DATA-DRIVEN RESERVOIR MODELING USING RECURRENT NEURAL NETWORK AND PHYSICS-BASED NETWORK MODEL

A Thesis

by

MASAHIRO NAGAO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Akhil Datta-Gupta
Committee Members,	Siddharth Misra
	Yalchin Efendiev
Head of Department,	Jeff Spath

August 2021

Primary Subject: Petroleum Engineering

ABSTRACT

We present efficient data-driven reservoir model workflows for a mature oil field involving large-scale CO_2 Water Alternating Gas (WAG) injection. The CO_2 WAG injection is conducted in more than two hundred wells in the entire field, and the operation area is spread throughout the field. Therefore, it is computationally prohibitive to implement history matching or optimization using full-field reservoir models. The objective of this study is to develop efficient data-driven approaches to optimize the CO_2 WAG operation and maximize oil recovery from the reservoir. The proposed workflows are useful for predicting future production rates and understanding the reservoir connectivity between producers and injectors.

We propose two different types of approaches. First, deep learning algorithms are utilized to develop efficient data-driven reservoir models. Long Short-Term Memory (LSTM) is a special kind of neural network architecture and has been successfully applied to many sequential and time series problems. We formulate time series problems of the production and injection histories, and the LSTM algorithm is used to forecast the future production rate and to estimate the reservoir connectivity. Second, we utilize a physics-based data-driven reservoir model, the 1D network model. The 1D network model characterizes a reservoir by a network grid system, which connects each producer injector pair via a series of 1D grid cells. Numerical reservoir simulators compute the solution of the network grid system. History matching is implemented by Ensemble Smoother with

Multiple Data Assimilation (ESMDA), and a streamline-based rate allocation optimization is implemented based on the calibrated network model.

The LSTM reservoir modeling workflow was validated using synthetic reservoir cases. It showed reasonable performance on production rates forecasting and reservoir connectivity estimation. Then, we successfully implemented this approach for a real field application. The 1D network model provided suitable history matching results for the entire field application of the mature oil reservoir. Moreover, a streamline-based rate allocation optimization was implemented, and it provided improved oil recovery from the reservoir.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my advisor, Dr. Datta-Gupta, for his invaluable suggestions throughout this research. I am likewise thankful to Dr. King for his guidance. It was also a great privilege to have Dr. Misra and Dr. Efendiev as members of my committee. I would like to extend my immeasurable appreciation to Tsubasa Onishi, Hongquan Chen, and Deepthi Sen, my wonderful mentors. Also, my gratitude goes to the MCERI research consortium members.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Datta-Gupta and Dr. Misra of the Department of Petroleum Engineering and Dr. Efendiev of the Department of Mathematics.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

The graduate study was supported by the MCERI research consortium members.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

1.1 Introduction to the Problem

The oil and gas industries are experiencing profound price collapse because of the COVID-19 pandemic. Governments worldwide were compelled to reduce business activity in order to minimize the threat of coronavirus, and the electricity demand has been reduced significantly (Madurai Elavarasan et al. 2020). Therefore, the oil and gas industries are looking to improve operational efficiency and enhance the profit on the investment.

This work presents two different types of efficient data-driven reservoir modeling approaches. The first approach uses a particular kind of machine learning algorithm called Recurrent Neural Networks (RNN). The second one is a physics-based data-driven approach, namely 1D network model. We applied our efficient data-driven workflows to a mature oil field involving large-scale CO_2 EOR.

1.2 Previous work on the mature oil field involving large-scale CO_2 EOR

The oil field was discovered about 80 years ago and has been in continuous operation ever since. Pre-EOR water injection started in mid-2016 for pressurizing the

target reservoir, and CO_2 water-alternating-gas (WAG) injection began at the end of 2016. First, a pilot operation of CO_2 WAG was conducted to gain insights into tertiary oil recovery potential via CO_2 flood in this field. Now, the CO_2 WAG injection is operated field-wide in the reservoir with more than a hundred injectors.

A simulation study on this mature oil field was presented in 2019. A hierarchical history matching was implemented for geologic-model calibration incorporating available pressure and multiphase field-level production data. The genetic algorithm was first used to match 70-year pressure and cumulative production history by adjusting pore volume and aquifer strength. Water injection data was then integrated into the model to calibrate the formation permeability. The full-field reservoir model was utilized for initializing the CO_2 WAG pilot sector model. A predictive dynamic sector model was developed, which integrates the pilot operation data. This sector model was then used to optimize the pilot CO_2 WAG operation in that region. This is the beginning of this research.

