were analyzed in CO_2 sequestration study. The study shows that the coarser model requires fewer runs for training purposes than the fine grid model. In alignment with that, the cascading procedure shows significant errors when observed in the last time step, as shown in **Figure 2.11**. The figure explains the saturation error for each grid in the first layer.

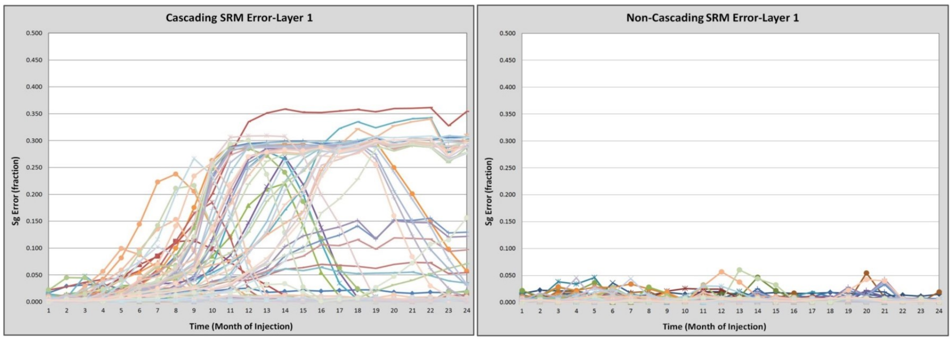

Figure 2.11: Cascading and non-cascading error results. (Amini, 2015).

This literature has been used as the primary reference for developing a good proxy model until now. In one of the recent studies performed by Chaki et al. (2020), they developed a proxy model to perform history matching using the Brugge field model as the reservoir model to be learned. They applied two methods, deep neural network (DNN) and recurrent neural network (RNN), to build a proxy that learns the behavior of the reservoir model (oil and water production rate and cumulative production). RNN shows better performance than DNN, yet the amount of time needed to construct it is 15 times higher than DNN.

Two other studies are aligned with this study. Optimization study for CO_2 EOR projects are performed by Amar et al. (2018), followed by their newest article (Nait Amar et al., 2020). They performed a CO_2-WAG optimization study using a proxy model in both articles. Both ANN and hybrid support vector were used to learn the reservoir behavior (oil and water rate) and then perform a CO_2-WAG optimization. One of the proxy model performances is shown in **Figure 2.12**. A recent study also shows that a proxy model was built for a fractured reservoir model (Ng et al., 2021).

Two studies are found in which align with this study. Optimization study for CO_2 EOR projects are performed by Nait Amar et al. (2018), followed with their newest article (Nait Amar et al., 2020). They performed a CO_2-WAG optimization study using a proxy model in both articles. Both ANN and hybrid support vector are used, and those methods can learn the reservoir behavior (oil and water rate) and then perform a CO_2-WAG optimization study. One of the proxy performance is shown in **Figure 2.12**. A recent study shows that a proxy model can be applied for fractured reservoir model (Ng et al., 2021).

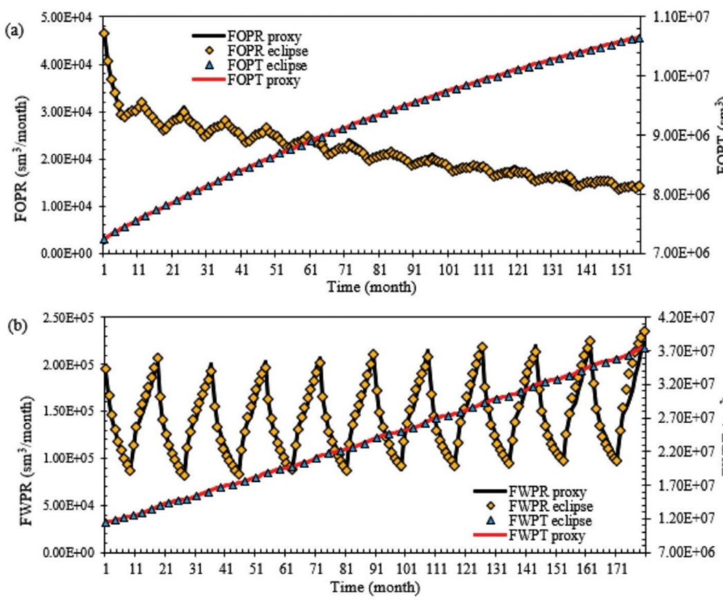

Figure 2.12: Proxy performance on CO_2 WAG study (Nait Amar et al., 2020).

2.2.3 Room for Improvements

Many studies related to proxy modeling have been performed, but many research gaps can be found. Most studies show that the proxy model can be used as a powerful tool for many tasks, such as a substitute for the reservoir model or an exhaustive study. It can break the current limitation in reservoir modeling (tremendous running time and high memory consumption). No studies are focusing on the feasibility of this idea to be applied in different geological models.

Most studies worked with only one geological model. Hence no information is available about the complexities faced when building a proxy model for different reservoir models. Many studies mentioned that the proxy would reflect the complexity of the reservoir model, yet quantitative results are to be found. Starting with this gap, later we can see whether a proxy model is worth being built as a substitute for the reservoir model or not. It is worth noting that the proxy model cannot be used as a substitute for the simulation model in different studies due to its case-specific limitation.

Alongside this, as optimization is one of the procedures that need to be performed for the EOR studies, proving the feasibility of the proxy model can shorten the duration of the pre-study of EOR projects. It can be used as one of the decision tools for the EOR study. Not only for the EOR study, but this will also help the study of production optimization.

2.3 Optimization

As one of three common proxy model applications, optimization will be performed in this study. Optimization or optimization problem is a study to find the best solution from all feasible solutions. When viewed from the mathematical perspective, optimization is a study to find the best-generated values from the objective function in the defined domain or solution space. It can be expressed as:

$$f : A \to \mathbb{R}^n$$
$$\tilde{f}(x) := -f(x), \tilde{f} : A \to \mathbb{R}^n$$

Most optimization algorithms, also known as minimization problems, are expressed as shown in the equation above. Maximization can be solved just by flipping the negative term of the study or the objective function. A itself is a subset of Euclidean space $\mathbb{R}^n$ with n-dimension(s), in which the constraints need to be satisfied. The domain A of f is called the search space, and the elements of A are called feasible solutions.

The goal of optimization is to find the global maximum or minimum. However, several local maximum or minimum are scattered throughout the solution space, as shown in **Figure 2.13**. A local minimum is the best solution in the nearby solution space area, while a global minimum is the best solution for the whole solution space.

Nowadays, most optimization problems are solved using optimization algorithms. These algorithms are employed to prevent the calculation convergence in the local minimum (or

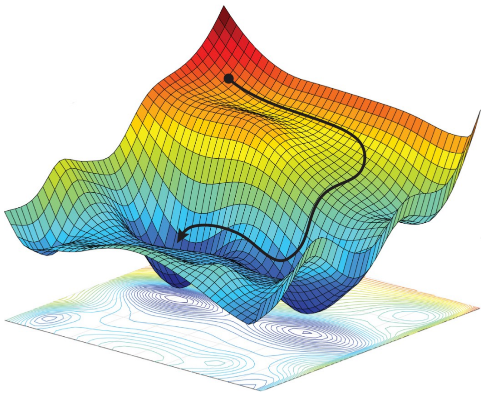

Figure 2.13: Non-convex optimization solution space illustration (Amini et al., 2018).

maximum). However, most of them cannot distinguish between local and global minimum in the solution space. To identify this, the whole solution space must be computed. However, this is nearly impossible as it is computationally demanding and time-consuming, especially for high dimensional optimization study. Most algorithms limit the study based on the number of iterations defined differently on each algorithm, such as the number of generations and swarm size.

Optimization in Petroleum Industry

Optimization is one of the most widely performed studies in the petroleum industry. Starting from the exploration, studies such as near-wellbore profile management and near-wellbore conformance management are performed. Moving to the planning for exploiting the reservoir, well completion and maximizing productivity index are performed, which count as a part of the optimization study.

Other typical optimization applications in this industry are gas-water coning and fingering prevention, well stimulation, sand control management, artificial lift performance study, and many more studies. It can be said that optimization is an integral part of petroleum study. From the reservoir to the production facilities, most optimization studies are performed mainly to maximize hydrocarbon production.

Optimization for CO_2-EOR Design

This study will be focusing on optimization for CO_2-WAG EOR design. Not very different from the CO_2 flooding study performed in the previous study (Matthew, 2020), most parameters studied here are the same, except we have water injection rate and half-cycle. Parameters that can be studied for CO_2-EOR design are listed below.

1. Gas injection rate
   This variable depends on the CO_2 availability, where both source and transportation will control this parameter. One of the main reasons why CO_2-EOR does not meet

the criteria is the CO_2 availability. Hence, this is one of the main study parameters in most CO_2-EOR studies.

2. Water injection rate
   CO_2 will increase the displacement efficiency while our water will increase the sweep efficiency indirectly. Most fields have no problems with the water source for injection, yet this can be studied to fit the field condition.

3. Halfcycle length
   This is the main parameter of the WAG project. Half-cycle length affects the recovery directly. Currently, more studies related to this parameter are being performed, such as a study about tapered WAG by Khan et al. (2016) and AlOtaibi et al. (2015). Other than that, this parameter will determine the schedule of workover to change the injection fluid.

4. Total slug size injected
   Most pilot studies use 1.2 reservoir pore volume as the target of total gas injected into the reservoir. The increase of total volume injected will increase the recovery, yet at some point, it will not be economical to be continued. Some projects determine this by project time rather than total slug size injected.

Other parameters can be studied if needed, such as well placement study and injection intervals. Most of the optimization studies are coupled with the economic study in which sensitivity analysis of oil prices, gas prices, and other economic variables are performed. However, this will not be our primary focus as only technical aspects will be analyzed for the optimization objectives and parameters.

2.3.1 Single, multi and many-objective optimization problems

Most optimization problems that we have in engineering designs have multiple conflicting conditions/criteria needed to be solved. Not to mention that more than one objective function might need to be solved. Other than that, the objective function can be said as a ”black box”. This is because the derivative of this function is not always available. To overcome this, existing optimization algorithms perform either in finite steps, iterative, or heuristics that may converge to the solutions.

Single-objective Optimization Problem

$$\begin{array}{rll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \ldots, m \\ & h_j(x) = 0, & j = 1, \ldots, p \end{array}$$

The common formulation of a single optimization problem is shown above. The objective function is followed by several sets of constraints, inequalities and equalities that needs to be satisfied. In a solution space, there exists an optimum value in which these conditions are satisfied. Most of the studies in petroleum engineering involve non-linear functions where derivations do not exist. Hence an optimization algorithm can be employed to solve this. The main idea is the same, where algorithms are employed to find the global minimum rather than trapped in the local minimum. Shown below is the solution space of single-objective optimization problem (**Figure 2.14**).

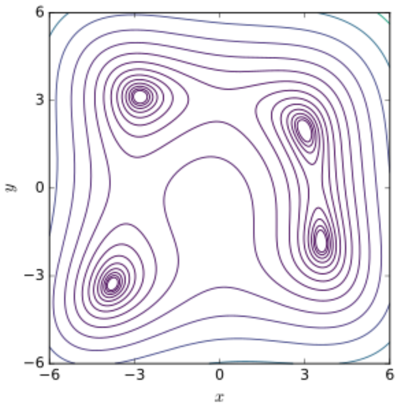

Figure 2.14: Himmelblau solution space (Himmelblau et al., 2018)

Different methods can be used, such as methods that approximate Hessians (e.g. Newton's method), the gradient (e.g. conjugate gradient method), or evaluate the function values (e.g. pattern search methods) or heuristics. The choice depends on the user. It is expected that the main problem of performing an optimization study is the computational load in evaluating the objective functions, where most of them are heavier than the optimizer. It needs to be considered when choosing the iterations or step size for every algorithm used.

Multi-objective Optimization Problem

$$\begin{aligned} &\text{min } (f_1(\vec{x}),\ f_2(\vec{x}),\ f_3(\vec{x})) \\ &s.t.\ \vec{x} \in X \end{aligned}$$

This class consists of two to three objective functions, as the formulation is shown above. **Figure 2.15** shows the illustration of the solution space for multi-objective optimization problem (illustrated for two-objective optimization problem). Rather than being illustrated with input parameters, the solution space is illustrated with the objective functions we have (2D for two objective functions and 3D for three objective functions). In that space, there exists a feasible solution region. For multi-objective optimization, typically there is no feasible solution that minimizes all objective functions simultaneously.

We will have several Pareto optimal solutions. Here, the solution cannot be improved unless we degrade one of the other objectives. All Pareto optimal solutions do not have any objective function that dominates it. Collection of Pareto optimal solutions is often called Pareto front (shown in red). Pareto front will be bounded by nadir objective vector ($\vec{z}^{nad}$) and ideal objective vector ($\vec{z}^{ideal}$). This will give upper and lower bound of objective function values, which can only be seen if the Pareto optimal set is known.

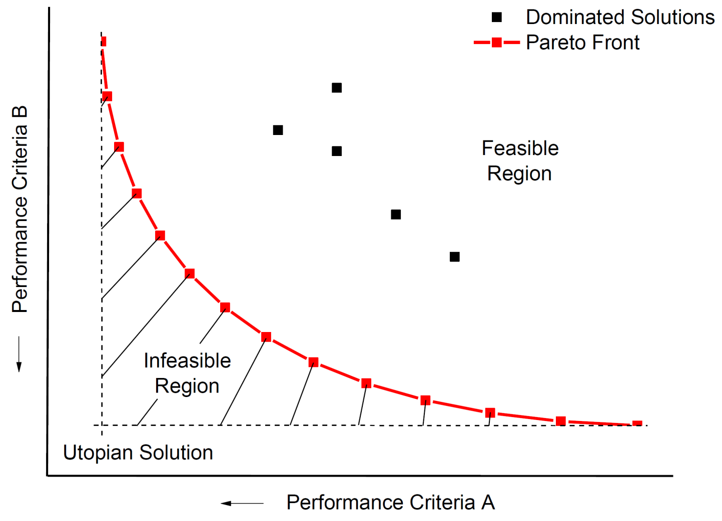

Figure 2.15: Multi-objective optimization solution space illustration (Schweidtmann, 2021)

Solving a multi-objective optimization problem is sometimes understood as approximating or computing all or a representative set of Pareto optimal solutions (Ehrgott, 2005). Then, deciding which point to be selected from Pareto optimal solutions will be subjective to the researcher's preferences. Most multi-objective optimization algorithms follow this concept to solve the optimization problems. Some available algorithms are Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA-2).

Many-objective Optimization Problem

$$\begin{aligned} &\min\ (f_1(\vec{x}),\ f_2(\vec{x}), \ldots,\ f_k(\vec{x})) \\ &s.t.\ \vec{x} \in X \end{aligned}$$

This class refers to optimization with more than three objectives. This became a hot topic over the past years due to the emergence of many objective optimization studies for real-world studies. As reviewed by Fleming et al. (2005), many studies are performed to tackle the ineffective Pareto dominance, the inefficiency of recombination, and other problems encountered in this optimization class. Several optimization algorithms exist to solve this problem, such as Non-dominated Sorting Differential Evolution based on Reference points (NSDE-R) and Non-dominated Sorting Genetic Algorithm (NSGA-III).

2.3.2 NSGA-II

There are a lot of multi-objective optimization algorithms proposed. We used NSGA-II in this study. Non-dominated Sorting Genetic Algorithm II (NSGA-II) is an improvement from NSGA due to its high computational complexity of non-dominated sorting, lack of elitism, and need for specifying the sharing parameter (Deb et al., 2002).

NSGA-II works based on genetic algorithm. Genetic algorithm is an optimization algorithm inspired from natural evolution theory. The algorithm reflects the natural selection process where the fittest individuals are selected to reproduce the new offspring of the next generation. Initial population is predetermined. Fitness function will be employed to determine how fit an individual and scored. The probability of each individual to be selected for reproduction is based on the fitness score. From two individuals, called parents, crossover point will be chosen within the genes.

The crossover will results to a new offspring. During the process, mutation may happen in the genes based on the mutation probability. The algorithm will be terminated if the population has converged or reach the maximum number of generations. In the application, the population is fixed. For each iteration of generation, the least-fitted individual will die, and new offspring will fill the position. This will give new population which is better than the previous generation.

NSGA-II can be categorized as an evolutionary algorithm. This algorithm type was developed due to issues found in the classical and gradient-based techniques including the performance that depends on the initial guess and the sub-optimal convergence issues. This algorithm uses genetic algorithm as its fundamental knowledge. Three features of this algorithm are:

1. Elitist principle
2. Explicit diversity preserving mechanism
3. Emphasis the non-dominated solutions

Figure 2.16 shows the illustration of NSGA-II procedure, where P indicates the Pareto solutions from the previous timestep, Q indicates the new Pareto solutions obtained from the new offsprings, R stands as the whole population and PF stands as Pareto Front with rank as its indices.

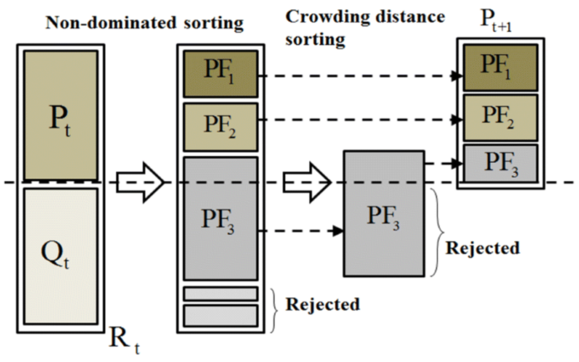

Figure 2.16: Procedure of NSGA-II (Kumar and Yadav, 2019)

Algorithm 1 and **Algorithm 2** list the iterations illustrated in the figure. This algorithm will perform a non-dominated sorting and a classification by fronts. This process is in line with combining the parent and offspring populations. They are then sorted based on the ascending level of non-domination. The new population will be taken based on front ranking. In the figure, the third rank data will need to be taken partially. For this, crowding distance sorting will be performed. Crowding distance is performed based on the density of solutions around them, where the less dense solution will be taken. After these processes are finished, the genetic algorithm will take over. New offspring of the population based on this population will be created, followed by crossover and mutation.

Algorithm 1: NSGA-II Algorithm, Fast non-dominated sort loop (Deb et al., 2002)

```
for each $p \in P$ do
    $S_p = \emptyset$;
    $n_p = 0$;
    for each $q \in P$ do
        if $p \prec q$ then                      // If $p$ dominates $q$
            $S_p = S_p \cup \{p\}$       // Add $q$ to the set of solutions
        else
            $n_p = n_p + 1$ // Increment the domination counter of $p$
        end
    end
    if $n_p = 0$ then                   // $p$ belongs to the first front
        $p_{rank} = 1$;
        $F_1 = F_1 \cup \{p\}$;
    end
end
$i = 1$                                // Initialize the front counter
while $F_i \neq \emptyset$ do
    $Q = \emptyset$                            // Used to store the next front
    for each $p \in F_i$ do
        for each $q \in S_p$ do
            $n_q = n_q - 1$;
            if $n_q = 0$ then                // $q$ belongs to the next front
                $q_{rank} = i + 1$;
                $Q = Q \cup \{q\}$;
            end
        end
    end
end
$i = i + 1$;
$F_i = Q$;
```

A comprehensive study assessing the NSGA-II performance was carried out by Deb et al. (2002). Their study shows that NSGA-II outperforms other optimization algorithms (SPEA and PAES) for almost all tested cases, followed by SPEA and PAES. We can see from **Al-**

gorithm 3 that the algorithm is very straightforward. We can use this algorithm that is pre-coded already in most computational languages.

Algorithm 2: NSGA-II Algorithm, crowding distance loop (Deb et al., 2002)

```
$l =| I |$                                        // number of solutions in $I$
for each $i$, set $I[i]_{distance} = 0$ do              // Initialize distance
    for each objective $m$ do
        $I =$sort$(I, m)$          // Sort using each objective value
        $I[1]_{distance} = I[l]_{distance} = \infty$   // to select boundary points
        for $i = 2$ to $(l - 1)$ do                  // for all other points
            $I[i]_{distance} = I[i]_{distance} + (I[i+1].m - I[i-1].m)/(f_m^{max} - f_m^{min})$;
        end
    end
end
```

Algorithm 3: NSGA-II Main Algorithm (Deb et al., 2002)

```
$R_t = P_t \cup Q_t$;
$F =$Fast-non-dominated-sort$(R_t)$;
$P_{t+1} = \emptyset$ and $i = 1$;
while $| P_{t+1} | + | F_i | \leq N$ do
    crowding-distance$(F_i)$;
    $P_{t+1} = P_{t+1} \cup F_i$;
    $i = i + 1$;
end
Sort$(F_i, )$;
$P_{t+1} = P_{t+1} \cup F_i[1 : (N- =| P_{t+1} |)]$;
$Q_{t+1} =$make-new-pop$(P_{t+1})$;
$t = t + 1$
```

Chapter 3

Methodology, Problem, and Model Description

To see the feasibility of the proxy model as a substitute for doing an exhaustive study, a study on building a proxy model from scratch was performed. Previous study (Matthew, 2020) shows that the proxy model is capable of performing an optimization study for CO_2 flooding case to represent the reservoir model (Egg Model). The study reported 1.53% error compared to results obtained from the reservoir simulation model (Eclipse).

Several recommendations were given, such as testing the capability of the proxy model as a field model substitute, deeper study on the proxy structure, and different optimization studies to be performed. These recommendations complement the room for improvements for proxy modeling study, as written in **Section 2.2.3** of this thesis. Based on that, a comprehensive study to see the capability of the proxy model and the complexity that we may encounter during the building phase is performed.

In this study, two or more proxy models will be built to represent two reservoir models. One of the models is the simple model taken from Jansen et al. (2014), while the other one is developed from Gullfaks K1/K2 dataset. Both proxies will solve the multi-optimization problem on CO_2-WAG design study using the same optimization algorithm. The details about the models, modified properties, optimization algorithm, and how the proxies will be built are described in this chapter.

3.1 Study Workflow

A workflow that illustrates how the study is performed is shown in **Figure 3.1**, inspired by our previous study. It is the proposed workflow for performing optimization study using proxy model as a reservoir model substitute. This workflow is proven to be working properly, which the details for each steps are described in the next chapter.

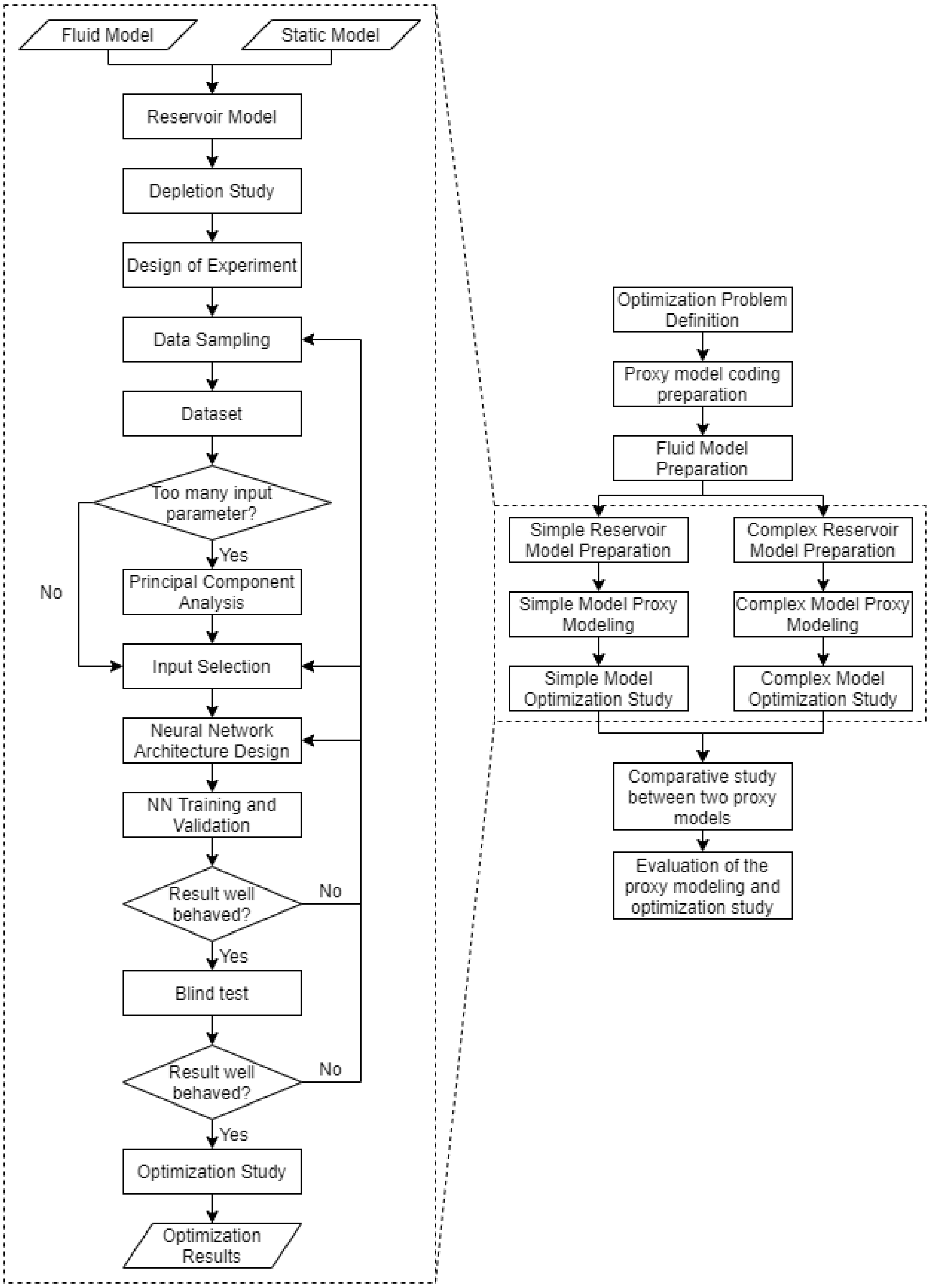
Fluid Model
Static Model
Reservoir Model
Depletion Study
Design of Experiment
Data Sampling
Dataset
Too many input parameter?
Yes
No
Principal Component Analysis
Input Selection
Neural Network Architecture Design
NN Training and Validation
Result well behaved?
No
Yes
Blind test
Result well behaved?
No
Yes
Optimization Study
Optimization Results
Optimization Problem Definition
Proxy model coding preparation
Fluid Model Preparation
Simple Reservoir Model Preparation
Complex Reservoir Model Preparation
Simple Model Proxy Modeling
Complex Model Proxy Modeling
Simple Model Optimization Study
Complex Model Optimization Study
Comparative study between two proxy models
Evaluation of the proxy modeling and optimization study

Figure 3.1: Study workflow.