Currently, CO_2 WAG injection is conducted with more than a hundred wells, and the operation area is spread throughout the entire field. Therefore, it is computationally prohibitive to implement history matching or optimization using a realistic full-field model. Efficient reservoir modeling and optimization workflows are then necessary for effective management of field-wide CO_2 WAG operation.

1.3 Machine Learning Applications in Oil and Gas Industries

In the past few years, artificial intelligence (AI) and machine learning have been gathering one of the greatest interests in various research areas with the growth of hardware capability and data acquisition. Machine learning algorithms automatically develop a mathematical model using training data, and the trained mathematical model can be utilized to estimate future states. Because of the capability to deal with large, complex datasets and simplicity in terms of implementation, machine learning is being used for various applications, including self-driving cars, spam email filters, speech recognition, facial recognition, and language translation (Chollet 2018). In the oil and gas industries, a variety of machine learning algorithms are utilized for many applications.

Tian et al. (2019) utilized a data-mining approach to estimate downhole pressure from permanent downhole gauge data. Three different machine learning techniques were implemented: linear regression, linear regression with kernel function for capturing non-linearity, and kernel ridge regression (KRR). It was shown that KRR recovered the reservoir behaviors successfully. For example, wellbore-storage effect, skin factor, infinite-acting radial flow, and boundary effects were captured.

Nwachukwu et al. (2018) provided a workflow to optimize the well placement under the CO_2 EOR operation. In this study, a proxy model was developed using the Extreme Gradient Boosting method for estimating the total profit, cumulative oil/gas production, and net CO_2 storage. This efficient optimization workflow was demonstrated using the top layer of the SPE 10 model.

For the surface network system, neural network models were utilized for several applications. Recently, Murray et al. (2020) developed a fast and flexible optimization model for surface network systems. They created a proxy model for the surface network and the hydraulic model using neural network models. The production and injection rates were optimized simultaneously by combining the proxy and genetic algorithms, which provide recommended well status, lift gas rate, and water injection rate.

Some specialized neural networks, namely Recurrent Neural Networks (RNN), are usually used for time series problems. Long short-term memory (LSTM) and gated recurrent unit (GRU) are more sophisticated versions of RNN, and they provide much better performance in terms of predictive abilities. Several researchers sought ways to utilize these algorithms in the oil and gas industry. Cheng et al. (2020) used LSTM to predict production rates from surrounding injectors' histories in water flooding cases. In this study, a new inter-well connectivity identification method was developed using the global sensitivity analysis method, and this approach was tested on synthetic reservoir models.

Convolutional neural network (CNN) is an attractive research area in deep learning, which is almost universally used for computer vision applications. Du et al. (2020) utilized CNN to compress a large amount of data effectively, and the inter-well connectivity was estimated from compressed production and injection data for water flooding cases.

Some novel neural network architectures have already been developed in petroleum engineering applications. Li et al. (2019) developed a new neural network architecture

called Long and Short-term Time-series Network (LSTNet). This architecture is a combination of RNN and CNN. They tested various neural network models and compared them using the public production datasets of the Volve Field. The study showed that LSTNet provided a more stable prediction performance than LSTM and GRU.

In this work, we present a workflow to predict the production history from surrounding injector rates and estimate reservoir connectivity using a variable importance method. This workflow is applied to a mature oil reservoir with CO_2 EOR.

1.4 Physics-Based Data-Driven Models

In the oil and gas industries, high-definition 3D geo-cellular models are often used for reservoir management. Geo-cellular models represent geology through a set of grid blocks and their geologic properties. This reservoir model can describe very complex geological features such as fault and natural fractures, and it provides comprehensive information on pressure, temperature, and saturation. However, such detailed models usually have millions of grid blocks and are computationally expensive. Such high computational cost becomes prohibitive for history matching and optimization because they require a large number of simulations. Also, building a realistic 3D geo-cellular model takes time and tremendous effort. For these reasons, the 3D geo-cellular model is not available in many cases (Ren et al. 2019).

In order to address these problems, various data-driven reservoir models are developed. Data-driven models are calibrated using production data, and the models can be utilized for prediction and optimization. Because of simplicity, the data-driven model does not require prior knowledge of geologic properties and can be calibrated if we have enough data. Although it may not precisely capture geological features present in the reservoir, they run much faster than 3D geo-cellular models (Ren et al. 2019).