3.2 Reservoir Model Description

As mentioned, two reservoir models are used in this study. The simple model will represent the Egg Model, while Gullfaks K1/K2 Model will represent a complex model. The overview of the complexity difference between the models is listed in **Table 3.1**. More details for each model and their alterations from the original model will be described.

Table 3.1: Model Overview

Parameter	Egg Model	Gullfaks Model
Permeability	Channel distribution	Heterogeneous
Porosity	Homogeneous	Heterogeneous
Faults	0	14
Fluid	From SPE136530	From SPE136530
Transmissibility	No multiplier	Heterogeneous multiplier
Relative permeability	Sand preset	Sand preset
Grid system	Cartesian	Cornerpoint
Grid size	Homogeneous	Heterogeneous
Initial condition	320 bar, 120.85°C, 1850m	320 bar, 120.85°C, 1850m
Wells	3 injectors, 3 producers	3 injectors, 3 producers
Perforations	Throughout all layers	Different for each well

ECLIPSE 300 and PETREL were used as the simulator for running the models. It is worth to be noted that an identical relative permeability relationship is applied to both models (**Figure 3.2**). Both reservoirs have the same rock characteristics (sandstone), so sand preset is used. The compaction function used for both models is the Newmann correlation for consolidated sandstone rock type.

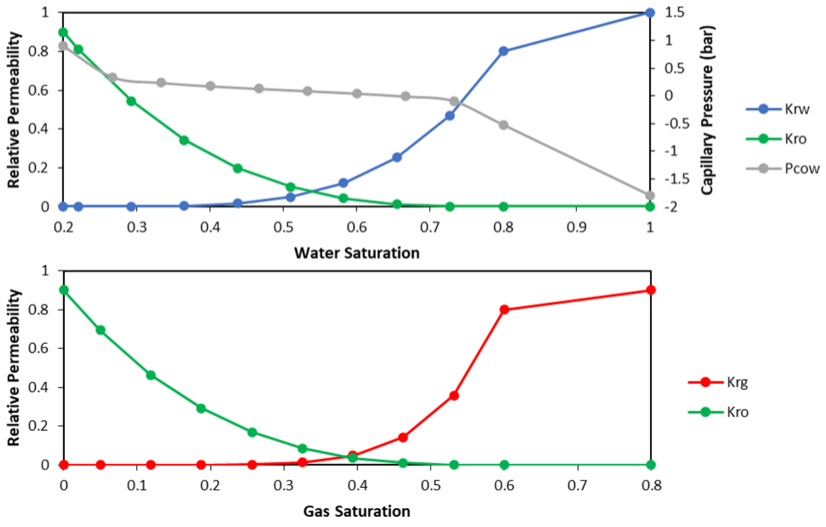

Figure 3.2: Relative permeability curve used from PETREL sand preset.

Both models are initialized using the same condition as described in the fluid model data. It is performed to ensure the miscibility condition is identical, as it will be applied to different geological conditions. The initialization condition might be a bit different for Norwegian continental shelf fields as the fluid model is taken from a Middle East field.

3.2.1 Fluid Model

Both models are applied using the same fluid model, taken from Negahban et al. (2010). The fluid model, categorized as light oil from its density, has low viscosity (based on EOS tuning results) and fulfills the screening criteria of CO_2-EOR (Al Adasani and Bai, 2011b). Constant composition expansion (CCE), differential liberation expansion (DL), multi-stage separator, slim tube, swelling, and multi-contact test data are available. These PVT test data are sufficient for fluid modeling in the CO_2-EOR study.

Due to the lack of molecular weight data on the matched pseudo-components, EOS tuning is performed from the available data. The primary process for EOS tuning here is splitting C_{20+} and regressing on conventional PVT test data (CCE, DL, multi-stage separator test). After that, lumping was performed into seven components. Regression is then performed based on available advanced PVT tests (slim-tube and swelling test), then the model can be used as reservoir fluid model.

This task was performed using CMG-Winprop, where Peng-Robinson is used as PVT EOS. The result was then converted to E300 with the help of PVTp. A multi-contact test is not used in this study as both software cannot work with it. No quality control (QC) was performed due to the lack of data data for QC material. The final composition after EOS tuning is tabulated in **Table 3.2**. The phase envelope can be seen in **Figure 3.3** and the EOS results with our PVT test data can be seen in **Figure 3.4**. The test results are present based CO_2 and hydrocarbon gas as injection fluid in the article. In this study, EOS tuning was performed based on the CO_2 test results.

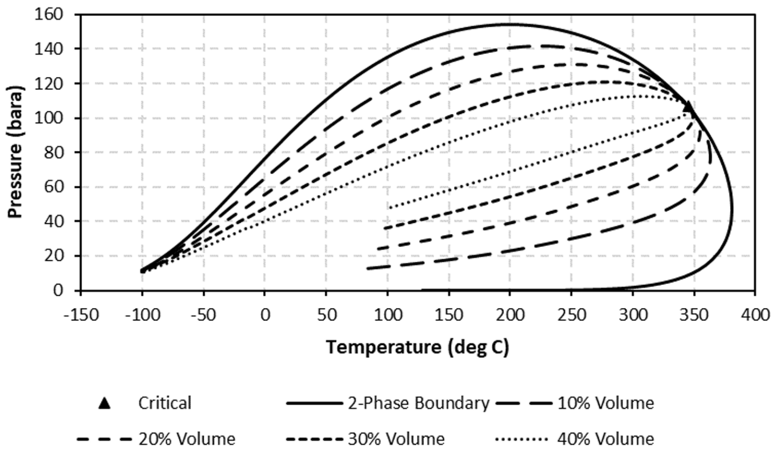

Figure 3.3: Reservoir fluid phase envelope.

Table 3.2: Component properties after EOS tuning.

Component	Z_i	P_c (atm)	T_c (K)	AF	MW	Vol Shift
CO_2	2.99	72.80	304.20	0.225	44.01	-0.054
N_2-C_1	29.43	45.26	189.77	0.008	16.19	-0.154
C_2-C_3	13.74	45.14	338.00	0.124	36.78	-0.454
C_4-C_6	15.32	34.56	460.09	0.229	70.30	-0.396
PS-1	20.02	24.89	563.92	0.815	112.67	-0.025
PS-2	12.96	16.58	699.03	1.060	198.94	0.048
PS-3	5.55	10.87	753.29	1.887	498.61	0.188

BIP	N_2-C_1	C_2-C_3	C_4-C_6	PS-1	PS-2	PS-3
CO_2	0.034	0.2	0	0.003	0.164	0

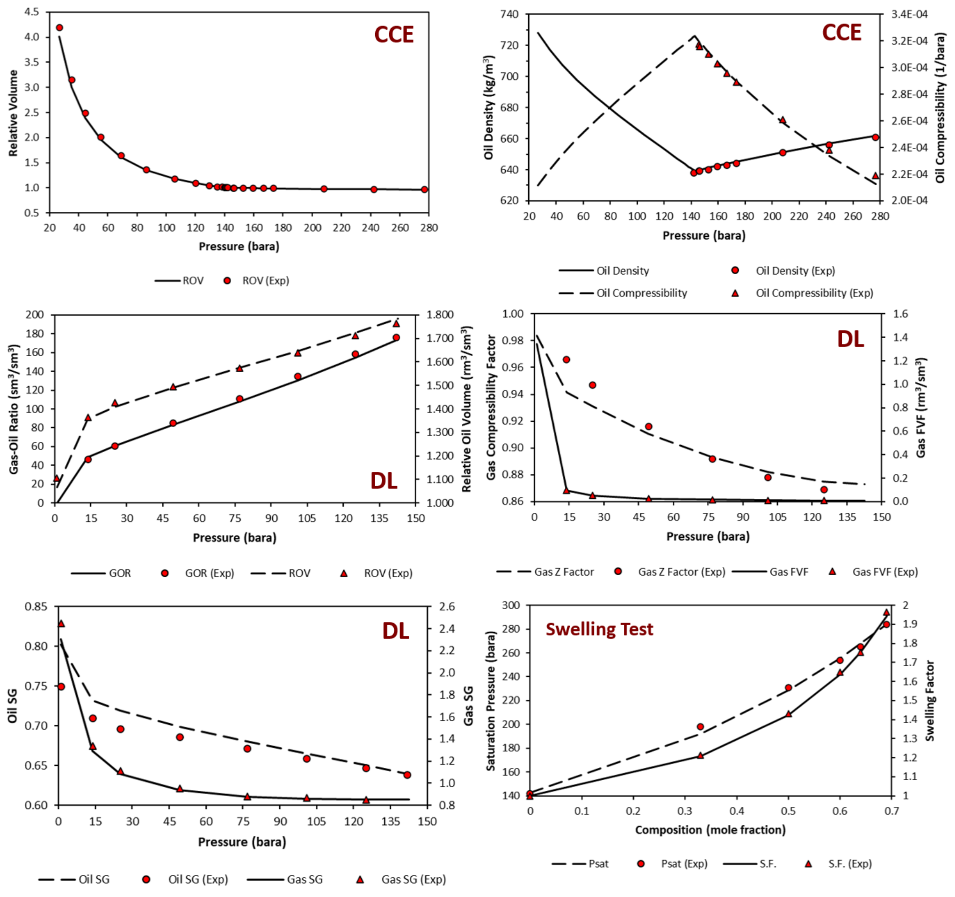

Figure 3.4: PVT tests result after regression.

Table 3.3: Single point regression result.

Single Point	EOS	Exp
P_{sat} (bara)	142.7	142.0
MMP (bara)	176.6	178.0
T_{res} (K)	120.85	120.85
Oil FVF (rm^3/sm^3)	1.535	1.551
Oil Density	0.8135	0.8139
GOR (sm^3/sm^3)	130.30	133.07

Table 3.3 shows the single-point regression results of EOS tuning. The reservoir condition of the fluid model is 287 bar and 394 K. This condition is applied as initial condition, and the reservoir will be depleted before the EOR study takes place.

3.2.2 Simple Reservoir Model - Egg Model

Taken from Jansen et al. (2014), this model is a synthetic reservoir model with 101 realizations of a three-dimensional channelized reservoir. Six out of 101 are shown in **Figure 3.5**. This model was made to test the available reservoir simulators to compare their output. With a 60 x 60 x 7 structure, this model consists of 18,553 active grid cells with no faults implemented.

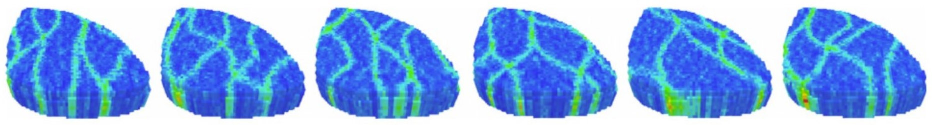

Figure 3.5: Six random realizations of Egg Model (Jansen et al., 2014).

As this model was used to see the performance difference between four simulators, the dimension of the grid is considered small. The model is too small compared to the complex model in terms of volume. For that, the grid dimension was altered by multiplying them by 3.125 (from 8 x 8 x 4 m to 25 x 25 x 12.5 m). This increases the volume to 30.5 times the original volume. The initial oil in place is now 16.5 million sm^3 after initializing the fluid model.

The leftmost permeability distribution in **Figure 3.5** is the realization being used in this study. Relative permeability curve shown in **Figure 3.2** and Newmann rock compaction is applied in this study. The model was initialized with the description from the fluid model (320 bar and 394 K). The number of wells is reduced to match the total wells in the complex model, where the chosen well positioning is shown in **Figure 3.6**.

After the modification and the initialization process, the model will then be depleted until the time CO_2-EOR is implemented. This will be described in the following subsection, as the same depletion scenario is applied to the complex model (Gullfaks).

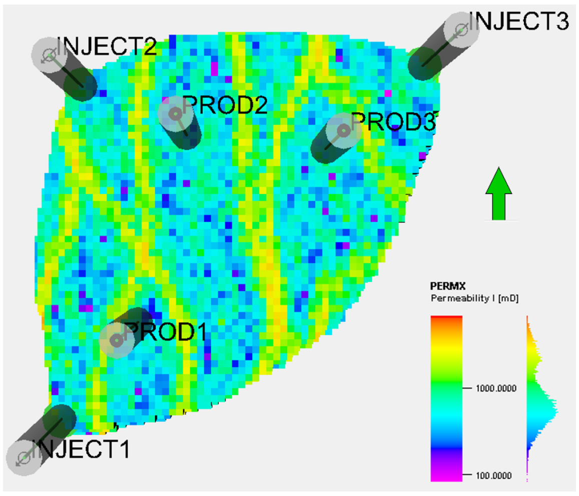

Figure 3.6: Egg model after alteration.

3.2.3 Complex Reservoir Model - Gullfaks K1/K2 Model

This model was taken from Shpak (2013), where the segmentation of the entire Gullfaks field was studied to analyze the performance of waterflooding and WAG coupled with well placement optimization. The optimization study showed higher increase in production when WAG is applied to the field compared with waterflooding. Three injector wells and three producer wells were used in the study.

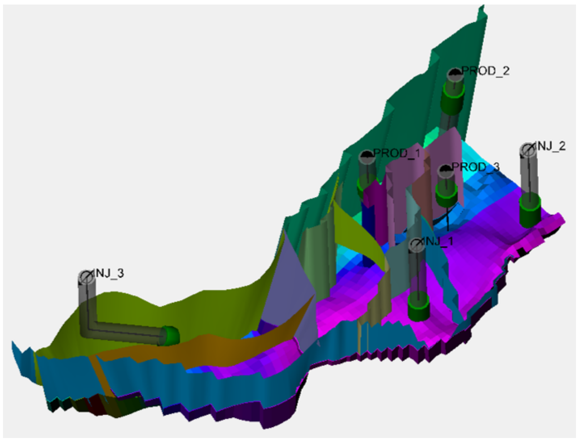

Figure 3.7: Gullfaks faults position.

Figure 3.7 shows the location of 14 faults modeled in the study, and **Figure 3.8** shows the distribution of porosity and permeability of the Gullfaks model. As no compositional fluid model is available for the Gullfaks model and due to lack of advanced PVT test data for

CO_2-EOR, the fluid model is replaced with the model described in the previous subsection. This changes the oil in place; where after initialization, the model has 17.2 million sm^3 oil in place. The reservoir is undersaturated initially.

Other than the fluid model, the Gullfaks model was constructed with relative permeability end-point values distributed heterogeneously throughout the reservoir. This increases the complexity of the model. For simplification, it was replaced with the sand preset relative permeability curve. Rock compressibility was also replaced with the Newmann compaction function. This is then followed by the fluid model initialization (320 bar and 120.85°C).

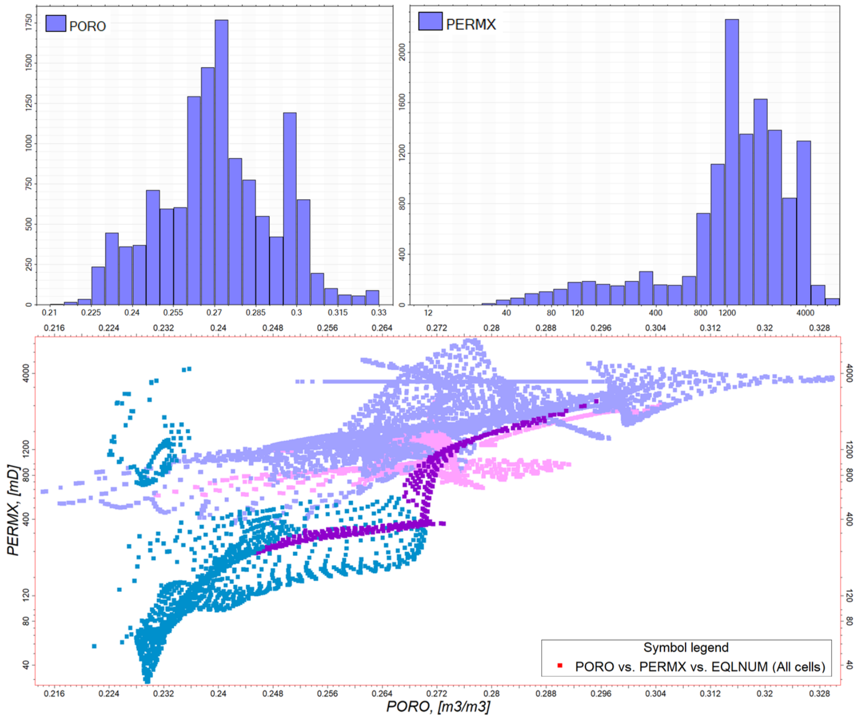

Figure 3.8: Gullfaks porosity, permeability and their crossplot in equilibrium number.

The well positions in this study are close to the study performed by Shpak (2013), yet the well locations are changed. The new positions are chosen based on the oil bank observed at initial condition. The perforation was also altered for all wells. All injector wells are perforated at the bottom of the reservoir, while the production wells are perforated at the uppermost reservoir layers. The perforation zone can be seen in **Figure 3.9**.

All relative permeability end-point values distributed throughout the model are deleted in this study, as new relative permeability relationships were applied. Other properties such as transmissibility multipliers, well connection factor, and net-to-gross (NTG) are kept. Before performing the EOR study, this model will be depleted to mimic the primary depletion in natural conditions, which will be explained in the following subsection.

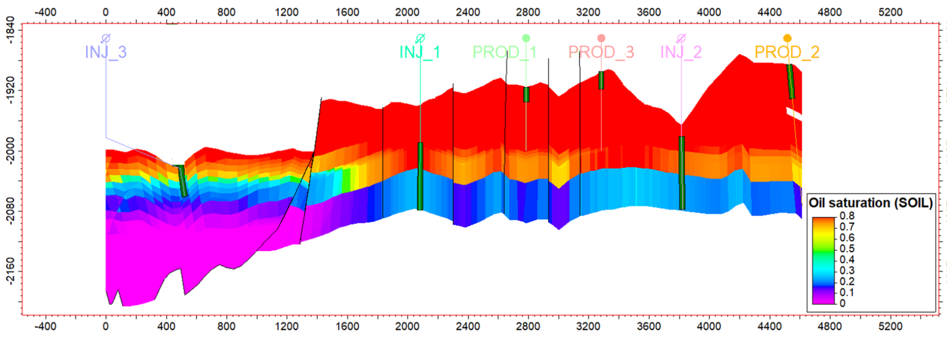

Figure 3.9: Gullfaks well cross section map.

3.2.4 Depletion Scenario

Both models now have about the same initial oil in place after the alteration (16.5 million sm^3 for the Egg model and 17.2 million sm^3 for Gullfaks). Both models will be depleted for eight years, supported with water injection to keep the reservoirs undersaturated. Equal injection rate and production rate limits are applied to both models. All wells are active (producing and injecting) starting in 2013 and continue to operate until 2021. **Table 3.4** shows the depletion study details and results obtained from each model.

Table 3.4: Depletion study details and results.

Parameter	Egg Model	Gullfaks Model
Injection rate	2000 sm^3/day/well	
Production rate limit	2000 sm^3/day/well	
Start of depletion	January 1, 2013	
End of depletion	January 1, 2021	
STOOIP	16.55 Msm^3	17.24 Msm^3
STOIP (2021)	8.94 Msm^3	8.90 Msm^3
RF	46.0 %	48.4 %
P_{avg} (2021)	254.9 bar	264.3 bar

Based on the results, both reservoirs deplete almost half of their initial oil in place. These results will be used as the initial condition for the CO_2-WAG study. Knowing the oil distribution for each model is important. **Figure 3.10** and **Figure 3.11** show the distribution of oil for each model after depletion study, plotted in pore volume multiplied by oil saturation function. No optimization study was performed on the models for depletion.

The oil distribution of the Egg Model shows an unrecovered oil bank on the southeast part of the model. Looking at the oil distribution, it can be concluded that the injected water already reached the production wells. Evaluating Gullfaks, we can see that there is still an oil bank near the second producer well. Both reservoirs can maintain the reservoir pressure over the MMP (176.6 bar). Hence, the miscibility process can be ensured to happen when CO_2 injection starts.

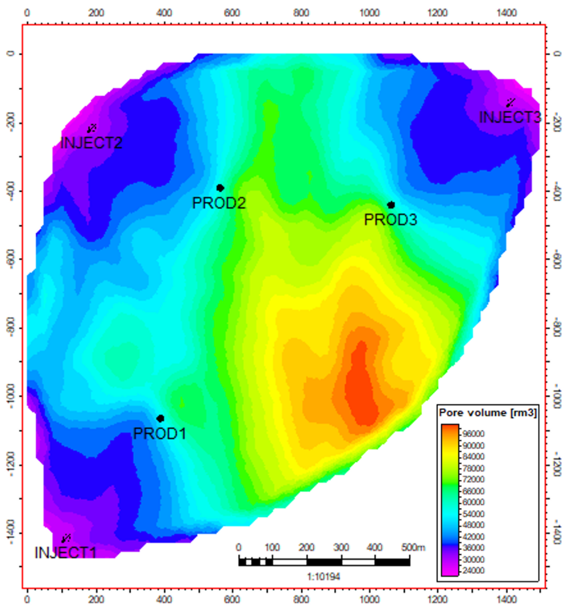

Figure 3.10: Egg Model oil distribution map after depletion scenario.

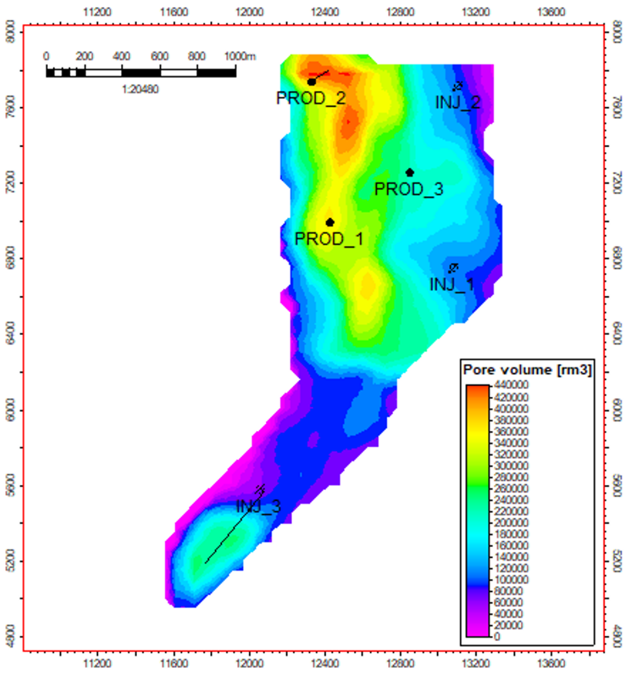

Figure 3.11: Gullfaks Model oil distribution map after depletion scenario.

Before performing a CO_2-WAG study for both reservoirs, we need to test the CO_2-WAG performance. The screening was performed based on Al Adasani and Bai (2011b) for both reservoirs. It can be seen that based on the technical screening criteria shown in **Table 3.5**, both reservoirs passed the screening. No economic screening was performed as this study will not be focusing on that part.

Table 3.5: Reservoir properties and Aladasani technical screening.

	Al Adasani and Bai (2011b)	**Egg Model**	**Gullfaks**
Oil Gravity	28 - 45	35	35
Viscosity (cp)	0 - 35	0.98	0.98
Porosity (%)	3 - 37	20	27
Oil Saturation	15 - 89	43	32
Formation Type	Sandstone / Carbonate	Sandstone	Sandstone
Permeability (mD)	1.5 - 4500	1137	1752
Net Thickness	[Wide Range]	87.5	204
Depth (ft)	1500 - 13365	5905	5905
Temperature (°F)	82 - 250	249.53	249.53

Test runs were also performed to test the performance of both reservoirs. Waterflooding study was performed to see the field performance compared to CO_2-WAG. **Figure 3.12** shows the results for both reservoirs. We can see a higher recovery when CO_2-WAG is applied (3 months half-cycle, 6000 sm^3/day field water injection rate, and 0.9 Msm^3/day field gas injection rate) compared to waterflooding (3000 sm^3/day field injection rate).

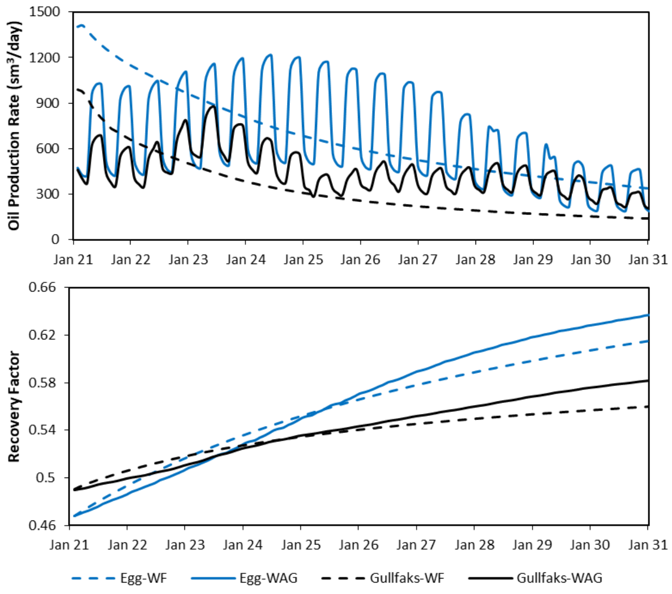

Figure 3.12: Reservoir performance comparison between waterflood and CO_2-WAG.

As both models show potential in recovery increase for CO_2-WAG EOR, an optimization study can now be conducted. The details of the objective functions, constraints, and the optimization algorithm used are explained in the next subchapters.

3.3 Optimization Problem

In a previous study (Matthew, 2020) optimization study was performed for CO_2 flooding design by calculating the total oil produced, represented by field oil produced total (FOPT). The results are tabulated in **Table 3.6**. In the current study, sequestration of CO_2 will also be observed, focusing only to the amount of CO_2 stored. CO_2-EOR study can be executed in line with CO_2 storage. Hence, in this study, oil production optimization will be coupled with CO_2 stored in the reservoir.

Table 3.6: Optimization results from a previous study (Matthew, 2020)

Parameter	Optimum Value	Range Value
Injection Rate (sm^3/day)	2	1 - 2
PV Injected (PV)	1.9976	1 - 2
Start Time (years)	3.75	2.5, 3.75, 5
CO_2 Composition (%)	0.9977	0.8 - 1
C_1 Composition (%)	0.0008	0 - 0.2
C_2 Composition (%)	0.0015	0 - 0.2
FOPT (sm^3)	3,014,442	

The same optimization problem will be applied to both reservoir models. The proxy model is the central part of this study. The optimization problem will be solved using the proxy model explicitly built for this purpose. This study will be focusing more on the technical study of the CO_2-WAG project. Hence, no economic evaluation/constraints will be applied.

3.3.1 Objective Functions

Two results that will be observed are the oil production and CO_2 stored in the reservoir. Based on that, two objective functions need to be defined. Before starting the formulation, it can be seen that the previous optimization study (Matthew, 2020) converged to the extreme conditions (pure CO_2 injected, injected from the start of depletion, highest injection rate, and largest slug size). This needs to be avoided in this study.

Responding to that information, the objection functions of this optimization study are:

1. Maximize total oil produced
   The total oil produced from the reservoir is the main concern, as the project's profit will be generated from this result. This can be reflected from the total oil produced or the multiplication of oil production rate with the duration of production. As no economic constraints are applied, this might lead to the same optimization result (extreme condition) we got before, so the other objective function is,

2. Maximize total CO_2 stored in the reservoir
   The CO_2 storage performance can be seen from the total CO_2 injected and total CO_2 produced. This will make the solution space more complex, as the total CO_2 produced will be minimized, yet more CO_2 injection is needed to maximize this. In alignment with that, a higher injection rate will yield a higher oil production rate.

An optimization study, especially for reservoir design, needs a period or targeted volume to be injected/produced as the study target. This study is focused on a limited project time, which is more common in real applications. For this reason, ten years is the limit of the CO2-WAG project. Based on that, the total slug size injected will not be used as a parameter in this study, and the optimization study will be performed to optimize the project for ten years.

3.3.2 Parameters and Constraints

Parameters that will be studied are as follow:

1. Half-cycle length
   Value range : 3 - 12 months

   We will only consider some values in this range for this parameter, 3, 6, 9 and 12 months. For simplification, the half-cycle for both injection stages will be the same. Hence, changes in this parameter will not affect the WAG ratio of the project if the injection rates are kept constant.

2. Field gas injection rate
   Value range : 1 - 2 Msm^3/day

   Along with the water injection rate, this parameter will affect the WAG ratio of the project. The values were selected after considering the injection rate from the previous CO2 injection study (Matthew, 2020). This parameter will be distributed equally among all existing injection wells.

3. Field water injection rate
   Value range : 3000 - 9000 sm^3/day

   Both fields were depleted using the same scenario, where water injection was applied to maintain the pressure. Assuming that the fields have existing facilities ready for water injection, the range is taken near the previous water injection rate (6000 sm^3/d). This parameter will be distributed equally among all injection wells.

We can formulate the objective functions and study parameters into mathematical expression, as shown below:

$$
\begin{aligned}
\text{maximize} \quad & \text{FOPT } (HC, q_g, q_w) \\
& \text{CO}_2 \text{ Stored } (HC, q_g, q_w) \\
\text{subject to} \quad & HC = [3, 6, 9, 12] \text{ months} \\
& 1 \text{ Msm}^3/\text{d} \leq q_g \leq 2 \text{ Msm}^3/\text{d} \\
& 3000 \text{ sm}^3/\text{d} \leq q_w \leq 9000 \text{ sm}^3/\text{d}
\end{aligned}
$$

3.3.3 Optimization Algorithm

Only NSGA-II was used to solve the optimization problem in this study. For doing this, the PyMoo package available in Python is used. No optimization of the hyperparameters in the optimization algorithm was performed here.

3.4 Proxy Model Development

Many machine learning algorithms or deep learning models can be employed to build a proxy model, such as ANN-RNN, ANN-DNN, PR, KG, and TSP. However, we will focus on artificial neural network (ANN), especially deep neural network (DNN) structure for this study. More into the DNN, optimization of the topology will be performed to build a better proxy.

Here, rather than using MATLAB like in the previous study (Matthew, 2020), we performed all the studies using Python as the programming language. This is due to its versatility, being easy-to-learn and work with, and open-source programming language with a vibrant community. The details, structure of the scripts, and how it was built are described in the next chapter.

Chapter 4

Proxy Model and Optimization - Building the Foundation

As we will make two different proxies with the same study objective, it will be better to construct the main script before performing all the studies. The idea is to get a grasp of all tools that we will be using for this study. In this chapter, the process of building the proxy, obtaining the needed inputs, and translating the optimization problems to programming language will be described. The main process was described in sequence in the workflow of **Figure 3.1**.

All reservoir model-related studies, including simulation and data extraction, are performed using E300 with PETREL as the interface. For the proxy modeling and optimization study, Python is used. In alignment with that, some of the processes will be performed using Excel, which will be mentioned later. Many packages used for this study are in Python, which will be mentioned in each segment of this chapter.

4.1 Optimization Problem Formulation

The equation of the optimization problem is shown below. The whole process of building the proxy will be started from here as a proxy model is case-specific.

$$\begin{aligned}
\text{maximize} \quad & \text{FOPT}\ (HC, q_g, q_w) \\
& \text{CO}_2\ \text{Stored}\ (HC, q_g, q_w) \\
\text{subject to} \quad & HC = [3, 6, 9, 12]\ \text{months} \\
& 1\ \text{Msm}^3/\text{d} \leq q_g \leq 2\ \text{Msm}^3/\text{d} \\
& 3000\ \text{sm}^3/\text{d} \leq q_w \leq 9000\ \text{sm}^3/\text{d}
\end{aligned}$$

First of all, based on the equation, we know that two functions are needed, total oil produced (FOPT) and CO_2 stored/sequestrated. For total oil produced, we can collect data easily as one of the outputs. For CO_2 stored in the reservoir, it will be approached using the equation shown below.

$$\text{CO}_2\text{ Stored } = \text{Total CO}_2\text{ Injected} - \text{Total CO}_2\text{ Produced}$$

From the simulator, it is known that we can get the results of CO_2 produced from one of the outputs (FCO_2PT), yet it is reported in kg-mole units. Hence, a simple conversion is used where 1 Msm^3 is equal to 44,122 kg-mole of CO_2 to change the unit system. This can be formulated as shown below.

$$\text{CO}_2\text{ Stored } = q_g * t_{g,inject} - \text{FCO}_2\text{PT}(HC, q_g, q_w)$$

As this study is a CO_2-WAG study, hence in a cycle, two types of fluid will be injected. Here, $t_{g,inject}$ refers to the total time of gas injection. The gas injection rate is being set to be constant. Based on the simulations, the highest gas injection rate will not reach the pressure limit of the injector BHP. So, this approach is valid for this study.

FCO_2PT is the total amount of CO_2 produced at the surface. FCO_2PT and FOPT can be obtained using the rate functions. Based on this, a proxy of FOPR (field oil production rate) and FCO_2PR (field CO_2 production rate) will be constructed. Both functions will be evaluated on a field scale rather than on a well scale since we will only perform a field optimization study.

Then, we can rewrite the formula of the optimization problem, as shown below.

$$\begin{array}{rl} \text{maximize} & \text{FOPT } = ts\ .\ \text{FOPR}(HC, q_g, q_w) \\ & \text{CO}_2\text{ Stored } = q_g\ .\ t_{g,inject} - \text{FCO}_2\text{PR}(HC, q_g, q_w)\ .\ ts \\ \text{subject to} & HC = [3, 6, 9, 12] \text{ months} \\ & 1\text{ Msm}^3/\text{d} \leq q_g \leq 2\text{ Msm}^3/\text{d} \\ & 3000\text{ sm}^3/\text{d} \leq q_w \leq 9000\text{ sm}^3/\text{d} \end{array}$$

To find the $t_{g,inject}$, we can put it in terms of HC, where we calculate the total cycle passed for ten years. As we have only four half-cycle values to be observed, we only have two possible values of $t_{g,inject}$. These values are 62 months for 3, 6, and 12 months, and 63 months for 9 months half-cycle. This needs to be translated into days by multiplying with 30 for unit conversion.

All variables shown in italic are the needed inputs of the functions. All of them are defined except ts which refers to the timestep. As mentioned before, the project is time-based, which means that we have the same total time step for each run, i.e. ten years. For this, the monthly time step (equal to 30 days) will be used. We will approach ten years with

122 months (3660 days). Finally, the final form of our optimization study is shown in **Equation 4.1**.

$$\begin{aligned}
\text{maximize} \quad & \text{FOPT} = \sum_{n=1}^{122} (\text{FOPR}(HC, q_g, q_w))_n \, . \, 30 \\
& \text{CO}_2 \text{ Stored} = q_g \, . \, t_{g,inject} - \sum_{n=1}^{122} (\text{FCO}_2\text{PR}(HC, q_g, q_w))_n \, . \, 30 \\
\text{subject to} \quad & HC = [3, 6, 9, 12] \text{ months} \\
& 1 \text{ Msm}^3/\text{d} \leq q_g \leq 2 \text{ Msm}^3/\text{d} \\
& 3000 \text{ sm}^3/\text{d} \leq q_w \leq 9000 \text{ sm}^3/\text{d}
\end{aligned} \tag{4.1}$$

Following this, we know what function we need to make. Now, we can prepare and extract the data from our reservoir simulator runs. The details of how this optimization problem will be translated into programming language will be described in the last section.

4.2 Reservoir Model Preparation

Both models have been modified already from their original forms. Before performing the study, the models were initialized as described in **Section 3.2.**. CO_2-WAG study will be performed after depletion ends by using the created RESTART file to reduce the run time. The results from the depletion phase will not affect our proxy and optimization study.

It will be helpful to identify what type of proxy model we will make beforehand by referring to its element volume. Based on the defined study objective and optimization problem, we can see that the proxy we need will be **field-based** proxy. We refer to **Table 2.2** as the reference for identifying which data that needs to be extracted from the reservoir simulator. It was performed to avoid reruns due to the missing data needed yet not reported for the proxy.

Based on the formulation of the optimization problem, two simulation outputs will be learned by our proxy, FOPR, and FCO_2PR, which is in line with **Table 2.2**. These variables are available as output results of a simulation run. These can be modified by defining the simulation case $\rightarrow$ Section Results $\rightarrow$ Report frequency $\rightarrow$ Report timesteps only in PETREL. Worth knowing that in the Development Strategy section, the timesteps and half-cycle need to be defined in days.

Other than that, to reduce the amount of memory needed for each run, unwanted simulation outputs are all removed. It hugely helps the amount of memory usage. For one case of the Gullfaks Model, it takes 185 MB of space if we report all outputs that were defined initially by the software. Modifying them only to report the needed output takes only 79-87 MB of space. This reduces the total storage usage by more than half. As we will run more than ten cases, this reduction hugely helps the simulation process.

4.2.1 Data Extraction

Runs are performed for both reservoir models. For easier proxy modeling, constant time step reporting is used while running the models. We will get a staircase production profile for all needed outputs, which the proxy will learn. Such results are shown in **Figure 4.1**. A test run was performed using both models with the study parameters based on the optimization study. The run statistics for one run is shown in **Table 4.1**. We can see that a lot of space and time is needed to run the simulator.

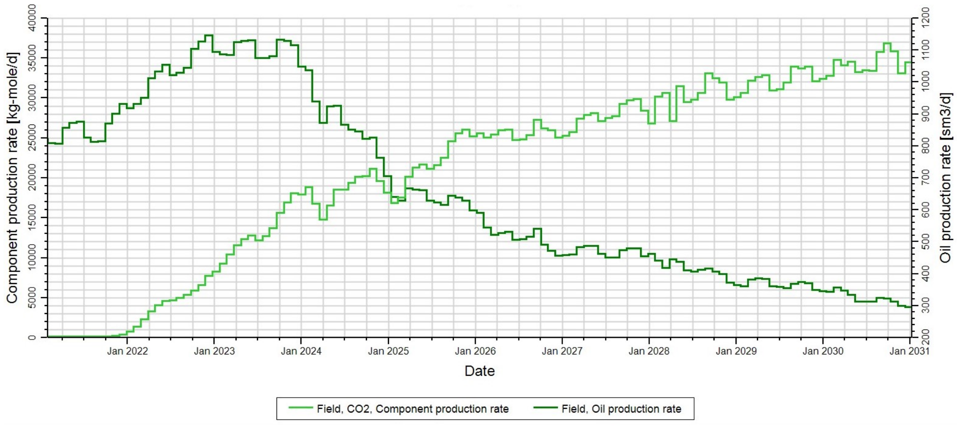

Figure 4.1: Staircase rate behavior from one Egg Model run.

Table 4.1: Run statistics for each reservoir model.

Parameter	Egg Model	Gullfaks Model
Run Time (s)	565	523
Memory Usage (MB)	97	85

4.2.2 Maintaining the Numerical Stability

It is vital to get high-quality data for our proxy learning material. Especially later in a more complex model, the numerical error might mask our simulator's physical behavior. To ensure this process, an additional keyword is added to our SCHEDULE section, as shown below.

```
TUNING
0.1 1* 0.05 0.1 6* /
11* /
24 1* 200 7* /
```

The first line shows the maximum length of the next timestep and the minimum timestep length. The second line shows the truncation and convergence controls, which are kept the same. Moreover, the last line is the iteration control. We tweaked the maximum number of Newton iterations and linear iterations for each timestep.

The difference between non-tuned and tuned runs can be seen in **Figure 4.2**. It might seem insignificant by bare eyes, but the maximum difference between the tuned (red) and non-tuned (green) is 38 sm^3 (January 2025), which is big (10% difference). The side effect of applying this control is that a longer time is needed to run a case. An increase of one minute per case was observed in each model after applying this keyword.

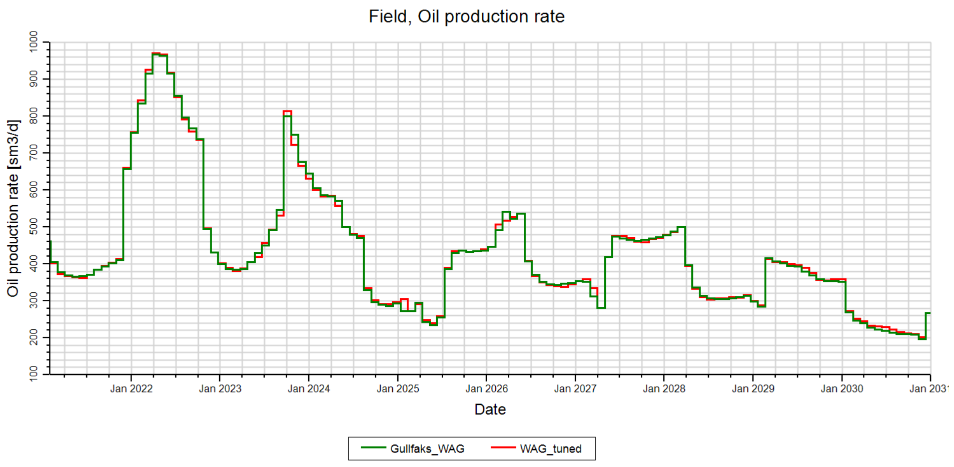

Figure 4.2: Non-tuned and tuned runs rate performance.

All data needed to make the proxy is then exported from PETREL to an Excel file. Then, this excel file will be processed to be used for building the proxy.

4.3 Proxy Building Preparation

The previous steps we performed are essential to be done first. The proxy can be built **only if** we know what proxy to be built and what data to use. Failure to perform in sequence will make an infinite loop between these three sections (problem formulation, model preparation and proxy building) until we figure the first two sections. An overview that shows the outline of the study, performed with the available software and programming language, is shown in **Figure 4.3**. In this subsection, these processes will be described in detail.

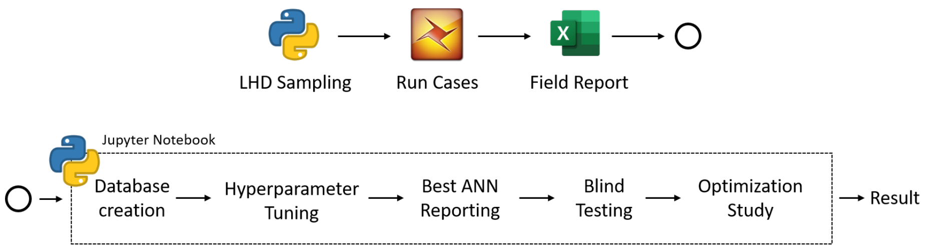

Figure 4.3: Workflow of proxy building, optimization study, and software involved in this study.

4.3.1 Proxy Model Rough Sketch

This part is in line with what we have explained in **Section 4.2**. We ought to know what input-output combination we will have in our proxy. The input of the proxy will be the optimization parameters and our rate data, which are extracted in each timestep. The output that we expect to have is the rates in every timestep (30 days).

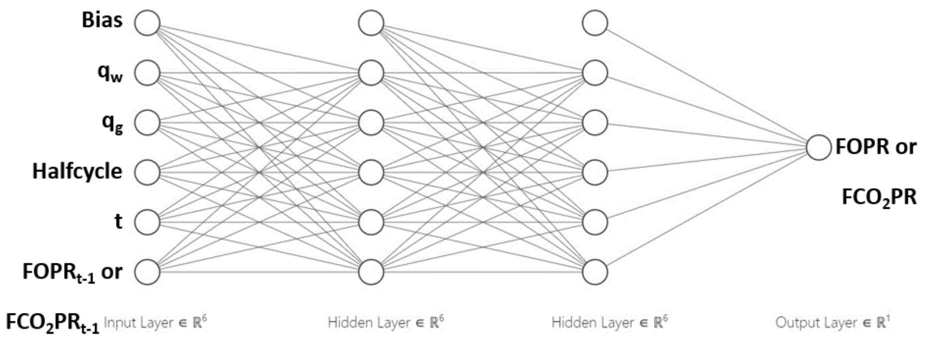
Bias
q_w
q_g
Halfcycle
t
$FOPR_{t-1}$ or
FCO_2PR_{t-1}
Input Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^6$
Output Layer ∈ $\mathbb{R}^1$
FOPR or
FCO_2PR

Figure 4.4: Rough sketch of the proxy model.

For better results, the rate in the previous timestep will also be used as an input, based on experience from the previous study (Matthew, 2020). **Figure 4.4** shows the predicted network structure that will be constructed. Note that the network topology shown in the figure is only an assumption. This is just a rough sketch to start the study. Later, if it does not work properly, we can adjust the input-output combinations, the topology, and any hyperparameter of DNN.

This rough sketch may be valid for most field-based optimization study problems. Different rough sketches may apply for different kinds of studies, with different types of volume element proxy. For well-based and grid-based proxy, Gholami (2014) and Amini (2015) results can be used as reference. More studies need to be performed for these smaller element volumes as we will have more input.

4.3.2 LHS Preparation

We performed the sampling using LHS in this study. The sampling can be considered as a leveled sampling. It was performed based on the previous study results (Matthew, 2020) and several success stories on sampling for the WAG study by for example Nait Amar et al. (2018). Two levels of sampling that were performed in this study are as follows.

1. Half-cycle length
   Sampling was performed for 3, 6, 9, and 12 half-cycle lengths. This was performed because we discretized our half-cycle length, so other values are not counted as part of our sampling space. Performing this means that the proxy we make will not predict the CO_2-WAG behavior out of this half-cycle length correctly (not even 3.00001 months).

2. Gas injection rate and water injection rate
   Gas injection rates are sampled for each possible half-cycle length equally since the probability for half-cycle length is equal for each possible value. More samples can be added if the proxy cannot learn from the given number of samples. The distribution of this parameter probability is uniform. All parameters are not dominating the others. Hence usual LHS can be performed.

Based on this, we will have four two-dimensional space sampling performed with pyDOE package in Python. The whole possible solutions must be represented, especially the extreme points. As we do not know the optimum condition of our optimization problem, the edges of the sampling space also need to be covered. These values can be approached just like the number of edges of an n-dimensional hypercube. As this study involves three parameters, 8 (2^3) points that can be created from the combination of extreme points of each variable need to be added to the sampling, outside the results we obtained from LHS. These edges are forced to be our training dataset.

We need to prepare a short script other than LHS sampling to ensure the quality of our sampling method. A script that was made to perform the leveled sampling for this study with the addition of the sampling space edges is shown below.

```python
import numpy as np
from pyDOE import *

# input
variables = 3
vlim = np.array([[3,12], [1,2], [3000,9000]])
                # the 1st column is the 1st level
N = 18 # number of samples for the 2nd level LHS

# create the array structure
sampling = np.zeros([1,variables])

# leveled LHS sampling
for half_length in range(1,5):
    s = lhs(2, samples=N, criterion ='center')

    #converting to the value ranges
    s[:,0] = vlim[1,0]+s[:,0]*(vlim[1,1]-vlim[1,0])
    s[:,1] = vlim[2,0]+s[:,1]*(vlim[2,1]-vlim[2,0])
    t = np.full((N,1), half_length*3)
    temp = np.append(t, s, axis = 1)
    sampling = np.concatenate((sampling, temp), axis = 0)

# deleting the array structure
sampling = np.delete(sampling, 0, 0)

# add extreme points
extreme = np.array([[vlim[0,0], vlim[1,0], vlim[2,0]], \
     [vlim[0,1], vlim[1,0], vlim[2,0]], [vlim[0,0], vlim[1,1], vlim[2,0]],\
     [vlim[0,0], vlim[1,0], vlim[2,1]], [vlim[0,1], vlim[1,1], vlim[2,0]],\
     [vlim[0,1], vlim[1,0], vlim[2,1]], [vlim[0,0], vlim[1,1], vlim[2,1]],\
     [vlim[0,1], vlim[1,1], vlim[2,1]]])
sampling = np.concatenate((sampling, extreme), axis = 0)

# save to csv file
a_file = open("samplingdata.csv", "w")
np.savetxt(a_file, sampling, delimiter=',')
a_file.close()
```

4.3.3 Data Preparation

After the sampling is performed and the run results are obtained, the data must be prepared before being used to construct the proxy. The structure obtained from PETREL is then arranged for each run case. Manual adjustment is performed to manage the obtained data. The structure of the main database as the result of this process is shown in **Figure 4.5**.

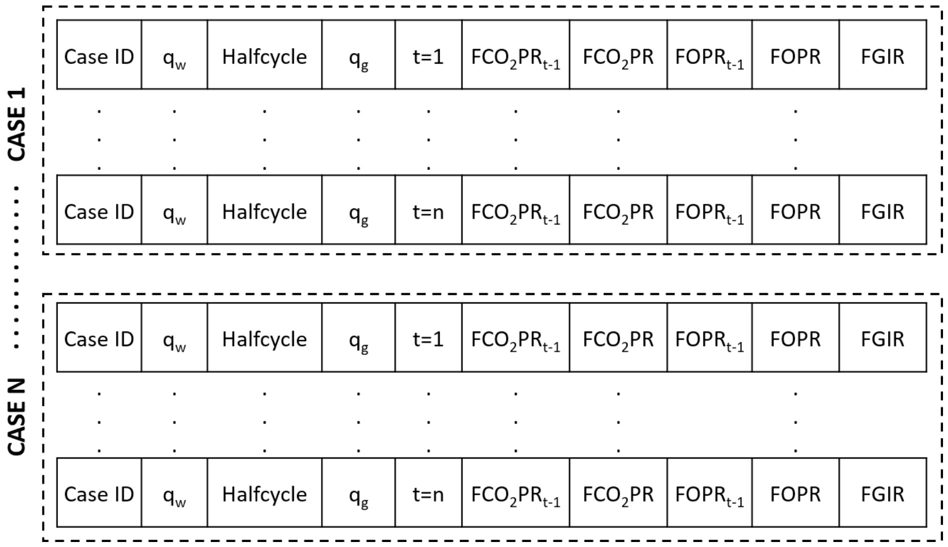

Figure 4.5: Main database structure

We will have 122 data for each run that can be used to train the algorithm, as the total timesteps are 122 months for each run. Field gas injection rate (FGIR) is added to the database as an identifier of fluid being injected at the current timestep. The structure is adapted from the needed input-output of the proxy.