Zhang et al. (2018) proposed an efficient data-driven reservoir modeling for unconventional reservoirs. For the primary depletion dominated by transient flow behavior, we can efficiently compute the well-drainage volume with Fast Matching Method by introducing the concept of diffusive time of flight (DTOF). Through this approach, the multidimensional simulation is decoupled to a series of 1D simulations (Fujita et al. 2016). Based on the DTOF coordinate, model properties were calibrated using production data.

Albertoni et al. (2003) and Yousef et al. (2006) proposed an efficient data-driven model for EOR reservoirs, which is called the Capacitance Resistance Model (CRM). CRM provides allocation factor and time lag information between producer and injector only based on the fluctuation of production and injection histories. Moreover, the multiphase flow rate can be computed based on the fractional flow model. While CRM successfully implemented several field applications (Yousef et al. 2006; Sayarpour et al. 2009; Laochamroonvorapongse et al. 2014), it has some limitations due to simplifying assumptions. For example, the production and injection rate need to be reservoir volume.

Field operators usually record the well histories in surface condition, and flow rate in reservoir condition is not generally available. And also, this approach is applicable only for slightly compressible fluids. Hence, it is not easy to apply to gas injection reservoir cases.

Zhao et al. (2016) provided the inter-well numerical simulation model (INSIM), which solved some drawbacks of CRM. In this approach, the reservoir was characterized as a coarse model consisting of several inter-well control units. The pressure drops from injectors to producers were computed using the implicit pressure explicit saturation (IMPES) method, and then the saturation was solved via Buckley-Leverett. The transmissibility and pore volume of the inter-well units were calibrated using production histories. Zhao et al. (2016) and Zhao et al. (2020) showed promising results for large-scale field applications. However, this approach is still limited to slightly compressible flow problems, and it does not allow bottom-hole pressure (BHP) control.

The data-driven numerical flow network model (StellNet) was recently proposed by Lutidze (2018), and it solved the drawbacks of CRM and INSIM. In this StellNet approach, the reservoir is characterized by a network grid system, which connects each producer injector pairs via a series of 1D grid cells. The numerical reservoir simulator computes the solution of the network grid system. This approach allows BHP control, and any governing equations are applicable. For example, three-phase water, oil, and gas problem, or compositional reservoir model. Ren et al. (2019) combined this workflow with a

commercial reservoir simulator, and a data assimilation algorithm, ensemble smoother with multiple data assimilation (ESMDA) was implemented for model calibration.

In this study, we applied this numerical flow network model for the large mature oil field involving large-scale CO_2 EOR. This is the first application of a 1D network model for a large reservoir field with WAG injection to the best of our knowledge. We used ESMDA for the model calibration. The calibrated network model was utilized for the optimization of CO_2 WAG operation.

1.5 Research Objectives

The objective of this research was to optimize the large-scale CO_2 WAG operation in a mature oil field. We applied two different workflows for reservoir modeling and WAG injection optimization, a deep learning-based approach and a physics-based numerical network model (1D network model).

For the deep learning-based approach, we used a sophisticated version of RNN called LSTM to predict the production rate from the injection histories and to estimate the reservoir connectivity using a variable importance method. The reliability of this approach was confirmed using a 2D synthetic case and a 3D large-scale synthetic reservoir case. Then it was applied to a mature oil field application.

For the 1D network model, we first explain the concept and detailed procedure of this workflow. The proposed workflow was tested for a 2D synthetic reservoir case to confirm the capability of the approach. Then, we applied it to the mature oil field with

CO_2 WAG injection. History matching was implemented by integrating the production histories using ESMDA (ensemble smoother with multiple data assimilation). We identified the reservoir connectivity by computing the flux allocation factor based on the calibrated 1D network model. Finally, a streamline-based rate allocation optimization algorithm was applied to this 1D network model in the large-scale CO_2 WAG optimization problem.

The contents of this thesis are as follows. In Chapter II, we provide the basic idea of deep learning algorithms and explain the architectures of recurrent neural networks. Two synthetic applications are presented first, and the mature oil field application is given in the last section of this chapter. In Chapter III, we first explain the workflow of the 1D network model and validate the reliability of this approach using a 2D synthetic case. In a real field application, we present an efficient and robust workflow, including history matching and rate allocation optimization. In Chapter IV, the research is concluded with the key findings and path forward for this area of study.