We can see different magnitudes from all data in the database, such as in a million for gas injection rate, thousands for water injection rate, and tens for the timesteps. The process of ANN learning will be a bit harder when using this data. The worst thing that may happen is that the network converges to local optima when updating the weights and biases. To solve this, linear normalization is performed for all data we have using **Equation 4.2**.

$$x_{\text{Norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{4.2}$$

This maximum-minimum normalization rescales our inputs into 0 to 1. This will help the training process of our network and reduce the possibility of convergence failure. A special treatment is applied for the FCO_2PR_{t-1}, FCO_2PR, $FOPR_{t-1}$ and FOPR. Maximum and minimum values are taken from the combination of the current and previous timestep population. This is performed because they are generated from the same dataset, where $t-1$ counts the timestep 0 value and t counts the final timestep value (122).

4.3.4 Building the Proxy

After preparing the database, the proxy can be developed. As we have a small number of input variables, principal component analysis (PCA) does not need to be performed (refer to **Figure 3.1**). For the smaller level of the volume element, either PCA or KPI (Key Performance Indicator) can be performed. It can reduce the amount of input based on its influence on the proxy and may avoid the curse of dimensionality.

The first thing that we need to do, as shown in **Figure 4.4**, is to perform the hyperparameter tuning of the neural network. All hyperparameters and their values are listed in **Table 4.2**. This study only focuses on the learning rate, number of layers, and neurons/nodes. For easier call in the whole loop for the study, the script below generates an ANN.

```python
def ANN_model(features, target, learn_rate, layers, nodes):
    """
    Input Data:
    features   : Input data for the ANN (numpy array)
    target     : Output of the ANN (numpy array)
    learn_rate : Learning-rate for the optimizer.
    layers     : Number of dense layers.
    nodes      : Number of nodes in each dense layer.
    """

    # Define model
    model = Sequential()
    if layers == 1:
        # Input layer
        model.add(Dense(nodes, activation='relu',
                        input_dim=features.shape[1]))
    else:
        model.add(Dense(nodes, activation='relu',
                        input_dim=features.shape[1]))
        for i in range(layers-1):
            model.add(Dense(nodes, activation='relu'))

    # Add output layer, 1 nodes
    model.add(Dense(1, activation='linear'))

    adam = Adam(lr=learn_rate)
    model.compile(optimizer=adam, loss='mse',
                  metrics=['mean_absolute_percentage_error',
                           'RootMeanSquaredError'])

    return model
```

Table 4.2: Hyperparameters in ANN design with Python.

Parameter	Value	Notes
Learning rate	10^{-6}-10^{-2}	Learning rate of the optimizer function
Number of layers	1 - 3	Number of hidden layers
Number of neurons	2 - 20	Number of nodes in each hidden layer
Activation function	Relu	Activation function on each node
Optimizer	Adam	Function to optimize the NN
Batch size	64	Data points that pass through NN every step
Dropout	-	Probability where nodes are randomly disconnected during training
Epoch	100	Amount of times to go through our training data

Then, the ANN topology optimization study was performed using this function.

```python
# Function
@use_named_args(dimensions=dimensions)
def ANN_study(learn_rate, layers, nodes):
    """
    Hyper-parameters:
    val_features: Input of the ANN for validation (np.array)
    val_target  : Output of the ANN for validation (np.array)
    """

    # Print the hyper-parameters.
    print('Learning rate: {0:.1e}'.format(learn_rate))
    print('Dense layers:', layers)
    print('Nodes:', nodes)

    model = ANN_model(features=features,
                      target=target,
                      learn_rate=learn_rate,
                      layers=layers,
                      nodes=nodes)

    # Use Keras to train the model.
    patience = 20
    es = EarlyStopping(monitor='loss', mode='min', verbose=1,
                      patience=patience, restore_best_weights=True)
    history = model.fit(features, target, verbose=0, epochs=100, batch_size = 64,
                validation_data=(val_features, val_target), callbacks=[es])

    # Get the regression validation loss
    # after the last training-epoch.
    if es.stopped_epoch == 0:
        loss = history.history['loss'][-1]
    else:
        loss = history.history['loss'][es.stopped_epoch-patience]

    # Print the validation loss.
    print()
    print('loss:', loss)
    print()

    # Save the model if it improves on the best-found performance.
    global best_loss

    # If the validation loss of the saved model is improved ...
    if loss < best_loss:
        # Save the new model to harddisk.
        model.save(path_best_model)

        # Update the regression accuracy.
        best_loss = loss

        #save the history of the best model
        ANN_performance = history.history
        temp = json.dumps(ANN_performance)
        f = open(path_best_model_performance,"w")
        f.write(temp)
        f.close()

    # Delete the Keras model with these hyper-parameters from memory.
    # And clear the Keras session
    del model
    K.clear_session()

    # NOTE: Scikit-optimize does minimization
    return loss
```

The script below was executed to do the hyperparameter optimization study, and it will report the results. The reported results will be the optimum learning rate, number of hidden layers, and nodes for each hidden layer. Training loss will be used as the objective function to be minimized. We do not need to be concerned about the overfitting issues in this stage, as later, the model will relearn.

```python
search_result = gp_minimize(func=ANN_study,
                            dimensions=dimensions,
                            acq_func='EI', # Expected Improvement.
                            n_calls=80,
                            x0=default_parameters)

print("""Best parameters:
- learning rate = %.6f
- number of hidden layers = %d
- number of nodes = %d
- best loss = %.6f""" % (search_result.x[0], search_result.x[1],
                            search_result.x[2], best_loss))
```

After we obtained the optimum topology, we will make a new model with the results. However, we will use a higher epoch, from 100 to 1500. Rather than using 1500 epochs during the hyperparameter study, we only perform this to the best topology to reduce the running time for building a proxy. The script is attached below.

```python
learn_rate = search_result.x[0]
layers = search_result.x[1]
nodes = search_result.x[2]

ANN_model(features, target, learn_rate, layers, nodes)

model = ANN_model(features, target, learn_rate, layers, nodes)
history = model.fit(features, target, verbose=0, epochs=1500, batch_size = 64,
                    validation_data=(val_features, val_target))

model.save(path_best_model)
```

Many error metrics can be used. For this study, MSE (mean squared error) is used as the loss function of the ANN, while R^2 and AE (absolute error) are also reported with the results. These metrics can be used as our proxy's performance indicator, as each metric has its pros and cons. All scripts shown here are small parts of the whole script. The whole script can be seen in the Appendix.

4.3.5 Proxy Output

The proxy will produce a normalized value of FOPR and FCO_2PR. Hence, denormalization needs to be performed. **Equation 4.2** will be adjusted to denormalize the results. Then, the denormalized values will be reported as our proxy results. After the proxy was constructed and the proxy's performance was confirmed to be sufficiently good, it can be used for the optimization study. A script to generate the output graph and run the proxy with all training-validation data is also written to evaluate the proxy performance (see Appendix).

For information, the processes described in this subsection are drafts made before building the proxy. The complexity of the data affects the learning process and performance of the proxy. Hence, the details of how the proxy is constructed are different for the two models. Nevertheless, the processes were performed for the same goal, to make a proxy that can behave like the simulator model. More details will be described in the next chapter.

4.4 Optimization Study Preparation

When the proxies are ready to be used, objective ought to be prepared before moving into the optimization study. In this study, FOPR was defined as $oilproxy$, while the CO_2 stored was defined as $CO2proxy$. The main inputs for both of them to work are the optimization parameters. Other proxy inputs, such as timestep and the previous timestep value, are generated inside the function. The previous timestep value will be the result that was obtained from the proxy based on its previous timestep. For the time 0, it is predefined as all runs start with the same initial time.

The whole optimization study was performed using PyMoo. As NSGA-II works as a minimization optimization tool, all objective functions are transposed to negative. This will make the maximization out of a minimization process result. Below is the main script that defines our study objective and study parameters.

```
from pymoo.model.problem import FunctionalProblem

objs = [
    lambda x : -oilproxy(x[0],x[1],x[2]),
    lambda x : -CO2proxy(x[0],x[1],x[2])
]

constr_ieq = [lambda x : x[0]%3]

functional_problem = FunctionalProblem(3,
                                       objs,
                                       constr_ieq=constr_ieq,
                                       xl=np.array([3,1,3000]),
                                       xu=np.array([12,2,9000]))
```

The parameters are half-cycle length, gas injection rate (in Msm^3/d), and water injection rate (in sm^3/d) in sequence. As we want to have the half-injection rate to be exactly 3, 6, 9, and 12, an extra constraint is defined by $constr\_ieq$. The bounded constraints that we have for each optimization parameter are defined in the functional problem. It might help to define the problem in class rather than as a function. Doing so will reduce the amount of time for performing the optimization. However, the proxy we have is complex, with many loops inside it. Hence it will be hard to define them as a class.

Outside of the constraints, some modifications need to be performed. We need to make it integer sampling (for half-cycle length), so the optimization study will converge faster.

Doing this will make the iterations work on the preferred solution space. The script is shown on the next page.

```
mask = ["int", "real", "real"]

from pymoo.factory import get_sampling, get_crossover, get_mutation
from pymoo.operators.mixed_variable_operator import MixedVariableSampling,
                    MixedVariableMutation, MixedVariableCrossover

sampling = MixedVariableSampling(mask, {
    "real": get_sampling("real_random"),
    "int": get_sampling("int_random")
})

crossover = MixedVariableCrossover(mask, {
    "real": get_crossover("real_sbx", prob=1.0, eta=3.0),
    "int": get_crossover("int_sbx", prob=1.0, eta=3.0)
})

mutation = MixedVariableMutation(mask, {
    "real": get_mutation("real_pm", eta=3.0),
    "int": get_mutation("int_pm", eta=3.0)
})
```

Worth mentioning that no study was performed for the hyperparameters in our optimization algorithm. To perform this, we must have a better understanding of how the function is built. Other than that, the population size will be kept at 40. The number of generations for one study was set to 100, where ten new offsprings will be generated in each generation. For the whole study, it will take more than 1000 proxy runs to be finished.

Following the descriptions, a script was made to perform the study, as shown below.

```
from pymoo.algorithms.nsga2 import NSGA2

algorithm = NSGA2(
    pop_size=40,
    n_offsprings=10,
    sampling=sampling,
    crossover=crossover,
    mutation=mutation,
    eliminate_duplicates=True
)

# Define termination criterion
from pymoo.factory import get_termination

termination = get_termination("n_gen", 100)

# Perform Optimization study
from pymoo.optimize import minimize

res = minimize(problem,
               algorithm,
               termination,
               seed=1,
               save_history=True,
               verbose=True)
```

The optimization study might take some time, as it depends on how long it takes to run one case with our proxy. However, it can be assured that it will take less time than the reservoir model. Assuming 1000 runs were needed, it will take 157 hours (6.5 days) for the reservoir model we have to finish the task (refer to **Table 4.1**). As we will get several Pareto optima from the optimization process, a plot can illustrate the possible combinations. The code below is made to do so.

```python
from pymoo.visualization.scatter import Scatter
plot = Scatter()
plot.add(res.F, color="red")
plot.show()
```

The whole outline of how the proxy model can be built and used to perform optimization was explained in this chapter. However, it might be a bit different for each model. In the next chapter, we will see how proxy models are developed for each reservoir model.

Chapter 5

Developing a Proxy Model for Optimization Study

In this chapter, the process of developing the proxy models is described. The first two subsections describe how the final proxies are developed. And the last subsection explains the learning process before the best proxy models are constructed.

Both models are employed with the same workflow as shown in **Figure 3.1** and as described in **Chapter 4**. Several adjustments and details of how the models are developed are described here. The final programming scripts to build and perform the optimization study are attached to this report.

5.1 Simple Model - Egg Model

The egg model is chosen due to its availability and simplicity. It is known that outside the outer boundary (initiated as a no-flow boundary), no faults are installed. The initial configuration of the injector and producer wells is inverted five-spot injection. It ensures the sweep efficiency for all layers, as the well is perforated through all layers.

The grid configuration is also very simple, where constant grid size is applied for the whole model. Also, the porosity is constant, with permeability defined as a channeling-like reservoir. Other than that, no complex multipliers (such as transmissibility or near-wellbore multipliers) and transformations were applied to the model. It can be used as a good starting point for performing the proxy model study.

To reduce the complexity, the main model is tweaked by reducing the number of wells and increasing the grid size. It was explained in **Section 3.2.2**. The goal is to make it easier to compare with the Gullfaks model regarding the reservoir volume and production rates.

5.1.1 Design of Experiments

Sampling is performed based on the parameters studied for the optimization study. As explained, we performed leveled sampling. Water injection rates and gas injection rates were sampled for all half-cycle possibilities of the study. The sampling for each half-cycle are shown in **Figure 5.1**.

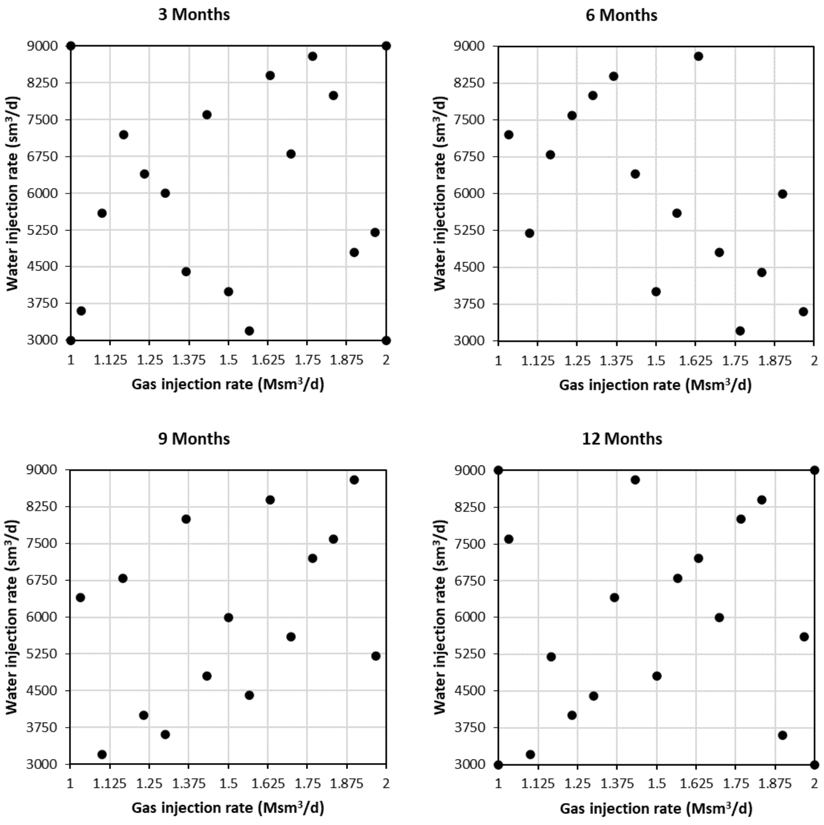

Figure 5.1: LHS performed for Egg model.

In total, 68 runs were performed as proxy training-validation dataset of our proxy. Fifteen runs are sampled for each half-cycle. The other eight samples are representative of our Latin hypercube space. It is based on a three-dimensional hypercube, which means eight edges based on the combination of the extremes of the studied parameters. We can see the edges in 3 months and 12 months half-cycle (the corners of the plots).

Based on the sampling, the rates were allocated equally for each injector well. Half-cycle length is also defined for 30 days per month for easier proxy modeling rather than the Gregorian calendar term. Then, runs were performed using PETREL and E300. The total time to do the whole run based on the LHS is **9 hours and 43 minutes**.

5.1.2 Data Preparation

Next, the data is extracted. FOPR, FCO_2PR, half-cycle, timestep, gas injection rate, and water injection rate are extracted. Normalization is performed to help the convergence of our ANN. As maximum-minimum normalization will be performed, **Table 5.1** shows the values for each parameter that will be used as input and output of our proxy (except WAG Ratio).

Table 5.1: Egg model variables statistical information.

Variables	Min	Max
Halfcycle (day)	90	360
Gas injection rate (Msm^3/d)	1	2
Water injection rate (sm^3/d)	3000	9000
WAG Ratio	0.60	3.61
Timestep	30	3660
FCO_2PR (kg-mole/d)	0	59843.75
FOPR (sm^3/d)	0	2026.89

As the zero timestep needs to be used as proxy input, the run results from the depletion process will be used (1 Jan 2021). After the normalization was performed, the database is then created in the same form as shown in **Figure 4.5**. All the data for proxy learning processes will be normalized, later denormalized when reported as an output.

The database will be separated by 75:25 for training and validation. The test dataset is not created as later the robustness will be tested using a blind test. Total of 8,296 data (68 runs x 122 timesteps) is separated as 6,222 data (51 runs) for training and 2,074 (17 runs) as validation. The edges of our Latin hypercube are forced to be the training data part, while the others are chosen randomly as training or validation in terms of its Case ID.

5.1.3 Proxy Building

WAG has a distinct behavior. For a WAG ratio higher than 1, we will get a sharp increase in production during the water injection phase. When the WAG ratio is less than 1, we will get an increase in production during the gas injection phase. This distinct behavior is then being used as the base of building the proxy. The database is then split based on its injection phase.

These concepts are being applied to our FCO_2PR proxy and FOPR proxy. Other than that, it is observed that there is unique behavior from all sampled runs. In the 1^{st} year of the injection (2021), we can see that CO_2 has not reached any of the producer wells. No cyclic behavior is then shown in the run results (**Figure 5.2**). The cyclic behavior can still be seen for a lower gas injection rate (1 Msm^3/day) until 1.5 years after injection started.

In addition, the FCO_2PR proxy is separated. This is done due to the low performance when FCO_2PR is combined as one ANN. All segments are then being integrated as one proxy (the whole proxy segmentation is shown in **Figure 5.2**.

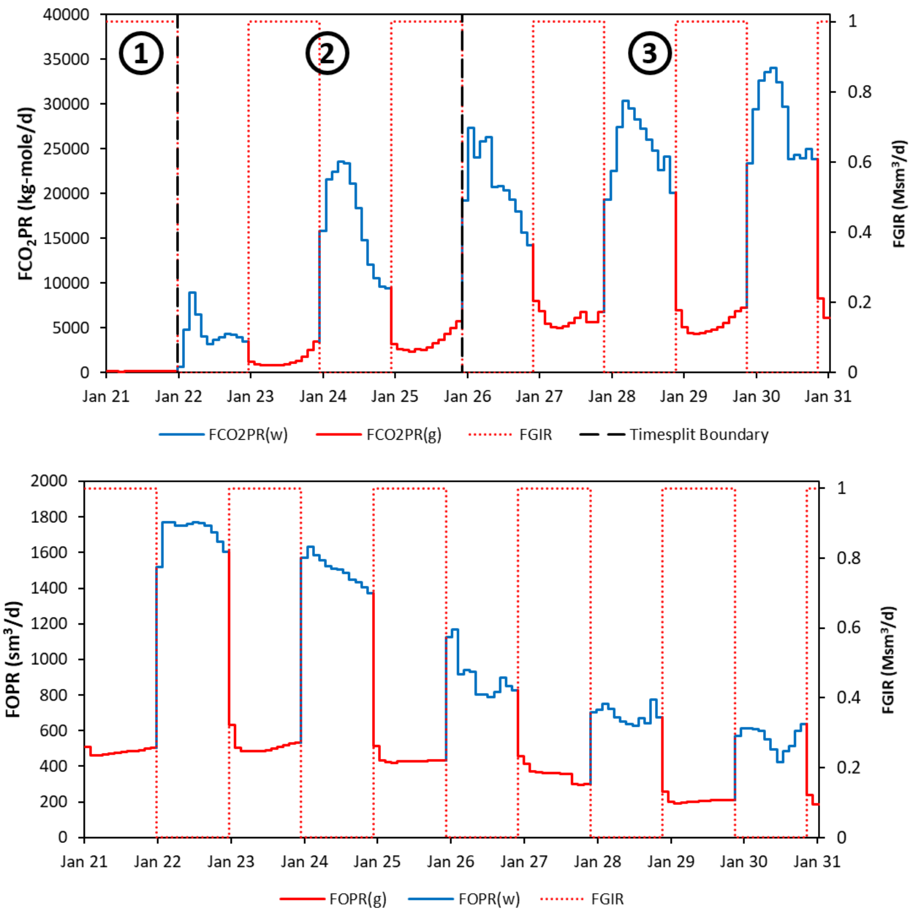

Figure 5.2: Proxy segmentation for Egg Model.

In conclusion, the proxies are segmented into several parts. For FCO$_2$PR, there are three segments. The first segment (1) is to mimic the first-year behavior, which is unique compared to the others. For this part, the proxy is not differentiated based on its injection phase. The second segment (2) is the second until the fifth year of injection. We separated based on the injection phase (blue for water, red for gas phase). In the third segment (3), the same concept is applied as the second segment for the sixth until the tenth year of injection. This results in 5 ANNs made to construct one proxy for FCO$_2$PR.

For FOPR, only two ANNs were made. These ANNs were made based on the injected fluid. For both proxies, seven ANNs were made. Then, a for loop and if function is used to do the whole work to run the proxy for 122 months (10 years). The database is then categorized based on these segmentations to be learned by our ANNs. We refer to **Figure 4.4** as ANN structure reference.

Proxy Training and Validation

As explained before, we performed an optimization process to find the best topology of our ANN for every segment. **Table 5.2** shows the best topology for every segment of our proxies. It can be seen that most of them need three hidden layers and higher than 17 nodes for every layer to learn the behavior. The learning rate itself varies between each proxy segment.

Table 5.2: Proxy segments topology (Egg Model).

Proxy			ANN Hyperparameter		
			Learning Rate	Hidden Layers	Nodes
FCO_2PR		1	0.0100	3	17
	gas	2	0.0031	3	20
		3	0.0029	3	17
	water	2	0.0036	3	20
		3	0.0059	3	20
FOPR	gas		0.0028	3	20
	water		0.0024	3	20

This optimum hyperparameter set is then being retrained. Epoch increased to 1500 to get their best performance. The whole process is using MSE as loss function, where the equation can be seen below.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - X_{i,predicted})^2 \tag{5.1}$$

The results after retraining are tabulated in **Table 5.3**. The whole segments show good fitness that is shown by R^2 values. The other error that is tabulated is the absolute error. Rather than presenting the MSE values which will be hard to interpret, normalized absolute error is reported. The absolute error values are about 0.1, where our data scale is between 0 and 1 after normalization.

Table 5.3: Proxy segments performance results (Egg Model).

Proxy			R^2		Absolute Error (Normalized)					
			train	val	Max (train)	Mean (train)	Min (train)	Max (val)	Mean (val)	Min (val)
FCO_2PR		1	98.788	97.665	1.41E-02	1.29E-03	2.00E-06	1.50E-02	1.45E-03	4.00E-06
	gas	2	99.380	98.582	3.93E-02	5.64E-03	1.40E-05	4.84E-02	8.32E-03	1.70E-05
		3	99.168	96.954	5.47E-02	8.51E-03	2.00E-06	6.47E-02	1.25E-02	4.20E-05
	water	2	98.921	96.793	7.57E-02	1.03E-02	4.00E-06	1.10E-01	1.70E-02	2.30E-05
		3	98.085	95.781	1.45E-01	1.40E-02	0.00E+00	1.19E-01	2.14E-02	4.90E-05
FOPR	gas		99.414	98.660	5.56E-02	5.51E-03	2.00E-06	7.68E-02	8.09E-03	2.00E-06
	water		99.454	98.694	1.29E-01	9.26E-03	5.00E-06	8.74E-02	1.37E-02	1.70E-05

It is essential to see the behavior of the segments as one proxy system. As one of the inputs is the results from the previous timestep, we used the value we obtained from the proxy as the input value. The segments were then tested as a part of one proxy to see its robustness. **Figure 5.3** shows a random case for each half-cycle length, compared to Eclipse results for both proxies. We can see that our proxy able to is learn the behavior, both FCO_2PR and FOPR.

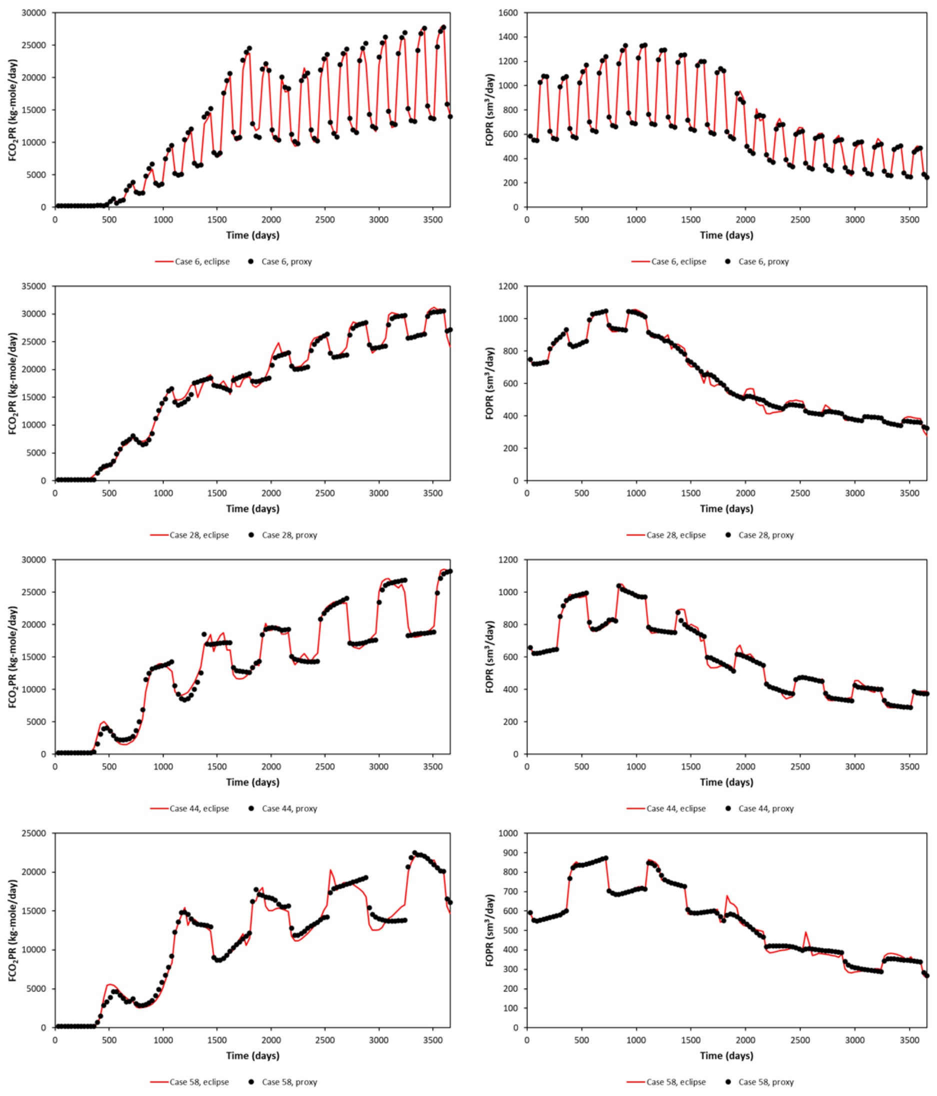

Figure 5.3: Proxy results and comparison with the reservoir simulation output (Egg Model).

Rather than relying on visual inspection, error calculation was performed to quantify the robustness of the proxy model. The values reported in **Table 5.3** cannot be used to see the whole proxy performance. For this reason, Average Percentage Relative Error (%) is used. The equation is shown below

$$\text{APRE} = \frac{X_{pred} - X_{real}}{X_{real}}.100 \tag{5.2}$$

Having a negative value means that the proxy underpredicts the results while having a positive value means overprediction. The preferred value is near zero.

Table 5.4: Proxy performance from training-validation process (Egg Model).

	APRE (%)		
	Min	Average	Max
FOPR	-2.307	-0.129	3.227
FCO_2PR	-9.882	0.842	10.962
FOPT	-2.435	-0.268	2.449
FCO_2PT	-3.252	1.052	6.114

Table 5.4 shows the statistics of APRE for the whole 68 runs we used for training and validation case. Overall, the average is near-zero values for the FOPR and FOPT (Field Oil Produced Total), which is a good indication of proxy robustness. Looking at its maximum and minimum value, we can see that they differ between 2.5% error. But overall, the proxy slightly underpredicts the oil production of thr reservoir model.

We can see a significant error in the maximum and minimum of FCO_2PR. This is due to the error we get at early times (first year), as a difference in small value leads to significant error calculated using the APRE term. The error window is slightly bigger than our oil production rate, but FCO_2PT (Field CO_2 Produced Total) can be the main focus as it will be used as our optimization study. The error window is around 5% for this proxy.

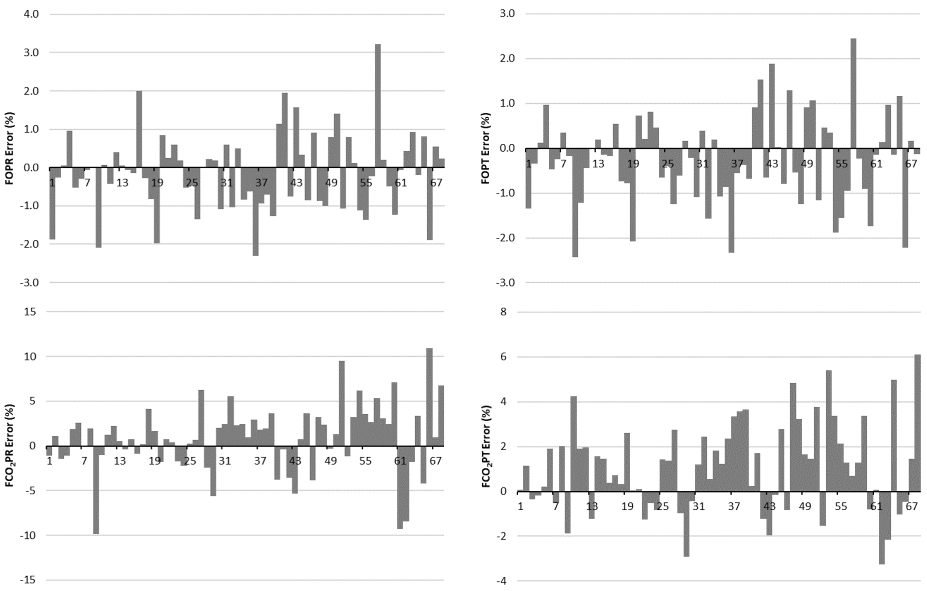

Figure 5.4: Relative percentage error for all training-validation dataset (Egg Model).

We can see the whole distribution of error in **Figure 5.4**. In general, FOPR and FOPT yields error in 1% window, and we have less than 25% of the data that yield error more than 1%. Looking into FCO_2PR and FCO_2PT, we have almost most of the error less than 5% (FCO_2PR) and 2% (FCO_2PT). Less than 25% of the data shows an error higher than that. Based on this, we can ensure that the proxy is good enough to be blind tested. Additional to that, the worst error for both FOPT and FCO_2PT is shown in **Figure 5.5**.

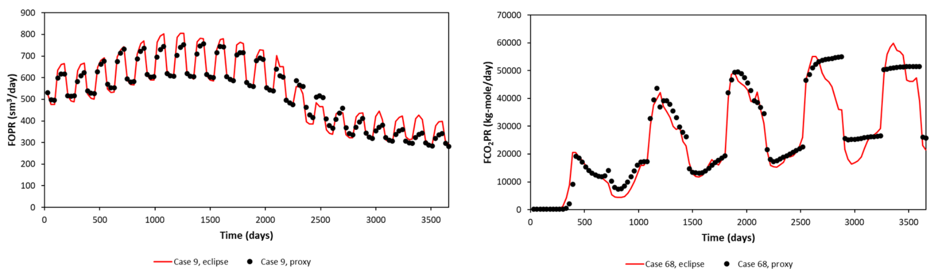

Figure 5.5: The worst performance from all training and validation data (Egg Model).

For the proxy to run one case, it takes **14KB** of space to report the results as Excel and **8.75 seconds** to run. This is 64.5 times faster and 6000 times smaller memory needed when compared to running with the reservoir simulator. For performing the whole process (separating database, topology optimization, relearning with higher epoch, running all training-validation data using the proxy model), it takes **1 hour and 23 minutes**.

Blind Test

Following the training, a blind test is then performed on the proxy models. Here, 12 runs outside of the sampling from LHS are prepared. The same assessment performed before is used, where APRE was used as the error calculation to see the performance. The error statistics of 12 blind test results are tabulated in **Table 5.5**.

Table 5.5: Proxy performance from blind test (Egg Model).

	APRE (%)		
	Min	Average	Max
FOPR	-1.078	0.646	2.850
FCO_2PR	-3.538	2.944	12.433
FOPT	-1.575	0.168	2.162
FCO_2PT	-1.783	1.740	5.810

The error window is about the same as the training-validation dataset. But we get a bit higher average values than our training-validation dataset, which is expected as the blind test data is not what we have used to train the proxy model. A considerable deflection is again shown in FCO_2PR maximum error. Observing the data for every timestep indicates that the error is due to the early times' error, the same as we get in the training-validation process. **Figure 5.6** shows the error from all blind test cases.

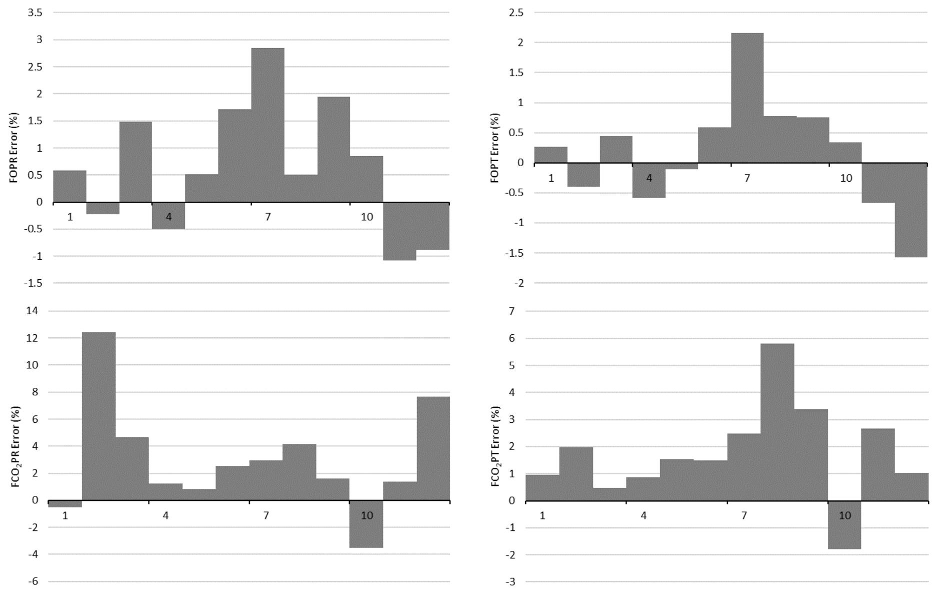

Figure 5.6: Relative percentage error for all blind dataset (Egg Model).

Figure 5.7 shows the proxy and Eclipse results comparison for the blind test. The upper part shows a random case selected out of 12, while the lower part shows the worst error case. We can see that mainly the high FCO$_2$PT error is due to the later time error (2028

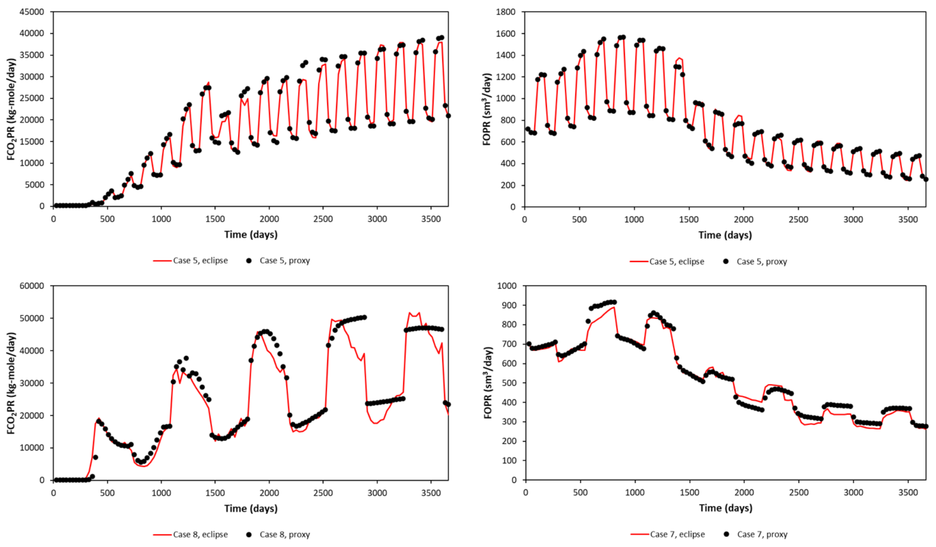

Figure 5.7: Blind test results and comparison with the reservoir simulation output (Egg Model).

and later). This period will yield a higher error in total CO_2 produced due to the magnitude of the variable. But, overall, we can see that our proxy can learn the behavior of FCO_2PR and FOPR. This confirms the robustness of the proxy, and it is ready to be used for optimization study.

5.1.4 Optimization Study

An optimization study will be performed using the proxy. As explained in the previous chapter, NSGA-II will be employed in this study. Here, possible optimum solutions non-dominating the others need to be found. The number of generations used in this optimization study is 100. No modifications in the programming script shown in the previous chapter is performed.

Total time needed to finish this optimization study is **2 hours 25 minutes** using NSGA-II with hyperparameter shown in **Section 4.4**. Forty Pareto solutions are found, as tabulated in **Table 5.6**. These solutions are also illustrated in **Figure 5.8**, where the FOPT and CO_2 stored are plotted in a negative term to show the front of the solution space easily. Reflecting on the values we get, all solutions converge to the extreme value of gas injection rate (2 Msm^3/day), while other optimization parameters vary between their possible value.

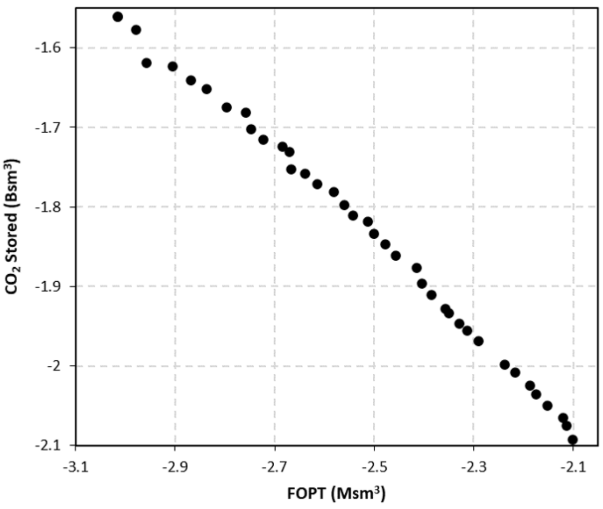

Figure 5.8: Optimization results for Egg model.

We can see from the solution space that our Pareto front almost has a straight-line behavior. We can choose any solutions from our Pareto front as we wish. But, if there are no objective functions that need to be prioritized, we can select the middle point of the front or the value near the mean of all possible positions. From all Pareto solutions, Case 28 is the one that has the closest value to the average of both objective solutions (2.523 Msm^3 of oil produced and 1.831 Bsm^3 of CO_2 stored) compared with the others.

Table 5.6: Pareto optimum of the optimization problem (Egg Model).

No	Halfcycle (month)	Gas Rate (Msm^3)	Water Rate (sm^3)	FOPT (Msm^3)	CO_2 Stored (Bsm^3)
1	3	2.000	3001	2.100	2.093
2	3	1.994	8996	3.016	1.561
3	3	2.000	8696	2.978	1.577
4	6	2.000	8999	2.957	1.619
5	6	2.000	7846	2.796	1.675
6	6	1.989	8695	2.904	1.623
7	6	1.998	8200	2.838	1.652
8	3	1.999	5252	2.456	1.861
9	3	1.983	5062	2.414	1.876
10	3	1.999	3409	2.151	2.050
11	3	1.999	4921	2.385	1.911
12	3	1.998	5924	2.581	1.782
13	6	1.993	7231	2.724	1.715
14	3	1.999	4489	2.290	1.969
15	9	1.998	6555	2.615	1.771
16	3	1.997	4023	2.217	2.008
17	3	1.998	5360	2.477	1.847
18	3	1.999	5793	2.561	1.798
19	3	1.997	5007	2.403	1.896
20	3	1.997	3768	2.186	2.024
21	3	1.998	4168	2.238	1.998
22	6	1.997	6555	2.639	1.758
23	3	1.999	5485	2.501	1.834
24	3	2.000	5689	2.542	1.811
25	6	2.000	7420	2.747	1.703
26	3	1.997	3172	2.120	2.065
27	3	1.996	6586	2.684	1.724
28	3	1.989	5554	2.513	1.818
29	3	2.000	3641	2.175	2.035
30	6	1.999	6768	2.666	1.752
31	6	1.993	8445	2.869	1.640
32	3	1.998	4601	2.312	1.955
33	3	1.998	4800	2.357	1.928
34	3	1.999	4676	2.328	1.947
35	3	2.000	3100	2.113	2.076
36	3	1.998	6502	2.671	1.731
37	3	1.994	8989	3.015	1.561
38	3	1.999	4768	2.349	1.934
39	3	1.998	7122	2.759	1.681
40	3	2.000	4341	2.265	1.985

It is crucial to test the relative error between the solution and the actual reservoir model run result. The selected Pareto optimum (Case 28) is then being tested using PETREL. All error results are shown in **Table 5.7** between the reservoir model and our proxy model. The work shows a tiny relative error for this selected case.

Table 5.7: Error of the selected pareto solution (Case 28, Egg Model).

	PETREL	Proxy	Relative Error (%)
FOPR	-	-	0.26
FOPT (Msm^3)	2.49	2.51	1.05
FCO$_2$PR	-	-	1.23
FCO$_2$PT (Mkg-mole)	78.84	79.43	0.75

Figure 5.9 shows the comparison between the reservoir simulator and proxy model. A good fit can be seen visually, while its error is quantified in the table above. We can see that our proxy is able to learn the behavior of the reservoir simulator model.

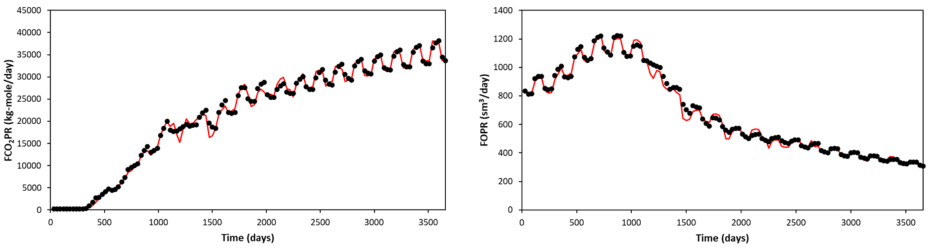

Figure 5.9: Comparison between reservoir simulator and proxy model for the selected pareto optimum (Case 28, Egg Model).

5.2 Complex Model - Gullfaks Model

There are some field models available to use, such as Volve, Norne, and Sleipner. But for this study, Gullfaks K1/K2 segment is used. As a field model, this model has complexity compared with Egg model. We can start with the grid system. Since it is a cornerpoint grid, neighboring cells can be identified by their XYZ coordinates rather than ijk coordinates. This needs to be observed when building a grid-based proxy model. But, it is masked in this study as the element volume is field scale.

Heterogeneous porosity and permeability are also one of the challenges in this model. Again, it is masked in this study due to the bigger element volume being observed. One that might be easily seen is the perforation zone. In this model, the depth for all wells is different. We can expect a delay in production response when we are changing between injection phases. **Figure 3.9** shows that the wells are injected from the lower layers while produced from the uppermost layers. This will delay the cyclic behavior that we saw in the Egg model, which we wanted our proxy model to learn.

The main model was already modified a bit, where we changed the position of all wells. It was explained in **Section 3.2.3**. The perforation was also changed. Some other parameters that are changed include the fluid model, relative permeability, and rock compaction. The changes are due to the complexity, but the originality (especially its geological features, such as faults and available multipliers) is preserved.

5.2.1 Design of Experiments

As expected, having a complex case requires a higher number of run samples than the simple case we performed before. For this model, we need in total 97 reservoir model runs as our proxy training-validation dataset. It consists of 89 samples from LHS and eight samples as the edge of our Latin hypercube. **Figure 5.10** shows the sampling for each half-cycle.

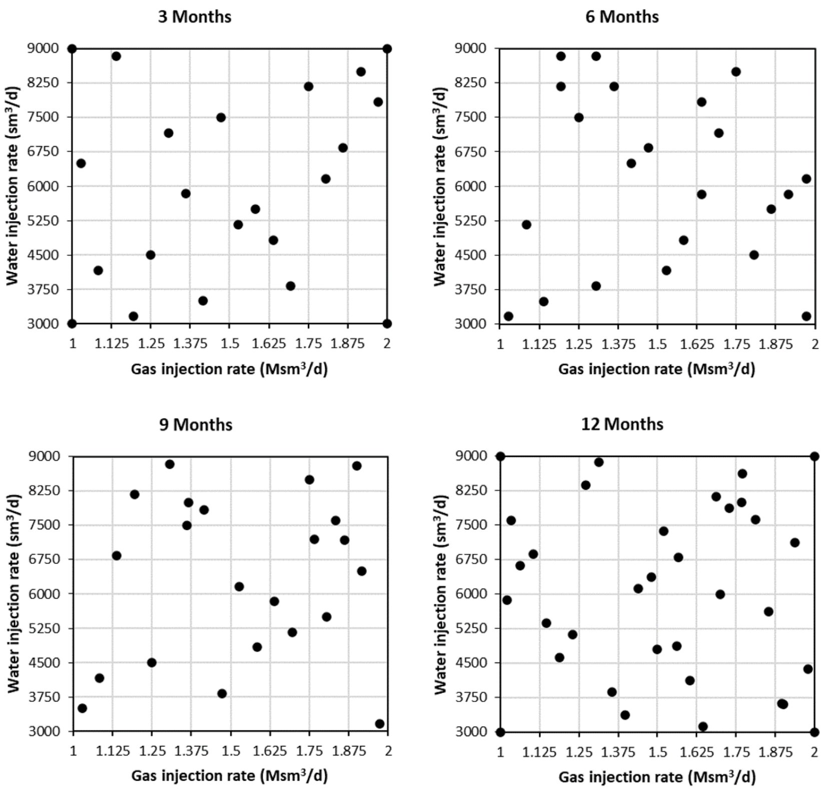

Figure 5.10: LHS performed for Gullfaks model.

The sampling itself is not equal between half-cycle lengths. It is found that the proxy fails to learn for higher half-cycle lengths (9 and 12 months) when we have an equal amount of sample with the lower half-cycle length (3 and 6 months). It is explained in the following subchapter. To fulfill this, 18 samples taken for the lower half-cycle, while we took 23 samples for nine months half-cycle length and 30 samples for 12 months half-cycle length. Then eight samples are added from the edge of our Latin hypercube.

After performing sampling, the same procedure was applied. The rates allocated equally for each injector wells and half-cycle length defined 30 days per month. The total time to do the whole run is **12 hours and 35 minutes**.

5.2.2 Data Preparation

After runs are completed, the same variables as the Egg model are extracted (FOPR, FCO_2PR, half-cycle, timestep, gas injection rate, and water injection rate). Normalization was also performed for building the database with the same format as for the Egg model case. **Table 5.8** shows the minimum and maximum values of each parameter used to build the proxy (except WAG Ratio).

Table 5.8: Gullfaks model variables statistical information.

Variables	Min	Max
Halfcycle (day)	90	360
Gas injection rate (Msm^3/d)	1	2
Water injection rate (sm^3/d)	3000	9000
WAG Ratio	0.42	2.55
Timestep	30	3660
FCO_2PR (kg-mole/d)	0	62301.46
FOPR (sm^3/d)	0	1558.03

Database will be created as before (refer **Figure 4.5**). Zero timestep value for the t 1 parameter will be using the value we got from the depletion study (1 Jan 2021). The database will be separated with the ratio (75:25), where we have 11,834 data (97 runs x 122 timesteps). It means 73 runs (8,906 data) will be used as training, while the other 24 runs (2,928 data) will be used as validation. We treat the edges of our Latin hypercube as training data like before.

5.2.3 Proxy Building

The distinct WAG behavior we discussed in the previous case is still recognized here. Yet, sharp production change can be seen after delays (of around one month) after the injection phase is changed. It is the basis for splitting the database. At first, we split the database based on injection phase (gas and water phase).

After that, the same behavior as explained before is identified. For FCO_2PR, the first year of the injection (2021) shows different behavior from the cyclic behavior. No injected CO_2 reached the production wells until early 2022 from all the cases we run. In addition, to enhance the quality of the proxy, the timespan is divided in 2.5 years intervals. This is due to the low performance when combined as one ANN. As it is a little bit more complex than what we had before, here we listed the whole proxy segments for building the FCO_2PR and FOPR proxy model:

1. FCO_2PR Proxy (total = 9 ANNs)
    - First segment (0 - 360 days): no separation between water and gas phase
    - Second segment (360 - 900 days): separated between water and gas phase
    - Third segment (900 - 1800 days): separated between water and gas phase
    - Fourth segment (1800 - 2700 days): separated between water and gas phase
    - Fifth segment (2700 - 3660 days): separated between water and gas phase

2. FOPR Proxy (total = 8 ANNs)
    - First segment (0 - 900 days): separated between water and gas phase
    - Second segment (900 - 1800 days): separated between water and gas phase
    - Third segment (1800 - 2700 days): separated between water and gas phase
    - Fourth segment (2700 - 3660 days): separated between water and gas phase

The details for segmentations for each proxy model is shown in **Figure 5.11**.