CHAPTER II

APPLICATION OF RECURRENT NEURAL NETWORK

In this chapter, we explain the basic algorithm of deep learning. Deep learning is a subfield of machine learning, and it uses deep neural networks to capture the relationship between predictor variables and response variables. *Deep* in deep learning stands for the idea that a deep neural network uses successive layers. The input information goes through multiple layers, and it increasingly becomes meaningful representations (Chollet 2018).

We first explain the basic concept and architectures of neural networks. Then, we mention the neural network training algorithm, which is called backpropagation. Neural networks and almost all machine learning models have hyperparameters that users must determine before the model training. The methods for hyperparameter optimization will also be discussed. We then explain a neural network architecture named Recurrent Neural Network (RNN), which is suitable for text processing and time-series problem. The Long Short-Term Memory network (LSTM) is a special kind of RNN, and this LSTM architecture is primarily used in this study.

2.1 Background of Neural Network

2.1.1 Neural Network Architecture

The neural network architecture is inspired by the biological neural network, which is shown in **Figure 2.1**. The human neural system has around 86 billion neurons, and they are connected with 10^{14}~10^{15} synapses that pass electrical signals. Each neuron can be

considered as a receiver of electronic signals. The received signals are processed, and it decides which messages pass to the next neuron. The selected information is transmitted to the next neuron through the axon. Therefore, the neurons have three functions: receiving the electrical signals from other neurons, processing the information, and passing it to the subsequent neurons.

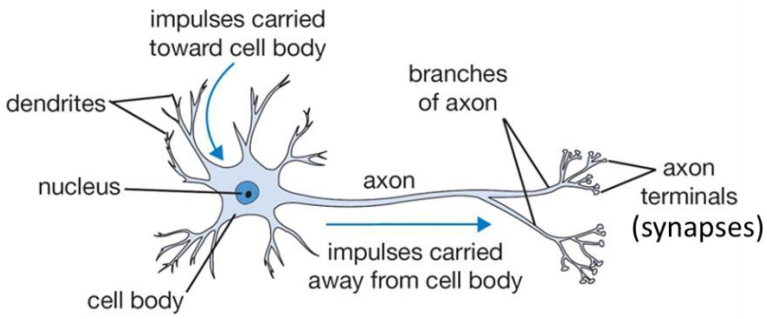


Figure 2.1: Biological Neural Networks (reprinted from Fumo 2017)

Artificial neural networks have very similar functions, and the visual representation of a single artificial neuron is provided in **Figure 2.2**. This network has one neuron, three axons from other neurons, and output information through an axon. In this system, we receive signals from three axons, which are x_0, x_1, and x_2. When the neuron gets the information, those signals are multiplied by weights, which are w_0, w_1, and w_2. These three weighted signals are summed up with bias. This bias can be considered as an intercept in linear regression cases. The neuron has a function f which is called an activation function. The activation function transforms the sum of weighted signals and

decides which information pass to the next neuron. Finally, the transformed information passes to the next neuron through the axon. During the model training process, the weights and biases are tuned to represent the correct relationship between predictors and response variables. **Table 2.1** summarizes the names of each variable and function in the artificial neural network.

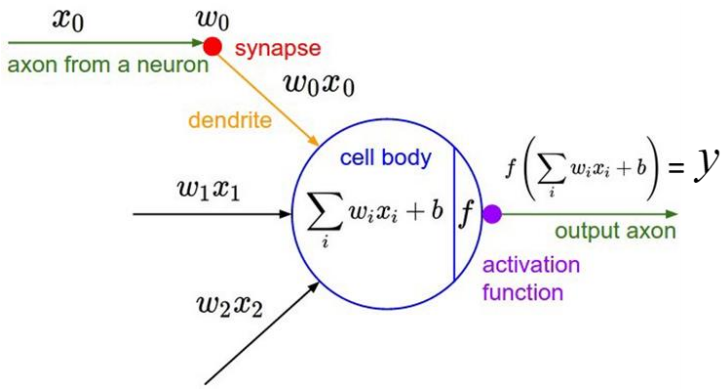


Figure 2.2: Visual representation of single-neuron artificial neural network (reprinted from Fumo 2017**)**

Table 2.1: Variables of neural networks

x_0, x_1, x_2	Input
y	output
w_0, w_1, w_2	weight
b	bias
f	Activation function
neuron	node or unit

Artificial neural networks have multiple nodes for a single layer. The visual representation of a neural network with multiple nodes is shown in **Figure 2.3**. In this representation, we have three layers: input layer, output layer, and hidden layer. The

hidden layer processes the information from the input layer and passes more meaningful information to estimate the response variables. Each node in this architecture also has the same functionality as explained in **Figure 2.2**.

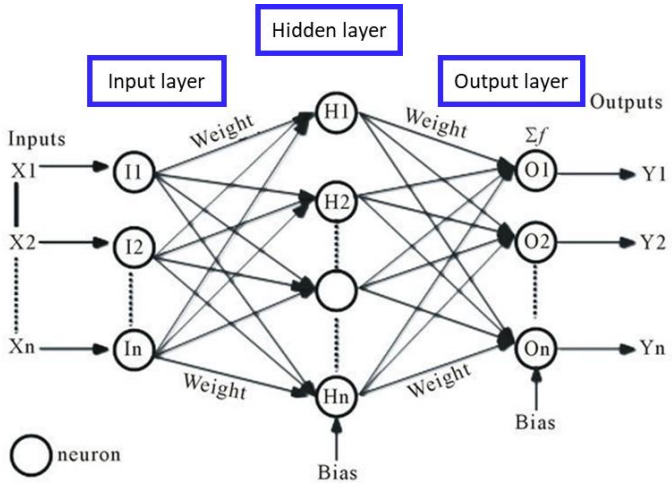


Figure 2.3: Visual representation of multilayer neural networks (reprinted from Fumo 2017)

Multiple hidden layers are used to process the information in deep learning, named deep neural networks. **Figure 2.4** shows an example of the neural network model for the digit recognition problem. The input data is the pixel of the digit image. The neural network is trained to reproduce the correct digits from the picture. It also shows the transformed image for each hidden layer. As shown in **Figure 2.4**, the converted images in the hidden layers are increasingly different from the original image and increasingly informative about the final result. It can be thought that a deep neural network is a multistage

information-distillation operation, where information goes through successive filters and becomes increasingly informative (Chollet 2018).

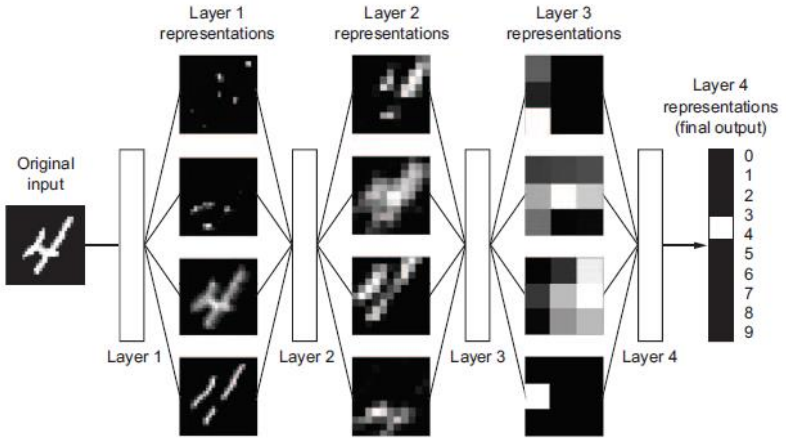


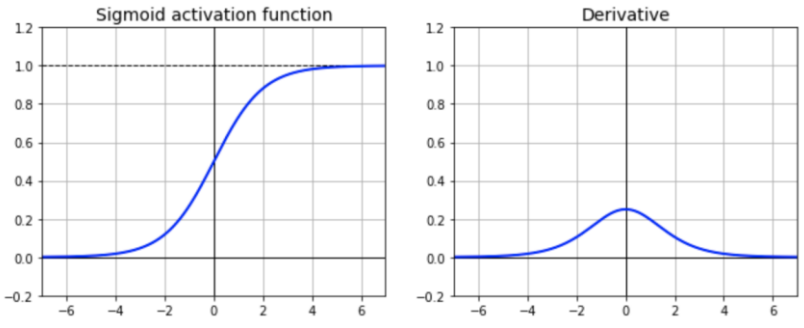
Figure 2.4: Deep neural network with transformed images for each layer (reprinted from Chollet 2018)