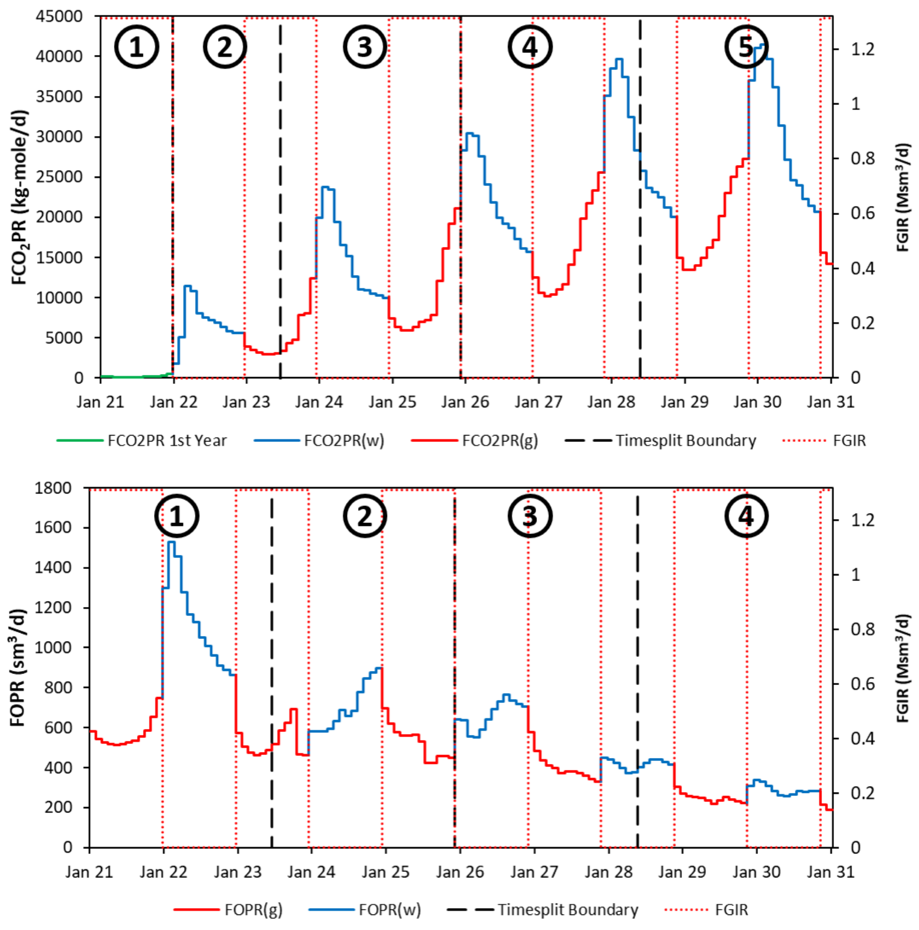

Figure 5.11: Proxy segmentation for Gullfaks Model.

In total, 17 ANNs are needed to create both proxies. These ANNs will be combined in a loop function to act as one proxy model to mimic the 10 years timespan defined. As a start of building the proxy model, the database will be segmented based on the segmentations listed above. **Figure 4.4** is used as the reference of ANN structure for all ANNs that we will make for this model.

Proxy Training and Validation

The first step is to find the best topology for all proxy segments. **Table 5.9** shows the best topology based on minimization of our loss function, MSE (**Equation 5.1**). We can see that most of the ANNs need three hidden layers with nodes of more than 15 to learn the segmented data.

Table 5.9: Proxy segments topology (Gullfaks Model).

Proxy			ANN Hyperparameter		
			Learning Rate	Hidden Layers	Nodes
FCO_2PR	1		0.0074	3	19
	gas	2	0.0079	3	20
		3	0.0025	3	20
		4	0.0100	3	19
		5	0.0042	3	20
	water	2	0.0100	3	20
		3	0.0023	3	20
		4	0.0037	3	18
		5	0.0100	3	19
FOPR	gas	1	0.0100	3	20
		2	0.0073	3	20
		3	0.0100	3	18
		4	0.0100	3	20
	water	1	0.0071	3	17
		2	0.0053	3	20
		3	0.0100	2	16
		4	0.0050	3	20

Following this, ANN with the best topology is then being retrained using a higher epoch (1500). Here, the fitness is shown in terms of R^2 and normalized absolute error for both training and validation dataset. The details can be seen in **Table 5.10**.

Table 5.10: Proxy segments performance results (Gullfaks Model).

Proxy			R^2		Absolute Error (Normalized)					
			train	val	Max (train)	Mean (train)	Min (train)	Max (val)	Mean (val)	Min (val)
FCO_2PR	1		99.482	98.541	8.90E-03	8.39E-04	6.35E-06	1.31E-02	7.73E-04	1.06E-05
	gas	2	99.119	97.722	2.91E-02	3.69E-03	7.11E-07	4.08E-02	4.92E-03	3.01E-05
		3	99.333	98.213	7.20E-02	7.80E-03	6.06E-06	5.80E-02	1.06E-02	1.88E-05
		4	99.636	99.151	4.57E-02	7.45E-03	9.53E-07	4.22E-02	8.87E-03	2.76E-05
		5	99.685	99.024	5.66E-02	7.20E-03	4.78E-06	4.87E-02	9.92E-03	7.59E-05
	water	2	98.590	97.344	5.49E-02	5.88E-03	2.36E-05	4.49E-02	7.25E-03	2.61E-05
		3	98.957	98.203	6.73E-02	7.94E-03	3.28E-06	6.00E-02	9.67E-03	6.44E-05
		4	99.612	99.321	5.05E-02	7.04E-03	9.36E-06	4.54E-02	8.51E-03	4.58E-05
		5	99.471	98.848	6.18E-02	7.53E-03	1.16E-05	5.24E-02	9.81E-03	1.41E-05
FOPR	gas	1	99.160	98.213	1.05E-01	7.28E-03	1.10E-06	8.68E-02	8.96E-03	2.57E-05
		2	97.206	90.133	8.09E-02	1.28E-02	1.98E-05	8.70E-02	2.04E-02	1.22E-05
		3	98.430	97.192	2.98E-02	5.65E-03	3.88E-06	3.03E-02	7.00E-03	1.96E-05
		4	99.222	96.900	1.44E-02	2.35E-03	9.79E-07	2.41E-02	4.14E-03	3.29E-07
	water	1	99.438	98.228	5.99E-02	8.66E-03	2.21E-06	1.20E-01	1.34E-02	5.17E-06
		2	98.059	95.077	7.97E-02	1.15E-02	3.53E-05	1.02E-01	1.58E-02	3.08E-05
		3	98.462	97.293	4.52E-02	6.94E-03	4.43E-06	5.29E-02	8.30E-03	2.21E-05
		4	99.283	98.083	1.86E-02	2.94E-03	5.37E-06	1.55E-02	3.84E-03	1.59E-06

As seen, the proxy segments behave well individually, and now we can consider it as one proxy model. The segments are combined based on the time split boundary, and the results generated by the proxy will be used as the input to the proxy at the previous timestep. **Figure 5.12** shows randomly chosen case for each half-cycle length, starting from 3 months half-cycle length (upper part) to 12 months (lower part). Eclipse results are also reported to compare with the result of the proxy model. Roughly, we can see that the combined proxy segments are able to work as one proxy model, both for FCO_2PR and FOPR.

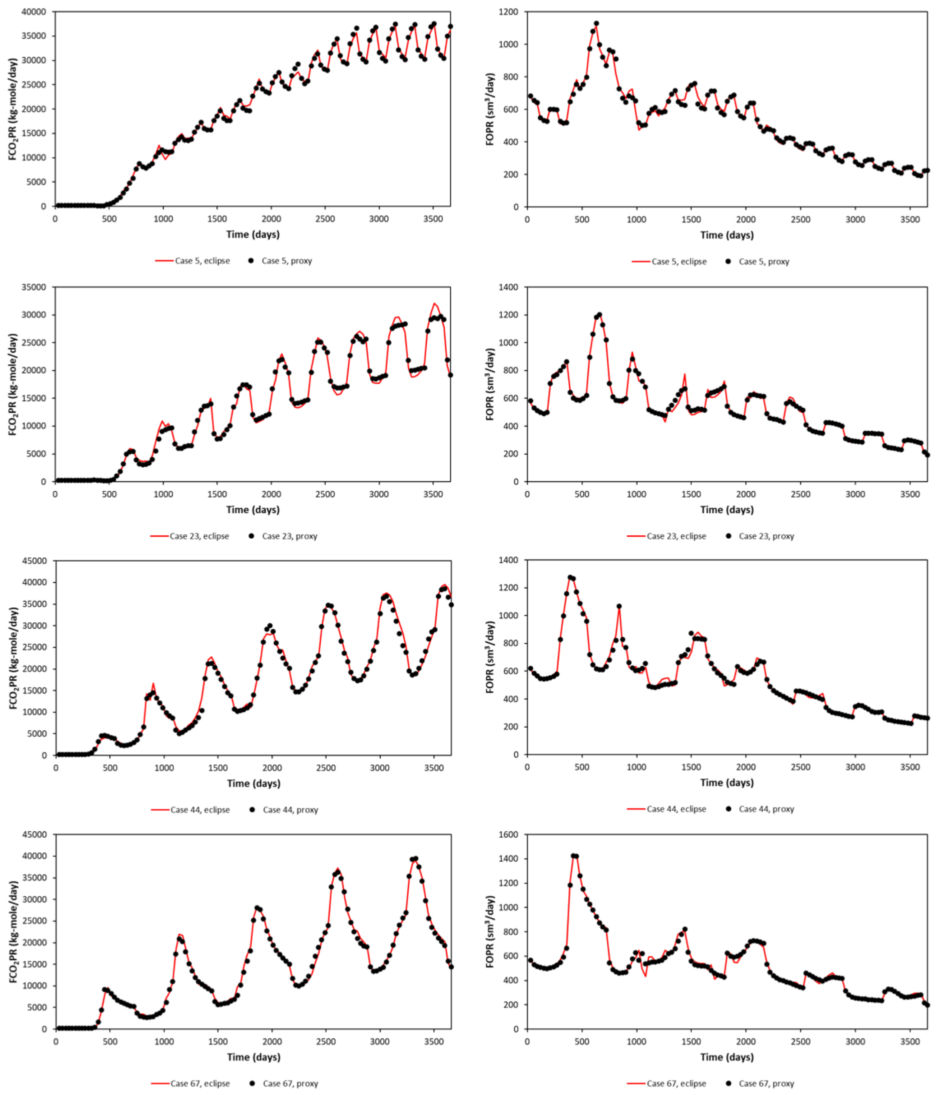

Figure 5.12: Proxy results and comparison with the reservoir simulation output (Gullfaks Model).

We can see that the cyclic behavior is masked at some points, as we see here for FOPR. Our proxy model can learn such complex behavior. We can confirm the robustness visually, yet we must confirm it quantitatively. **Table 5.11** shows the performance based on average relative error for 97 training-validation cases.

Table 5.11: Proxy performance from training-validation process (Gullfaks Model).

	APRE (%)		
	Min	Average	Max
FOPR	-2.364	0.042	1.443
FCO_2PR	-2.871	1.270	11.166
FOPT	-2.533	-0.125	1.429
FCO_2PT	-6.048	-0.909	2.778

We can see that the average of APRE is near zero. The error window is pretty small for FOPR and FOPT, which is around 2.5% error. This confirms the robustness quantitatively. FOPT and FOPR will be used for optimization, and we can see that both proxies under-predict the total oil and CO_2 produced, yet it is below 1%.

For FCO_2PR and FCO_2PT, we encountered high maximum and minimum error. For FCO_2PR, the error is mainly due to the early times' error, as a smaller magnitude can yield a higher error even with a slight difference with the actual value. While for FCO_2PT, the error accumulated in the later timestep (since we have higher magnitude there) affects significantly the total CO_2 produced. We need to confirm this by looking through all the runs used. **Figure 5.13** shows the results for every case and their error.

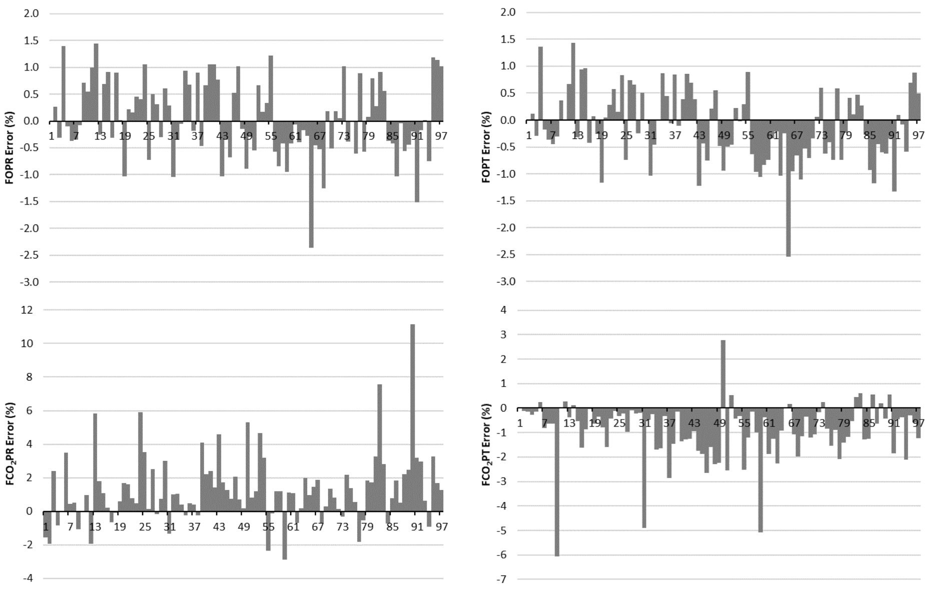

Figure 5.13: Relative percentage error for all training-validation dataset (Gullfaks Model).

We can see that for FOPR and FOPT, 75% of the trained-validated data fall in the 1% error window, while the others are out of this window. For FCO_2PR, more than 50% fall in the 2% error window. Eight out of 97 cases have errors more than 4% here. FCO_2PT shows good results where 80% of the data fall in the 2% error window, while only three out of 96 cases yield error higher than 3%. Based on this, we can conclude that the proxy is good enough from training-validation results. The worst performance, evaluated from the error of FOPT and FCO_2PT is shown in **Figure 5.14**.

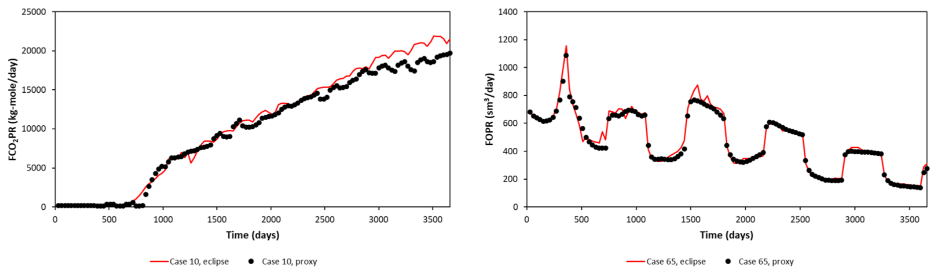

Figure 5.14: The worst performance from all training and validation data (Gullfaks Model).

For the proxy to run one case, it takes **13KB** of space to report the results as Excel and **9.16 seconds** to run a case. Comparing to the results we got in **Table 4.1**, this result is 57.1 times faster and 7400 times smaller memory is needed to perform one task. For performing the whole process (separating database, topology optimization, relearning with higher epoch, running all training-validation data using the proxy model), it takes **2 hours and 49 minutes**.

Blind Test

To test the robustness of the proxy model, 12 runs were prepared for blind test. It should be noted that these 12 cases were randomized, so the blind test case that we have in the Egg Model is different from what we have here. We performed an APRE assessment for error calculation for each run. The error statistics is tabulated in **Table 5.12**.

Table 5.12: Proxy performance from blind test (Gullfaks Model).

	APRE (%)		
	Min	Average	Max
FOPR	-0.700	0.511	3.930
FCO_2PR	-1.823	2.006	10.055
FOPT	-0.728	0.194	3.405
FCO_2PT	-2.191	-0.071	5.612

For FOPR and FOPT proxy, we can see that the error is slightly higher than the training-validation case. While looking into FCO_2PR and FCO_2PT, we can see that the error window is about the same as we have during the proxy training. We can see more details in **Figure 5.15** as the error is presented for each blind test. We can see that almost all cas-

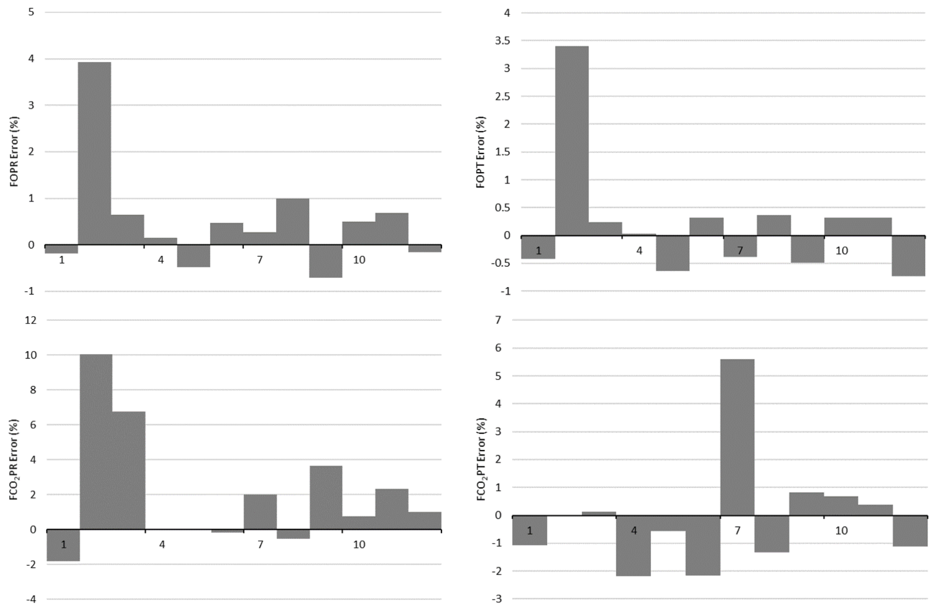

Figure 5.15: Relative percentage error for all blind datasets (Gullfaks Model).

es have an error less than 2% in general, which is a good indicator of the robustness of the proxy model. As a reference, random blind test (upper) and the worst blind test (lower) in terms of the case FOPT and FCO$_2$PT error are plotted in **Figure 5.16**.

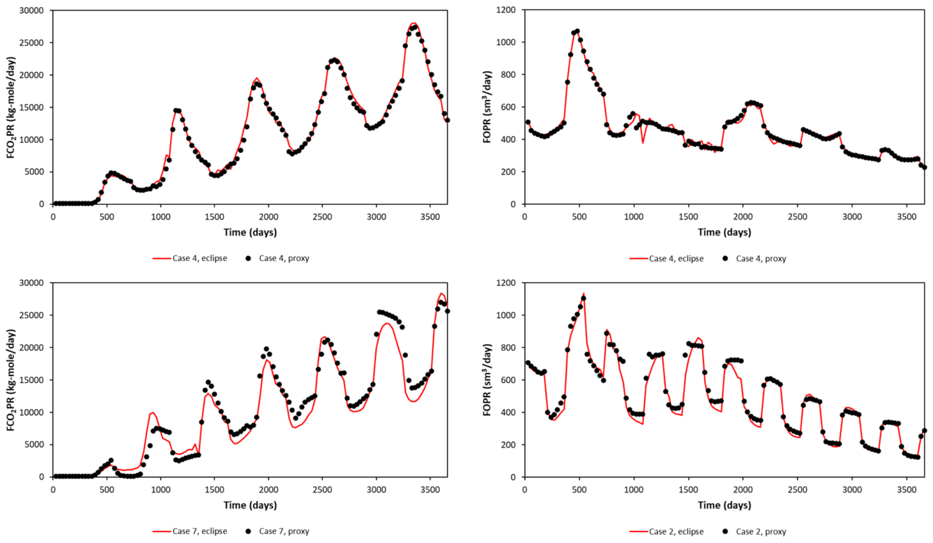

Figure 5.16: Blind test results & comparison with the reservoir simulation output (Gullfaks Model).

We can see that the proxy can learn the cyclic behavior and the masked cyclic behavior of the reservoir simulation model observed due to the complexity of the geological model. From the error, the proxy shows promising results and confirms its robustness. This proxy is then ready to be used for optimization study.

5.2.4 Optimization Study

Using the proxy built in the previous section, an optimization study can be performed. The same method and hyperparameters (as for the Egg model) are used. With 100 generations, we got Pareto front as shown in **Figure 5.17** and tabulated in **Table 5.13**. To finish the whole optimization runs, it takes **2 hours and 23 minutes**.

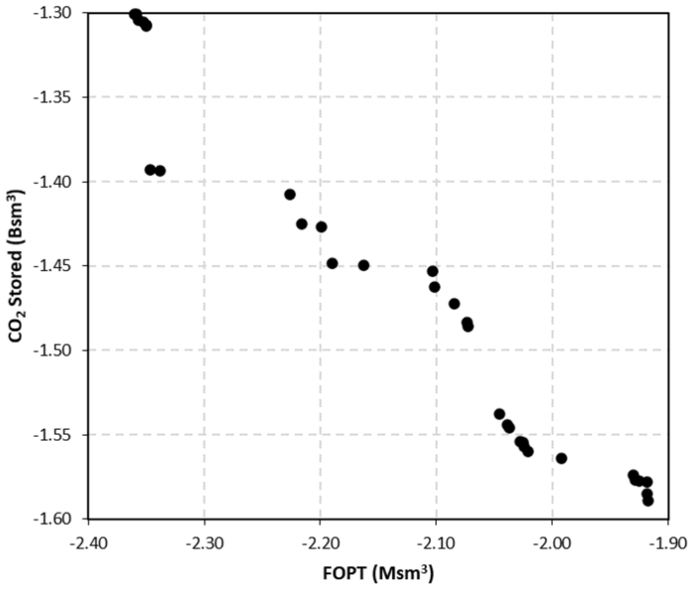

Figure 5.17: Optimization results for Gullfaks model.

We can see that the solution space of Pareto front is not linear, unlike for the Egg Model. This is because there exist faults, perforations installed differently between wells, and other effects due to the heterogeneity of the model. Overall, the optimization converges to maximum gas rate (2 Msm^3/d), while half-cycle varies between 3 to 9 months, and water injection rate varies along all possible values.

In the Egg model, we saw that increase in water rate will increase the amount of oil produced and decrease the amount of CO_2 stored. The same behavior is observed here. Egg model was modified to have about the same in-place volume as Gullfaks model. Yet overall, Gullfaks has lower recovery and CO_2 storativity compared to the Egg model.

Table 5.13: Pareto optimum of the optimization problem (Gullfaks Model).

No	Halfcycle (month)	Gas Rate (Msm^3)	Water Rate (sm^3)	FOPT (Msm^3)	CO_2 Stored (Bsm^3)
1	6	1.994	3825	2.025	1.555
2	9	1.950	6454	2.163	1.450
3	6	1.994	3981	2.039	1.544
4	6	1.994	4428	2.073	1.483
5	9	1.989	7616	2.226	1.407
6	9	1.996	3202	1.926	1.577
7	9	1.994	3107	1.918	1.585
8	6	1.999	3489	1.992	1.564
9	6	1.988	8918	2.353	1.305
10	9	1.985	3165	1.919	1.578
11	6	1.983	8903	2.350	1.307
12	6	1.999	4933	2.102	1.463
13	6	1.982	8903	2.350	1.308
14	6	1.988	8999	2.357	1.304
15	6	1.998	3802	2.024	1.557
16	6	1.994	3965	2.037	1.546
17	6	1.989	5035	2.104	1.453
18	9	1.998	3245	1.931	1.574
19	9	1.965	7136	2.199	1.427
20	6	1.995	3850	2.028	1.554
21	6	1.995	4051	2.045	1.538
22	6	1.997	4408	2.073	1.486
23	3	2.000	8994	2.347	1.393
24	6	1.999	8969	2.360	1.300
25	9	2.000	3046	1.917	1.589
26	9	1.991	6735	2.190	1.448
27	9	1.999	7259	2.216	1.425
28	9	2.000	3219	1.928	1.577
29	6	1.993	4646	2.084	1.472
30	3	1.999	8834	2.338	1.394
31	6	1.999	8938	2.358	1.301
32	6	2.000	3766	2.021	1.560

To test the results, as no preference of objective functions is prioritized, the nearest mean value for both objective functions will be selected to test the performance. Case 17 has the closest value to the mean of the Pareto front (The average of FOPT is 2.12 Msm^3 and CO_2

Table 5.14: Error of the selected pareto solution (Case 17, Gullfaks Model).

	PETREL	Proxy	Relative Error (%)
FOPR	-	-	1.04
FOPT (Msm^3)	2.08	2.10	1.19
FCO_2PR	-	-	0.45
FCO_2PT (Mkg-mole)	95.91	94.83	-1.12

stored is 1.47 Bsm^3). The selected case will then be tested and compared by running the case with PETREL. The results are tabulated in **Table 5.14**. We can see that small error was obtained compared with PETREL. **Figure 5.18** shows the rate comparison between proxy (black dot) and reservoir simulator (red line). We can see a jump in the middle of the FCO$_2$PR curve, yet the overall behavior aligns with reservoir simulator.

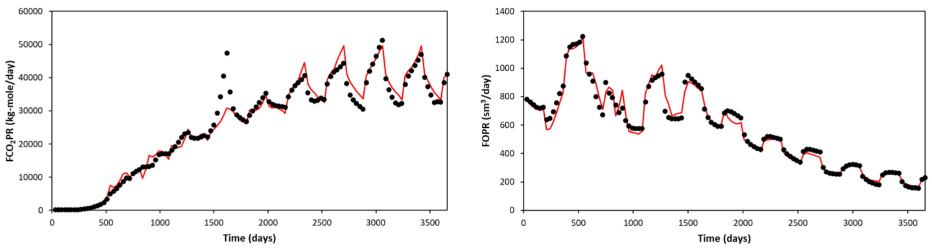

Figure 5.18: Comparison between reservoir simulator and proxy model for selected pareto optimum (Case 17, Gullfaks Model).