The task of activation functions is to transform the weighted sum of the transmitted signals and pass the informative signal to the next neuron. The activation function is one of the hyperparameters, and model users need to determine which activation function they use before the model training. Here, we introduce some standard activation functions, summarized in **Table 2.2** and **Figure 2.5**. The sigmoid function is one of the most widely used activation functions for binary classification problems. This function has a continuously differentiable S-shape. The range of this function is always from 0 to 1. For the binary classification problem, the final output needs to be between 0 and 1 because it is a probability. Therefore, sigmoid is convenient for this situation. The hyperbolic

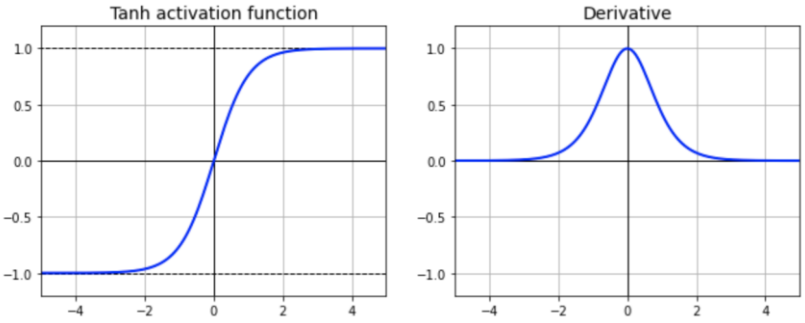
Tangent function is similar to the sigmoid function, but it is symmetric around the origin. The range is from -1 to 1. The slope of tanh is steeper than the sigmoid around the origin. The ReLU stands for the rectified linear unit, and it is one of the most widely used activation functions. The function is composed of two linear parts. It provides positive value only when the input is positive. Because of the simplicity of the operation, it runs very fast and works well for many problems (Sharma 2017).

Table 2.2: Common activation functions

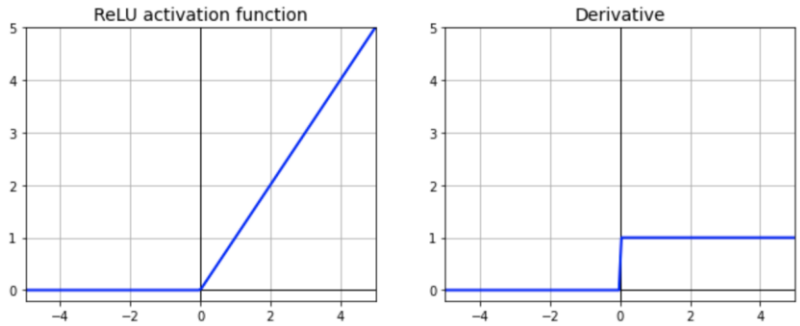
Activation Function	Equation
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
ReLU	$f(x) = \begin{cases} 0 \ \ for \ x \le 0 \\ x \ \ for \ x \ge 0 \end{cases}$



Sigmoid



tanh(x)



ReLU

Figure 2.5: Plots of the standard activation functions (reprinted from Chen 2021)

2.1.2 Training Algorithm

Based on the relationship between predictor variables and response variables, the neural network parameters (weight and bias) are optimized during the training. The training process is summarized here (Chollet 2018).

1. Prepare sets of training predictors ***x*** and corresponding targets ***y***.
2. Initialize the weight and bias of the neural network by random variables.
3. Run the neural network on ***x*** to obtain the estimation of ***y_pred***.
4. Compute the loss function based on the misfit between ***y*** and ***y_pred***.
5. Compute the gradient of the loss function with respect to all neural network parameters.
6. Move the parameters slightly in a direction to reduce the loss function
7. Repeat this process multiple times until it converges

After preparing the training data, first, initialize the model parameters using the random variables. Next, we estimate response variables ***y*** based on the initialized parameters. At this step, the estimation result is very different from the true response. From this result, compute the loss function based on the misfit. The form of the loss function is dependent on the problem type. For classification problems, cross-entropy loss is used in many cases. For regression problems, mean square error (MSE) is the most common option. Then, we compute the gradient of the loss function with respect to every neural network parameter. In this step, the network architecture applies the chain rule to the gradient computation, and this algorithm is called backpropagation. Backpropagation starts from the final loss function and goes back to the first layers. The gradient is