5.3 Behind the Success Story

The results we described above are obtained after several trials. **Figure 5.19** shows trials performed to create the proxy model for both models, where the red-colored ones are the proxy models presented above. The details are explained in this subsection.

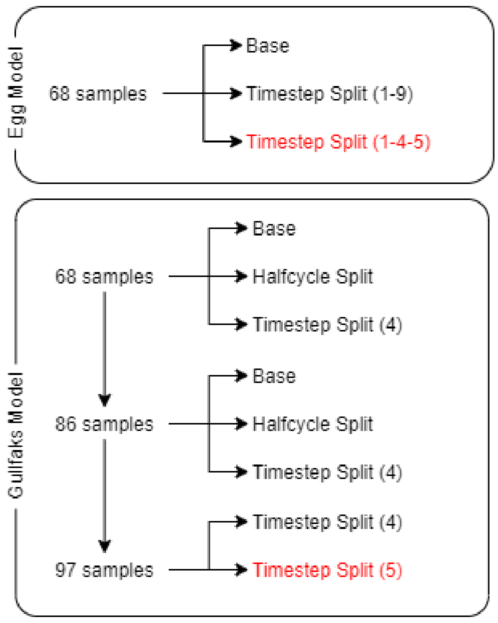

Figure 5.19: Trials to develop the proxy models based on sampling and proxy segmentation.

Figure 3.1 is the foundation of how the proxies are developed and improved. When proxy fails to learn the whole data, segmentation is the first effort employed to improve its performance. The definitions of proxy segmentations listed in **Figure 5.19** are listed below.

1. Egg Model
   - Base : Both proxies (FOPR and FCO_2PR) are segmented based on the injection phase.
   - Timestep split (1-9) : base idea is implemented. Then, for FCO_2PR, the proxy is segmented in two parts, the first year and later times.
   - Timestep split 1-4-9 : Explained in **Section 5.1**.
2. Gullfaks
   - Base : Both proxies (FOPR and FCO_2PR) are segmented based on injection phase.
   - Halfcycle split : base idea is implemented. Then, the proxies segmented based on low halfcycle length (3 and 6 months) and high halfcycle length (9 and 12 months).
   - Timestep split (4) : base idea is implemented, then the proxies segmented in 2.5 years time interval.
   - Timestep split (5) : Explained in **Section 5.2**.

When the efforts to improve the proxy performance by segmenting the proxy fail, the number of sampling is increased. This is performed until the proxy performance is acceptable.

The whole training-validation dataset is evaluated by comparing with reservoir simulator results to assess the proxy performance. The performance is then assessed qualitatively (by looking at the behavior from plots) and quantitatively (APRE results). This quick assessment helps the proxy development process be more efficient, rather than blindly moving to blind test right after constructing the proxy. The process of building the proxy is through ANN topology optimization, retraining using higher epoch, and assessing individual segments loss function and fitness criteria (R^2 and normalized absolute error).

When the plot comparing proxy and the reservoir simulator results shows different trends, it will be counted as a failure instantly, and the assessment is stopped right away. When the proxy results match the simulator results (visually), we assess qualitatively using APRE. If a near-zero APRE value proves the high quality, we can move to a blind test. The assessment for the blind test is also the same as the training-validation dataset. When the robustness is confirmed from the blind test, the proxy model is ready for the optimization study.

5.3.1 Egg Model

Due to the simplicity of the data, this model took less time and ideas to develop the proxy model. We can see from the previous discussions that the cyclic behavior is not masked in this model. Hence, the behavior can be learned just using 68 run samples (**Figure 5.1**). **Figure 5.20** shows the fitness of the first two trials to develop the proxy model.

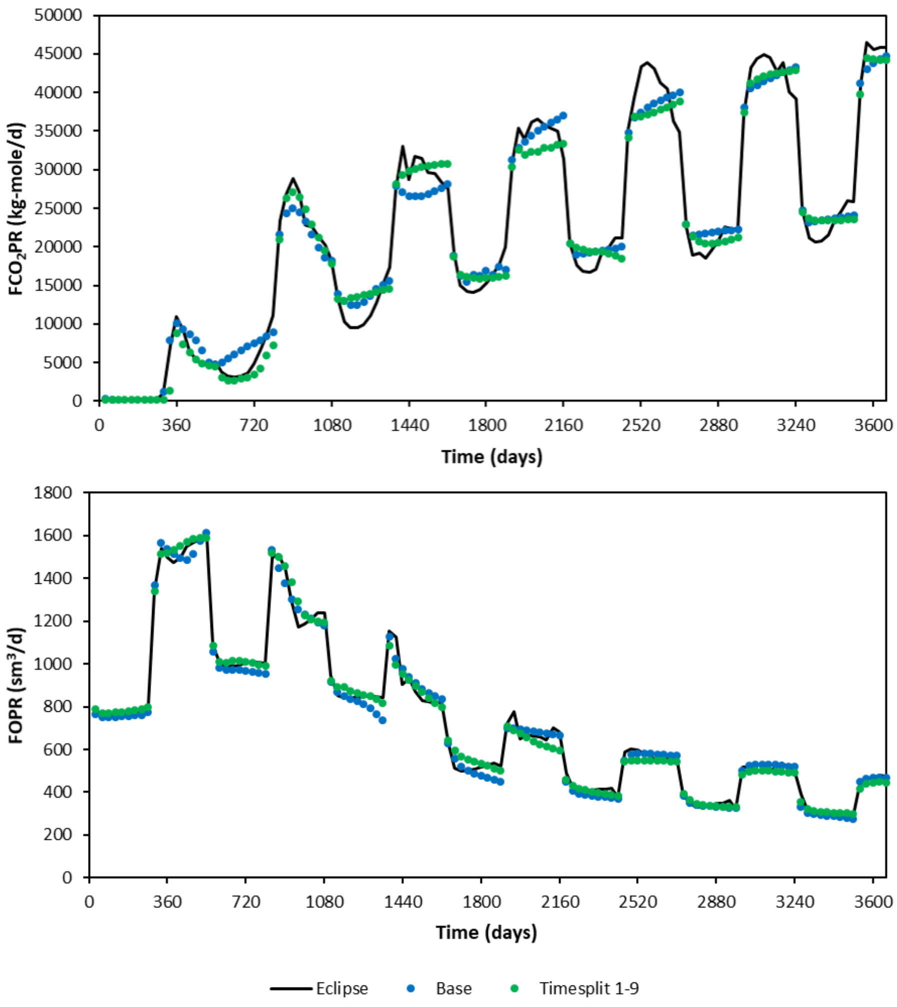

Figure 5.20: Proxy and Eclipse results for the trials to develop the proxy model of Egg Model.

The figure above is selected randomly from the 68 samples we have. We can see visually that for the base model, the proxy can mimic the behavior for FOPR, but it fails to mimic FCO_2PR. As it fails the visual assessment, we tried to segmentize the proxy for FCO_2PR. The first year is separated from the other data, as no cyclic behavior can be seen in that interval. The results show better trend compared to the base idea.

We can see some deflection in later times, where the proxy fails to mimic the behavior. Because of this, we segmented the time again into 1 year - 4 years - 5 years for FCO_2PR, as explained in **Section 5.1**. The results we obtained for this trial yield good performance both visually and in APRE statistics. Then, the proxy was tested using a blind test dataset. As the performance is sufficient, the proxy is then used for the optimization study.

5.3.2 Gullfaks Model

Looking at the geological model, we understand the complexity of the data in response to the changes in injection rates, and the half-cycle length of the EOR study compared with the simple Egg model. We observe sharp increase in production during low WAG ratio as CO_2 will go to the uppermost layer, where producer well perforations are located. In a lower WAG ratio, the effect will be delayed as the perforation of injector wells is located in the bottom layer of the reservoir. Based on this, we know that it will be harder to build the proxy for this model.

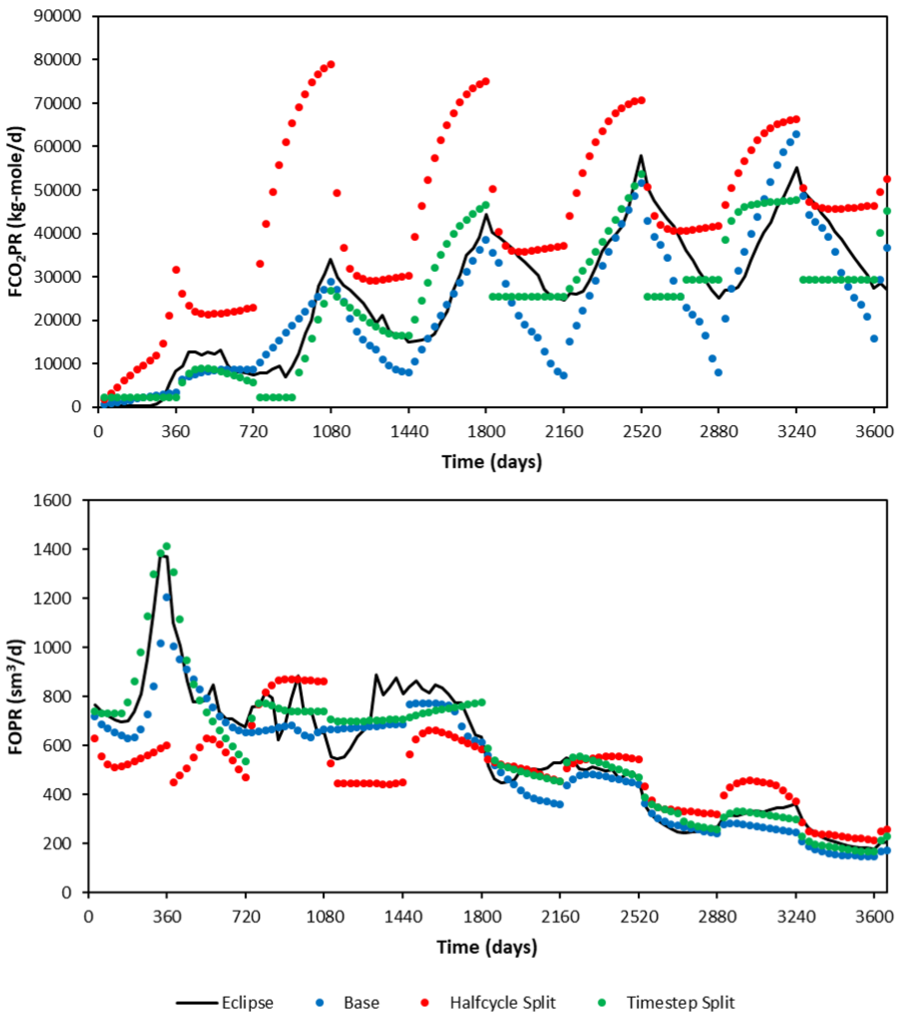

Figure 5.21: Proxy and Eclipse results for the trials to develop the proxy of Gullfaks Model using 68 run samples.

At the start, we are using the exact sampling as we have used for the Egg model (refer to **Figure 5.1**). Sixty-eight samples are used to develop the proxy model, where the results of proxy segmentation ideas are illustrated in **Figure 5.21**. Here, we will explain them one by one.

We can see visually that overall, all ideas employed using 68 run samples could not learn FCO_2PR and FOPR behavior. For the base case, most segments perform around 98% R^2 value for training dataset and 96-98% R^2 value for validation dataset. This seems reasonable enough, but in terms of normalized absolute error, it varies greatly between 0 to 0.17, which is not a good sign for this cascading proxy model.

For the half-cycle split and timestep split ideas, both models have similarities, where some segments have R^2 more than 99%. But, we still see errors of more than 0.14 when looking at the normalized absolute error. Having an error of 0.14 for data that varies between 0 and 1 means that we have more than 10% error. This error is carried over throughout the later timestep, as we use the value of the previous time step as the input of the proxy.

Then, the number of samples is increased. **Figure 5.22** shows the new sampling with 86 samples taken from the sampling space. Seventy-eight samples are taken, where for 3, 6,

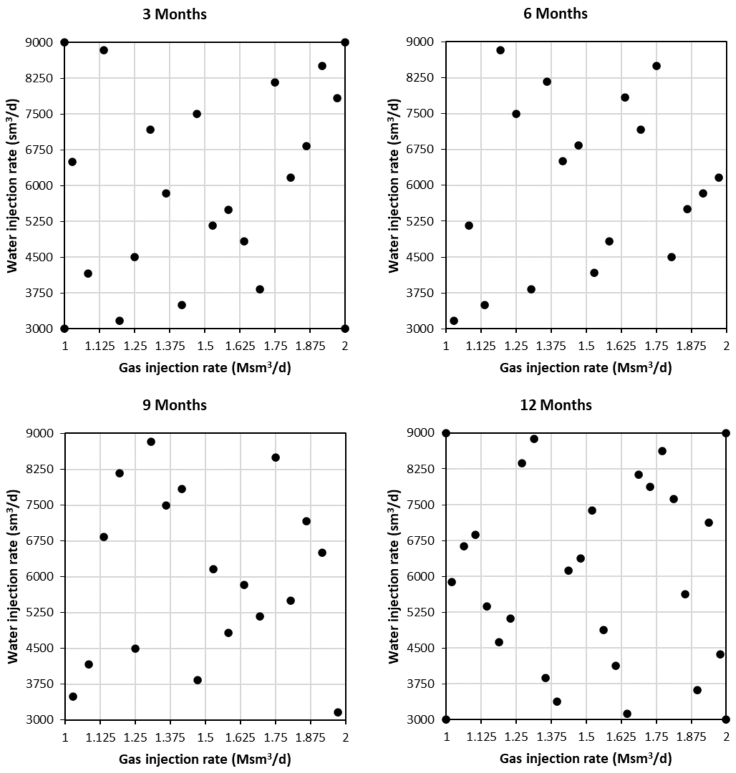

Figure 5.22: New sampling for Gullfaks model.

and 9 months half-cycle, we took 18 samples each, and 24 samples for 12 months half-cycle. Additional eight runs for the edge of Latin hypercube were also added to the dataset, resulting in 86 run samples. This unbalanced sampling was performed due to the proxy's failure in 68 samples to mimic the behavior of high half-cycle length.

We employed the same ideas explained before also for this new sampling. **Figure 5.23** shows the results of all the proxy segmentation ideas we used. The figure shows the behavior of the proxy for 9 months half-cycle. Even though all proxies fail to mimic the behavior of higher half-cycle length (9 and 12), all proxies now can mimic the behavior for the lower half-cycle length (3 and 6 months) which is not reported here.

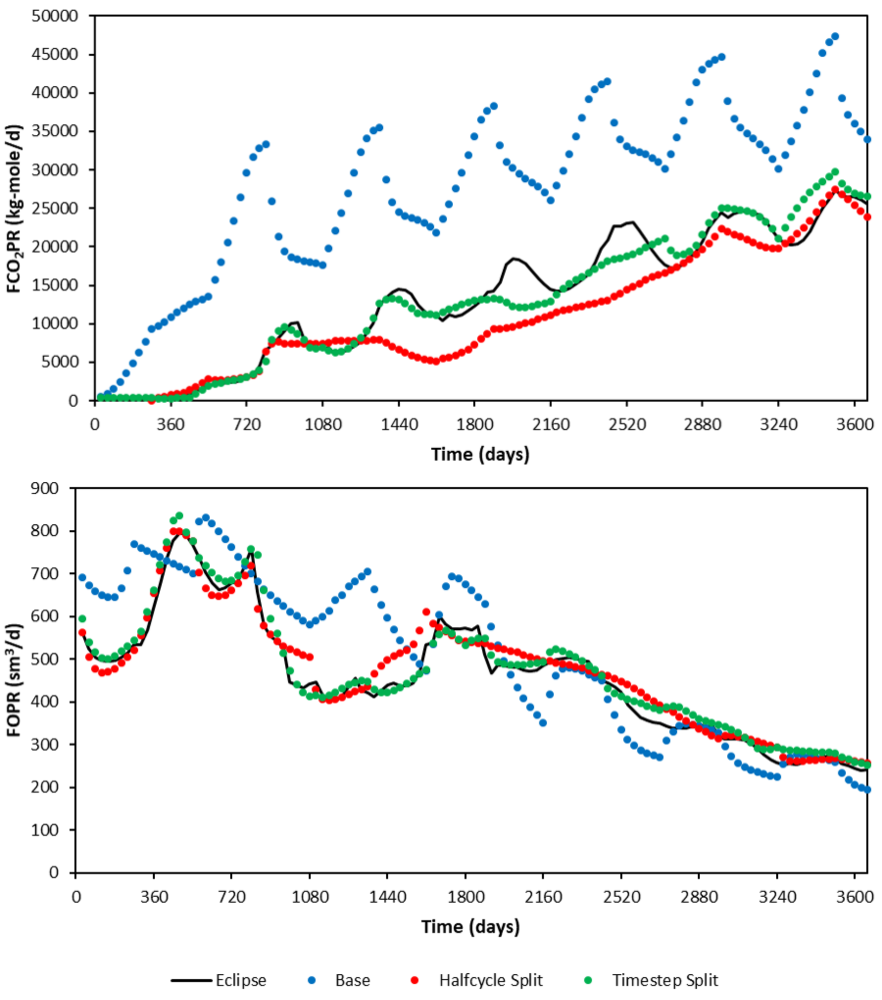

Figure 5.23: Proxy and Eclipse results for the trials to develop the proxy model of Gullfaks Model using 86 run samples.

All ideas have a good performance on the individual segment assessment from R^2 and normalized absolute error. We can see from the base idea that the proxy fails to visually perform as a good proxy model. We see high jumps in FOPR and FCO_2PR plot for the base idea. Hence, this idea will not be assessed further. We can see that for half-cycle split idea, it performs better visually compared to the base model, but it is not sufficient to be evaluated further when compared to timestep split.

Compared with the other results, the timestep split shows the best performance. We can see that in the FOPR curve, the model can mimic the masked cyclic behavior in the early time. Other than that, we can see the fitness is better compared to the other two ideas. For the FCO_2PR proxy, we can see that it fits better than the others, except for the middle time (1800-2880 days) when it fails to mimic the behavior. This is an indication of potential room for improvement. To do that, more sampling is added for the proxy model training-validation dataset, as shown in **Figure 5.24**.

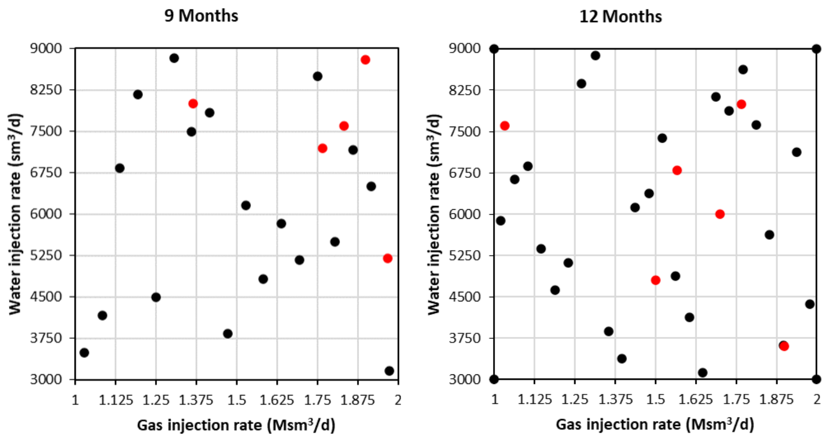

Figure 5.24: Additional runs (red) in addition to 86 run cases to improve Gullfaks proxy performance.

In total, 97 samples are now used as proxy learning materials. Five additional runs for 9 months half-cycle length and six additional runs for 12 months half-cycle are added to 86 samples from before. With the same procedure, we assign the training-validation, create and assess the performance of the proxy. Here, we directly use timestep split as our proxy segmentation idea. The result is shown in **Figure 5.25**. Better matches are now seen in the figure, where we considered the same case as in **Figure 5.23**.

Now, as it performs well visually, APRE assessment is performed. For the FCO_2PR curve, we see huge errors in early times (first year). The error at the first year varies between 5 to 140%, which is not desirable. So, the first year dataset was separated and a new ANN is made to represents this timespan. The final results were reported in **Section 5.2**. As the proxy passes the APRE assessment, a blind test can then be performed. The results are satisfying and can be concluded as a robust proxy. Then, the proxy is used for the optimization study.

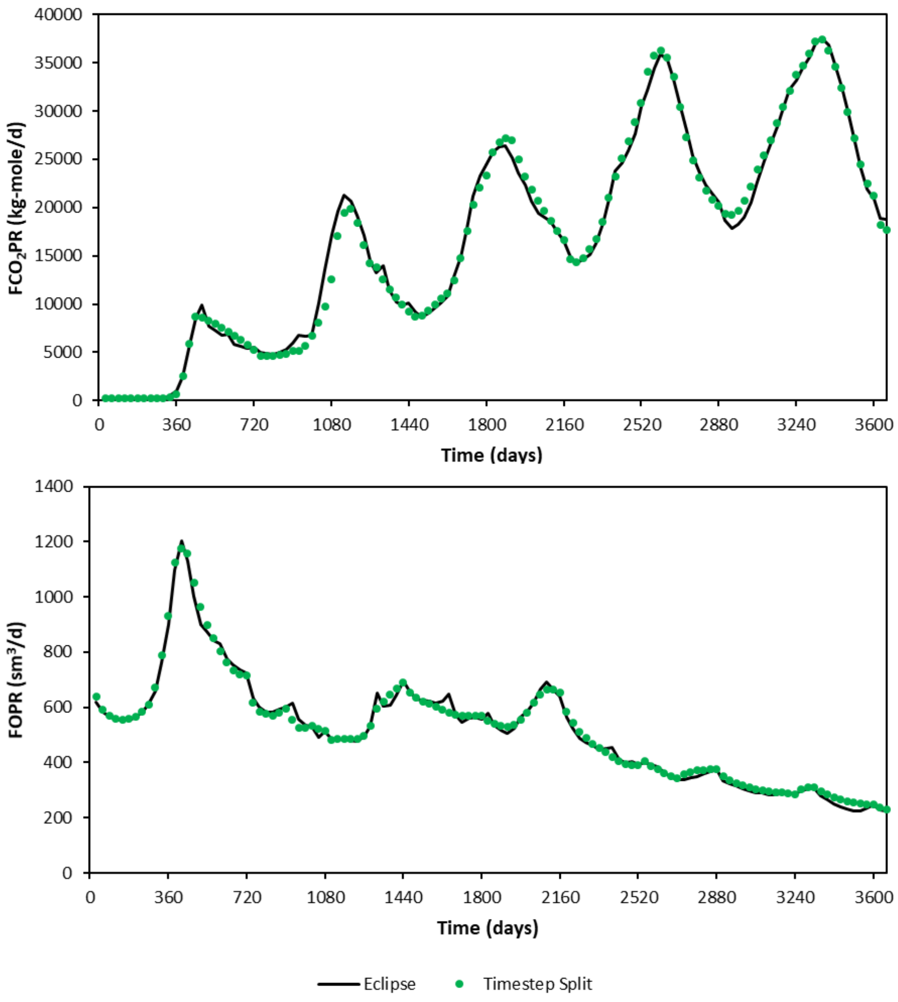

Figure 5.25: Proxy and Eclipse results for the trials to develop the proxy of Gullfaks Model using 97 run samples.

Chapter 6

Discussion and Evaluation

From the results obtained in this study, some points can be discussed and evaluated. All evaluations can be used to improve when a deeper study related to the proxy model will be performed later. Moreover, the time needed to build a proxy model will be discussed here to decide whether it is worth to be explored deeper or not.

6.1 Comparative Study

Every proxy model is case-specific, which is based on the data provided to be learned. Two proxy models that represent different reservoir models were made for solving optimization problem in this study. One of thee models (Egg Model) was modified to make both models similar in size, for comparison purposes. We will present a comparative study based on the steps of building the proxy model.

Data Complexity

The rate response to the WAG injection phase for both models is shown in **Figure 6.1**. Two cases are presented. The first part is for 90 days half-cycle length, 2 Msm^3/d gas injection rate, and 3000 sm^3/d water injection rate. This case is shown until 1800 days only. The second part is for 360 days half-cycle length, 2 Msm^3/d gas injection rate, and 9000 sm^3/d water injection rate. The magnitude of rates can be neglected for now.

These cases are being run as training cases on both models, as it is one of our Latin hypercube edges. First, for both FCO_2PR behaviors, we can see sharp increase in CO_2 production in the 90 days half-cycle . However, in 360 days half-cycle, for the Gullfaks model, there is a smooth transition rather than a sharp increase or drop in the production behavior. This different behavior will be learned using the same proxy model if it is not segmented based on its half-cycle length.