sequentially computed by applying the chain rule, and each parameter's contribution is estimated. One of the most popular Python libraries named ‘Tensorflow’ has a unique strategy to compute gradients called symbolic differentiation. This means that, given a chain of operation with a known derivative, we can calculate a functional form of the gradient for the chain that maps network parameter values to gradient values. If we have such a function, the backpropagation operation is reduced just to call the gradient function. Therefore, the Tensorflow library provides very efficient training performance (Chollet 2018). Once the gradients are computed, we modify the neural network parameters using the stochastic gradient descent algorithm (SGD). SGD is an intuitive approach to solve a minimization problem. However, if the problem has a high dimension and is highly non-linear, this algorithm does not give us the correct solution. **Figure 2.6** shows one example case that SGD might get the wrong solution. In this situation, if we start the SGD from the left side with a small learning rate, the optimization process would get stuck at the local minimum.

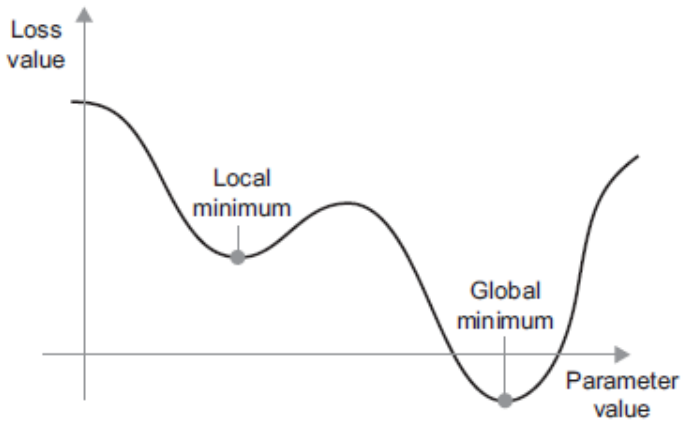


Figure 2.6: A local minimum and a global minimum (reprinted from Chollet 2018)

In order to overcome this problem, the concept of momentum is introduced. Suppose in a hypothetical scenario we put a ball at the left edge of this curve, and the ball will roll into the right. If the ball has enough momentum, it will not get stuck in a local minimum and end up at the global minimum. In other words, the momentum is not only based on the current slope value but also the current velocity (past acceleration). Mathematically, it means updating the parameters based not only on the current gradient but also on the previous parameter value. In this way, we can improve the robustness of the optimization algorithm.

2.1.3 Hyperparameter Optimization

The neural network model consists of model parameters (weight and bias) and hyperparameters. Model parameters are tuned during the training process. In contrast, the

hyperparameters need to be determined before the training, and it determines the model complexity or architecture and learning process. For example, if we increase the model flexibility, it decreases the bias of the estimation (here bias is different from the model parameters) and increases the estimation variance. If we reduce the model flexibility, it increases the bias and decreases the variance of the estimation. This relationship between model flexibility and bias/variance is called bias-variance trade-off. A visual representation of this concept is given in **Figure 2.7**. The vast amount of data points on the left plot, which are drawn by white symbols, show three different regression models: linear, quadratic, and spline. Of course, linear regression is the simplest approach, and spline is the most flexible model of these three. In the middle, the plot shows a graph of MSE against a measure of flexibility for test data and training data. The yellow dots result from linear regression, and blue dots are from quadratic regression, and green dots are from the spline model. As it can be seen, the MSE of training data continues decreasing as the model flexibility increases. On the other hand, test error increases at some particular point. On the right plot, it shows three lines: MSE, bias, and variance. At a very low flexibility point, we have a large bias and large MSE. We call this condition underfitting. As we increase the model flexibility, it decreases the bias first. At a certain point, the variance starts growing, and MSE also turns to increase. We call it overfitting. Before training the model, we need to find optimal model flexibility by tuning the model hyperparameters. We summarized the common hyperparameters in **Table 2.3**.