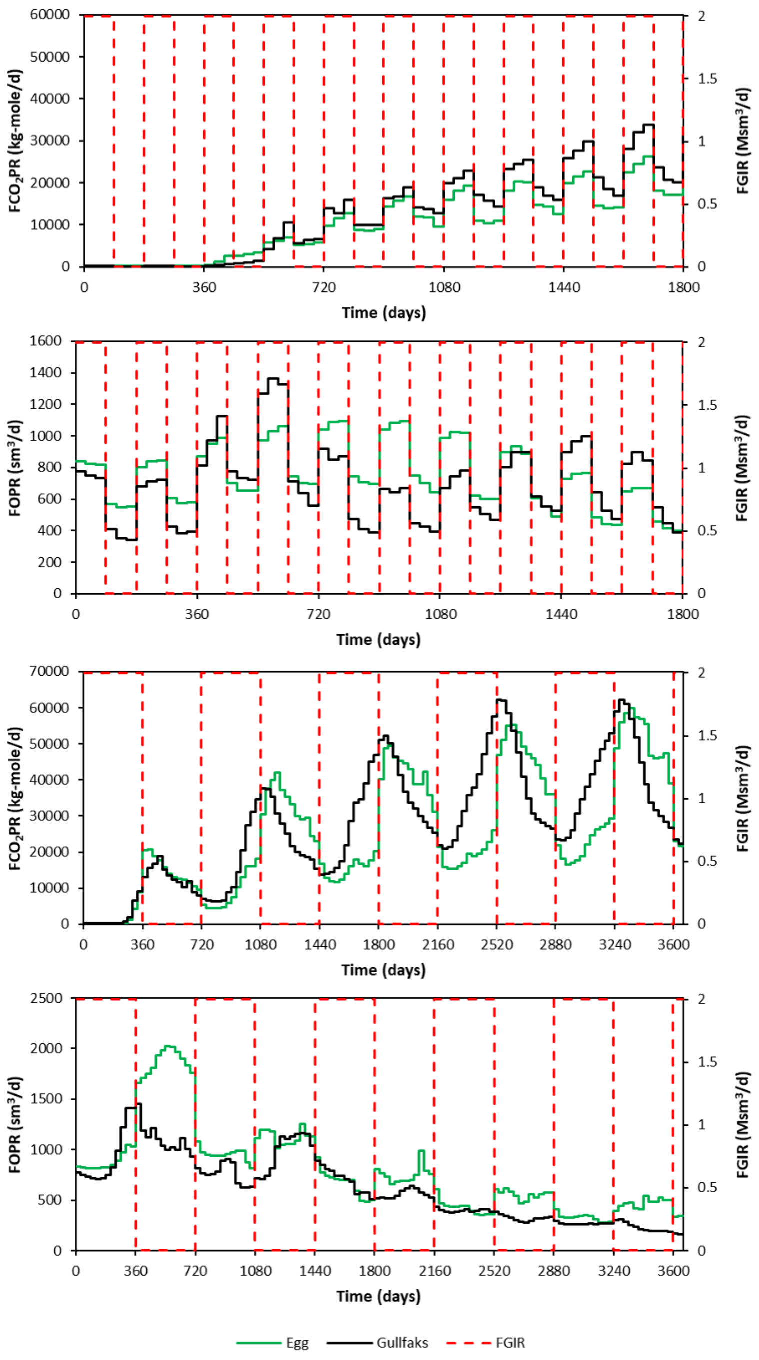

Figure 6.1: Differences between Egg Model and Gullfaks Model rate response to WAG injection phase, upper part for 90 days half-cycle, lower part for 360 days half-cycle.

Now we can observe the FOPR curve. Overall, we have an increase in production in the second year of production. Then, it will decline over time. Other than that, we can see sharp rate changes in both reservoir models in between each injection phase. However, again, it is masked in higher half-cycle length in the Gullfaks model. The smooth transition is observed in this model.

Two half-cycle lengths that are not reported here are six months and nine months. Six months half-cycle length still inherit the behavior of three months half-cycle length. For nine months half-cycle length, we start to see smoother changes as we have in higher half-cycle length. Other than that, the smoothness increases when the WAG ratio is between 0.9 - 1.3. As the injection rates of both fluids become almost equal, we can witness decline-like flow behavior in our FOPR.

We observe here the different responses of the two models primarily due to different geological model and well placement. Other parameters such as the fluid model, relative permeability relationships, and initialization condition might not have any impact since they are identical for both models. These complex behaviors were learned by both proxies that we made for their respective reservoir models.

Number of Samples Needed

Here, we can conclude that complexity aligns with the number of samples needed for developing the proxy. For the Egg Model, 68 samples are enough, yet in Gullfaks, we need 97 samples to develop the proxy model. The total time to run the whole sample is in alignment with the number of samples needed. This phase is the most time-consuming part as both models need more than 9 hours to complete all runs from the training-validation dataset.

Proxy Development Process

Just looking into **Figure 5.19**, we can see that more efforts are needed to build the Gullfaks proxy model compared with Egg model. However, we will evaluate each step of this process.

To improve the robustness, rather than increasing the number of samples directly, segmenting the proxy is the first effort. Segmenting the time into several parts shows the best improvement compared to other ideas. The question that might pop up is the number of segmentations needed. Three segmentations are enough for the Egg model, but more than five are needed for the Gullfaks model.

Each segment involves gas-water segmentations inside due to distinct behavior of each injection phase. Then, each segment is trained individually, which involves topology optimization, ANN relearning with higher epoch and error, and fitness analysis. Almost all segments require a high number of nodes (15-20 nodes, 20 is the maximum number of nodes for each layer) and hidden layers (2 to 3). It shows that a complex topology is needed to learn the rate behavior we want to mimic.

Moving to the assessment of each proxy segment, we see that for both models, we need at least R^2 near 98% to make the segments work as one proxy model later. The error range for each segment (normalized absolute error) also needs to be minimized. This takes into account the error that later timesteps have due to the cascading error of the proxy.

Before assessing the proxy robustness, we will be looking into the time-cost we need to build all proxy segments. Here, the time presented in the previous chapter includes the database segmentation, topology optimization, relearning ANN with higher epoch, and running the whole training-validation-blind dataset using the proxy model. Two hours and 49 minutes is needed for the Gullfaks model, and an hour and 23 minutes is needed for the Egg model. A higher amount of data and segmentation will increase the amount of time needed.

Proxy Robustness

To assess the robustness of each segment as one proxy, both visual and APRE are used. As explained before, when it is visually not robust, we will improve it right away rather than accessing quantitatively and moving to the blind test. We used APRE since all ANNs learned the model using normalized values. Assessing normalized values is hard as the order of magnitude is masked by the normalization.

In this study, the average APRE was used as the target is less than 2% for the training-validation phase. Other than that, the error window should be in the range of 10%. These numbers are selected without any solid basis but as the boundary for this study. Both proxy models must fulfill these criteria before being tested by the blind test dataset. As we observed, both models fulfill these criteria, and a more profound assessment was also performed to see the error variations for each training-validation case.

Both models are assessed using the blind test as the final test to check the robustness. In general, both models have similar APRE values for both training-validation and blind test. The robustness of blind test is assessed in the same way as we did for the training-validation test. In this study, when the proxy model meets the robustness criteria during the training-validation set, it tends to show the same robustness in the blind test.

Optimization Result

Both proxy models need an almost equivalent amount of time to finish the optimization study. This is true as the difference between the proxy models to finish a run is only 0.41 seconds. No hyperparameter was changed, and the total generations used for both models are the same, 100 generations.

The number of Pareto optimum is different, where we have 40 Pareto optima in the Egg model and only 32 Pareto optima in the Gullfaks model. The Pareto front of both models is different. Both still have the same behavior. If we increase the water injection rate while maintaining the gas injection rate, we will increase oil production while reducing the amount of CO_2 stored in the reservoir. For the Egg model, this behavior is almost linear. However, for the Gullfaks model, it is not anywhere near linear.

The nearest Pareto optimum with each objective function's mean value is tested in both models and compared with Eclipse runs. Their APRE is near the average APRE in the training-validation case, which is a good indication that the proxy is robust. It is noted that all running times are assessed using the same computer specifications. To run the case using PETREL, building the proxy, and performing the optimization study, PC with Intel(R) Core(TM) i7-8565U with 1.99 GHz processor with 8GB RAM is used. No parallel runs were performed during the study to allocate the whole CPU to do each task.

6.2 Evaluation and Improvements

After the whole study is finished for building the proxy model and performing an optimization study, a comprehensive evaluation is reported.

Data Sampling

In this study, 68 samples are the number of LHS performed initially, reflecting the number of samples used by Nait Amar et al. (2018) and Nait Amar et al. (2020) for CO_2-WAG study. It shows that it is sufficient for the Egg model, while it is not for the Gullfaks model. There are no guidelines nor rule of thumb about the number of runs needed to do the sampling for the field-based optimization study. The number of samples is important as this part takes most of the time for building the proxy.

As observed in this study, resampling means re-running the reservoir model. This is proven when we resampled the Gullfaks model from 68 to 86 samples. As we know that it is computationally heavy to run many cases, resampling should be avoided. To avoid this, the first ”guess” of the number of samples needed is essential. It seems to be okay if we sampled too many rather than not enough. However, the cost of too much data means that our ANN learning process will be longer, even though it might still be way shorter than resampling and re-running the reservoir model for creating the database.

Another way to tackle this is by adding several run samples by adding extra samples into our existing LHS results. It is proven when we added extra runs for the Gullfaks model from 86 to 97 samples. It needs to be reconsidered, as LHS divided the whole sampling space based on their equiprobable intervals, and each sample is taken from each interval. Adding extra runs as proposed here means that the number of samples will not be equiprobable again as before. More study about this is needed.

Another way is to change the sampling method. There are many sampling methods. Quasi-Monte Carlo sequences sampling, such as Halton, Hammersley, and Sobol, can be tested. Most proxy modeling for petroleum engineering relies on LHS as the sampling method. Studying further into this can be a new insight for proxy modeling.

Developing the proxy

From the start of the study, we limited the proxy to be built using ANN. There are many machine learning and deep learning methods to be used for building a proxy model. ANN was selected in this study due to its simplicity compared to the others. We also performed the previous study using ANN (Matthew, 2020), which showed good performance.

Working with other machine learning and deep learning methods can be a good start for improving the performance. This can improve from many aspects, such as a more straightforward proxy learning process and the total amount of time used to develop the proxy.

We will focus on ANN now as this study used it as a proxy model. The main question that pops up is, ”Do we need to do hyperparameter optimization?”. All study results show that we need 2 to 3 hidden layers and 15 to 20 layers. This can be a guideline if later a hyperparameter study is not preferred. The problem with the hyperparameter study is that it consumes more than half of the total time we need to build and run the proxy. Hence, not performing this will save a huge amount of time.

Nevertheless, when hyperparameter optimization ought to be performed, adding an extra hyperparameter is recommended. Dropout can be included first, as this helps to reduce the probability of overfitting. Other than that, increasing the possible range of existing hyperparameters is recommended, to enhance the proxy performance. In this study, we limited the maximum number of nodes to 20, and most proxy sections converged to 20.

Our database is normalized. It makes the output of our proxy between 0 to 1, and then it will be denormalized. During the study, we obtained a negative value of proxy output, which results in negative rate (both oil and CO_2) output. The only way to solve this is to have the lowest rate and highest rate possible as our training dataset. In this study, we tried forcing the minimum value of both FOPR and FCO_2PR to 0 even though there are no zero values in the training-validation dataset. It can be used as a reference for further study.

Last but not least, we used 75% and 25% of the data as proxy training and validation datasets, respectively. At first, we used 70% for training, 15% for validation, and 15% for testing. As we needed more data for training while maintaining validation to prevent overfitting, test data was then allocated to both of them. The robustness was then only tested with a blind test. More studies can be performed to find whether we need to test proxy with test dataset and blind test. The cost that needs to be concerned is the amount of time to generate a new dataset from the reservoir simulator.

Proxy Performance Indicator

In this study, to assess each proxy segment, MSE was used as our loss function, R^2 and the normalized absolute error were reported as proxy segments performance indicator. For testing the whole proxy robustness, APRE was used.

For improving the performance, MSE is used. In this study, MSE performs well enough. Proxy segments are first assessed with R^2 as the most common parameter to assess the fitness between proxy and learned data result. R^2 cannot determine whether the coefficient estimates and predictions are biased. For this reason, normalized absolute error was used as the second indicator of the proxy fitness.

In this study, rather than minimizing the average value, we focused more on reducing the error window. So, rather than choosing R^2 with 99% and 0.1 normalized absolute error window, we will choose R^2 with 98.8% and 0.07 normalized absolute error window. This is because the results from the previous proxy output will be used as input for the next timestep. Hence, the error will be carried over throughout later timestep.

The normalized absolute error is used because the input and output of proxy are normalized before being used. This results in 0 to 1 values. Interpreting this might be hard as the magnitude is ambiguous. Using other available errors might help, such as relative error, yet zero values need to be concerned as this will count the error to converge to infinity. Another option is to do the normalization inside ANN or using mean-standard deviation normalization.

We are moving on to quantifying the whole proxy performance. In this study, we used APRE. The main idea to use this is to easily identify whether the proxy underpredicts or overpredicts the production performance. The main problem encountered is highly negative percentage error can be neutralized with highly positive error. This will yield near-zero APRE when calculated. Taking absolute value to assess the whole performance can be implemented, and APRE can be reported as an additional assessment.

Blind Test

We used 12 run set, which is not included in the training-validation dataset. It can be increased if needed, yet the problem that needs to be considered is the total amount of time to run the blind test cases. Worth to be noted that a blind test must be taken in the range the proxy is trained.

Optimization Study

The optimization study is not studied as deeply as the other parts of this study. One of the most common things to be done for optimization study is optimizing the hyperparameters of optimization algorithm. This was not performed as many optimization processes were already performed before we obtained the representative proxy model. To tackle the convergence failure, we used a high number of generations. Optimizing the hyperparameter can be done as one of the ideas for further study. In addition, many parameters can be studied for CO_2-WAG study, yet these were not observed here. Some of these are gas injection composition, well completion, total slug volume injected, and well positioning.

PETREL has a built-in optimizer. Here, we can test whether the results using the proxy model are somewhere near the real optimum results or not. This study is not performed either as a multi-objective optimizer is not available in PETREL, especially NSGA-II. An idea that can be employed is to couple the reservoir simulator with optimization algorithm.

It is important to consider that the solution space of proxy model and reservoir simulator will be different. It is an intuitive hypothesis as the results obtained with the proxy model are not the same as the reservoir simulator model. This can be a new study to see the differences between the solution space of the proxy model and reservoir simulator. The thing to be taken into account is the amount of time to compute the solution space and the way to minimize the difference between both models.

All evaluations described here can be used as the starting point for new proxy modeling study. Moving into smaller element volume can also improve the study, starting with a simpler reservoir model such as Egg model.

6.3 The ”Real” Truth of Building Proxy Model

It seems attractive if we see the time reduction in performing the study using a proxy model rather than a reservoir model. The truth is, the ”real” time that needs to be spent developing a proxy model to perform the optimization study is not only the time used for running the whole programming script. As a starting point, **Figure 6.2** shows the workflow of employing optimization study into reservoir simulator and proxy model.

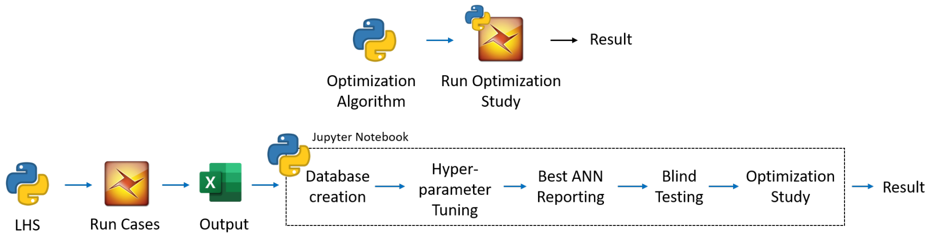

Figure 6.2: Workflow of employing optimization study using PETREL (top) and proxy model (bottom).

The blue-colored arrows involve scripting to be built. When not accounting the time needed to prepare the scripts, the study takes 17 hours and 47 minutes (Gullfaks model). 1030 evaluations were performed using the NSGA-II based on the hyperparameter configuration. The amount of time needed to solve the optimization problem directly using the reservoir simulator model will be about 6 days, 5 hours, and 38 minutes (approximating using total evaluations multiplied by reservoir model run time from **Table 4.1**). This proves the time reduction when we compare directly.

Scripting time is needed to be taken into account. In this master thesis study, we scripted everything from a blank script. It took us around two months to script for the whole study, where script adjustments are also accounted into that amount of time. The thing that needs to be noted is that it was the first time we performed a proxy model study using Python as the programming language. These two months account for the amount of time to understand the needed packages(pyDOE, Keras, and scikit-learn). Starting a new proxy modeling study with the tools we have now in place will definitely take much less time.

Mastering the packages and experience of developing a proxy model can reduce the amount of time to build the proxy. There is much room for improving this work, such as creating a GUI and main script that adjusts to the needed study objective.

So, is it Worth be Continued?

Yes. It was proved that the proxy model is able to tackle the common problems we have in conventional reservoir simulations (time and storage space needed), even with high complexity models such as the Gullfaks model used in this study. There is much room for improvement of this study to support the idea. The potential to be unleashed, such as application in history matching, is one of the main interests for proxy modeling study.

Chapter 7

Conclusions

From this study, we can conclude that developing a proxy model is one of the answers to tackle the problems associated with conventional reservoir simulation: runtime and amount of storage needed to store the results. When built in the right way, this can be strong and powerful modeling tool, as explained by Gholami (2014). Our previous study shows that it can learn decline and plateau behavior (Matthew, 2020), and here more complex cyclic and decline behavior were able to be modeled by the proxy model.

In addition to that, several points can be highlighted:

1. As a proxy modeling application, an optimization problem needs to be formulated carefully from the start of the study. It affects the reservoir model behavior that needs to be learned by the proxy model. No major problem was reported when solving a single-objective (Matthew, 2020) and multi-objective optimization study using a proxy model as reservoir model substitute. It is valid as long as the proxy robustness is confirmed before the optimization study is performed.

2. The workflow developed in the previous study can be used as a guideline for performing proxy modeling study from scratch (Matthew, 2020). In this study, the workflow is updated by combining the whole workflow with the preparation and post-proxy model development, as shown in **Figure 3.1**.

3. Proxy models were built to mimic oil production rate and CO_2 production rate from Egg Model as simple reservoir model and Gullfaks model as complex reservoir model. Several proxy segments constructed these proxy models. Each segment was made using ANN, assessed using R^2 and normalized average error. The robustness as a whole proxy was tested using APRE, both during training-validation and blind testing.

4. Higher number of samples is needed for a more complex reservoir. In addition , more proxy segmentations are needed to enhance the proxy performance. This was

proven by the amount of samples and proxy segmentations (based on the timestep) are for the Egg model (68 samples, 3 segmentations) compared with the Gullfaks model (97 samples, 5 segmentations).

5. To reach the maximum oil recovery for CO_2-WAG study, we need to have a maximum gas injection rate with a minimum water injection rate. It will reduce the amount of CO_2 stored in the reservoir. We can see a linear relationship between water injection rate, total oil produced, and total CO_2 stored in a simple reservoir model. The complex reservoir model has the same trend, but it is not linear. This trend is because of the reservoir heterogeneity that affects the sweep efficiency and CO_2 entrapment.

References

Al Adasani, A., Bai, B., 2011a. Analysis of eor projects and updated screening criteria. Journal of Petroleum Science and Engineering 79, 10–24.

Al Adasani, A., Bai, B., 2011b. Analysis of eor projects and updated screening criteria. Journal of Petroleum Science and Engineering 79, 10–24.

AlOtaibi, F.M., Zhou, X., Kokal, S.L., 2015. Laboratory Evaluation of Different Mode of Supercritical CO2 Miscible Flooding for Carbonate Rocks. URL: `https://doi.org/10.2118/177986-MS`, doi:`10.2118/177986-MS`. sPE-177986-MS.

Amini, A., Soleimany, A., Karaman, S., Rus, D., 2018. Spatial uncertainty sampling for end-to-end control. arXiv preprint arXiv:1805.04829 .

Amini, S., 2015. Developing a Grid-Based Surrogate Reservoir Model Using Artificial Intelligence. Ph.D. thesis. West Virginia University.

Carcoana, A., 1992. Applied enhanced oil recovery. Richardson, TX (United States); Society of Petroleum Engineers.

Chaki, S., Zagayevskiy, Y., Shi, X., Wong, T., Noor, Z., 2020. Machine learning for proxy modeling of dynamic reservoir systems: Deep neural network dnn and recurrent neural network rnn applications, in: International Petroleum Technology Conference, OnePetro.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6, 182–197.

Ehrgott, M., 2005. Multicriteria Optimization. Lecture notes in economics and mathematical systems, Springer. URL: `https://books.google.no/books?id=yrZw9srrHroC`.

Fleming, P.J., Purshouse, R.C., Lygoe, R.J., 2005. Many-objective optimization: An engineering design perspective, in: International conference on evolutionary multi-criterion optimization, Springer. pp. 14–32.

Gholami, V., 2014. On the optimization of CO2-EOR process using surrogate reservoir model. Ph.D. thesis. West Virginia University.

Hawthorne, S.B., Miller, D.J., Grabanski, C.B., Sorensen, J.A., Pekot, L.J., Kurz, B.A., Gorecki, C.D., Steadman, E.N., Harju, J.A., Melzer, S., et al., 2017. Measured crude oil mmps with pure and mixed co 2, methane, and ethane, and their relevance to enhanced oil recovery from middle bakken and bakken shales, in: SPE Unconventional Resources Conference, Society of Petroleum Engineers.

Himmelblau, D.M., et al., 2018. Applied nonlinear programming. McGraw-Hill.

IEA, 2017. Number of eor projects in operation globally, 1971-2017, iea, paris. URL: `https://www.iea.org/data-and-statistics/charts/number-of-eor-projects-in-operation-globally-1971-2017`.

Iman, R., 1999. Latin hypercube sampling doi:`10.1002/9780470061596.risk0299`.

Jansen, J.D., Fonseca, R.M., Kahrobaei, S., Siraj, M., Van Essen, G., Van den Hof, P., 2014. The egg model–a geological ensemble for reservoir simulation. Geoscience Data Journal 1, 192–195.

Jarrell, P.M., Fox, C.E., Stein, M.H., Webb, S.L., 2002. Practical aspects of CO2 flooding. volume 22. Society of Petroleum Engineers Richardson, TX.

Khan, Mohammad Yunus, K., et al., 2016. Water alternating gas wag optimization using tapered wag technique for a giant offshore middle east oil field, in: Abu Dhabi International Petroleum Exhibition & Conference, Society of Petroleum Engineers.

Kumar, H., Yadav, S.P., 2019. Fuzzy rule-based reliability analysis using nsga-ii. International Journal of System Assurance Engineering and Management 10, 953–972.

Lake, L.W., Lotfollahi, M., Bryant, S.L., 2019. Chapter 2 - co2 enhanced oil recovery experience and its messages for co2 storage, in: Newell, P., Ilgen, A.G. (Eds.), Science of Carbon Storage in Deep Saline Formations. Elsevier, pp. 15–31. doi:`https://doi.org/10.1016/B978-0-12-812752-0.00002-2`.

LLC, C.E., 2020. Co2 enhanced oil recovery. URL: `http://www.coreenergyllc.com/co2-enhanced-oil-recovery/`.

Matthew, D.A.M., 2020. Specialization project report - co2 eor, optimization study and smart proxy model development.

Nait Amar, M., Zeraibi, N., Jahanbani Ghahfarokhi, A., 2020. Applying hybrid support vector regression and genetic algorithm to water alternating co2 gas eor. Greenhouse Gases: Science and Technology 10, 613–630.

Nait Amar, M., Zeraibi, N., Redouane, K., 2018. Optimization of wag process using dynamic proxy, genetic algorithm and ant colony optimization. Arabian Journal for Science and Engineering 43, 6399–6412.

Negahban, S., Schou Pedersen, K., Sah, P., Basioni, M.A., Azeem, J., et al., 2010. An eos model for a middle east reservoir fluid with an extensive eor pvt data material, in: Abu Dhabi International Petroleum Exhibition and Conference, Society of Petroleum Engineers.

Ng, C.S.W., Jahanbani Ghahfarokhi, A., Nait Amar, M., Torsæter, O., 2021. Smart proxy modeling of a fractured reservoir model for production optimization: Implementation of metaheuristic algorithm and probabilistic application. Natural Resources Research , 1–32.

Satter, A., Iqbal, G.M., 2016. 17 - enhanced oil recovery processes: thermal, chemical, and miscible floods, in: Satter, A., Iqbal, G.M. (Eds.), Reservoir Engineering. Gulf Professional Publishing, Boston, pp. 313 – 337. doi:`https://doi.org/10.1016/B978-0-12-800219-3.00017-6`.

Schweidtmann, A., 2021. Multi-objective optimization algorithm for expensive-to-evaluate function. `https://github.com/Eric-Bradford/TS-EMO`.

Shpak, R., 2013. Modeling of miscible wag injection using real geological field data. URL: `http://hdl.handle.net/11250/240011`.

Verma, M.K., 2015. Fundamentals of carbon dioxide-enhanced oil recovery (CO2-EOR): A supporting document of the assessment methodology for hydrocarbon recovery using CO2-EOR associated with carbon sequestration. US Department of the Interior, US Geological Survey Washington, DC.

Whitson, C.H., Brulé, M.R., et al., 2000. Phase behavior. volume 20. Henry L. Doherty Memorial Fund of AIME, Society of Petroleum Engineers ....

Zubarev, D.I., et al., 2009. Pros and cons of applying proxy-models as a substitute for full reservoir simulations, in: SPE Annual Technical Conference and Exhibition, Society of Petroleum Engineers.

Appendix

Access the whole script and databases here:

`https://github.com/aqnanp/CO2WAG-Proxy-Study`

To get the access to the reservoir models, please contact me via:

`aqnanp@gmail.com`