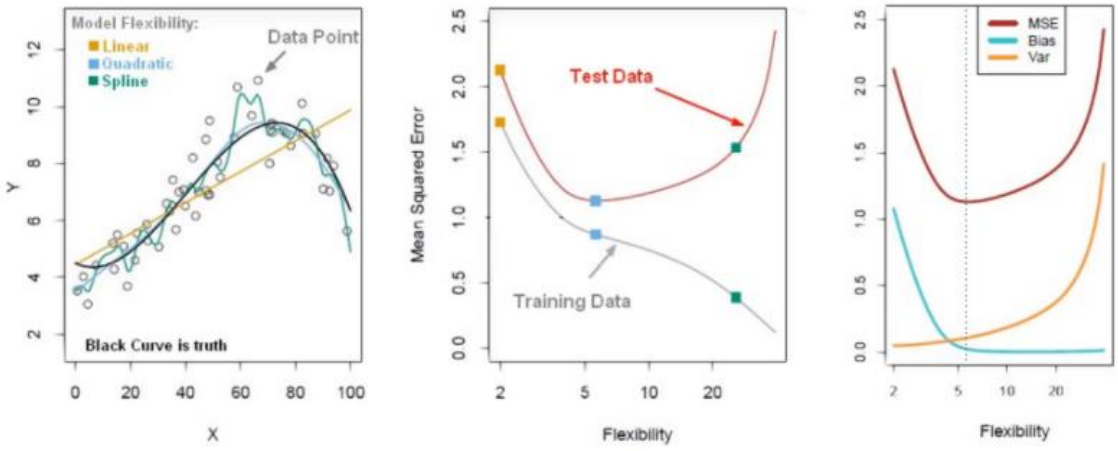


Figure 2.7: Bias and variance trade-off (reprinted from Perlato 2020)

Table 2.3: Common hyperparameters of neural networks

Number of Epochs (training iteration)
Number of hidden layers
Number of nodes
Activation function
Learning rate

One of the most common ways to optimize the hyperparameters is the validation approach.

The procedure is as follows. **Figure 2.8** provides the idea of the validation approach.

1. Split the data into three subsets, training, validation, and test data
2. Fit the machine learning model using only the training dataset
3. Assess the model performance using the validation dataset
4. Repeat these steps for several different hyperparameters
5. Select the hyperparameter that provides the best model performance

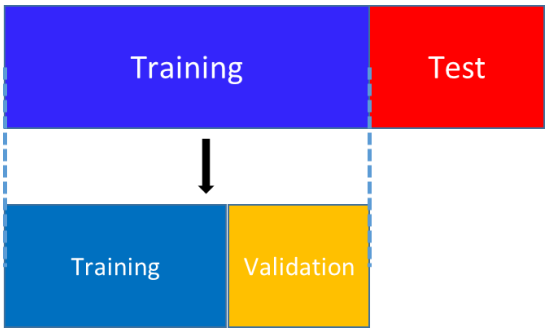


Figure 2.8: Data split for validation approach

If we need to assess the model fit performance more accurately, the k-fold cross-validation approach is usually used. **Figure 2.9** gives the idea of the cross-validation approach. This example has four folds, and each fold is split into training data and validation datasets. For each fold, we fit the model using the training data and assess the fitting performance using the validation dataset. Then, we average the model performance for each fold. This approach is more expensive than the validation approach, but it provides stable optimization results.

1	2	3	4
Validation	Training	Training	Training
Training	Validation	Training	Training
Training	Training	Validation	Training
Training	Training	Training	Validation

Figure 2.9: Visual representation of k-fold cross-validation

2.2 Neural Networks for Time Series Data

This section explains the deep learning models suitable for text processing (sequences of words) and time-series data. In text processing, we are processing the sentence word by word as we are reading. Moreover, we are keeping memories of what came before at the same time. Biological intelligence processes information incrementally, and it is built and updated based on the current information and the past information (Chollet 2018). Recurrent Neural Networks (RNN) are suitable for text processing and time-series data problems because it adopts the same principle with the biological intelligence.

2.2.1 Recurrent Neural Networks

Figure 2.10 shows the architecture of the single RNN unit. RNN processes the sequences by iterating through the sequence elements and maintaining a state that contains the past information. In other words, RNN is a type of neural network that has an internal loop (Colah 2015). In this diagram, 'X_t' is the input, and 'h_t' is the output of the RNN unit.

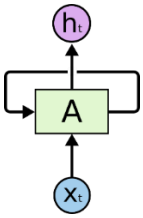


Figure 2.10: Single RNN unit (reprinted from Colah 2015)

From this picture, it may not be clear how the RNN processes the sequential information. However, once we expand the loop, it would make more sense. This is provided in **Figure 2.11**. Now, we can see that we have sequential input, and every RNN unit provides output (h_t) and state. Here, the state is the information passed from the previous timestep. For the simple RNN architecture, the state is just an output of the previous timestep, and it is passed to the next timesteps' RNN unit. In this way, the memory of the previous timestep information is given to the next timestep.

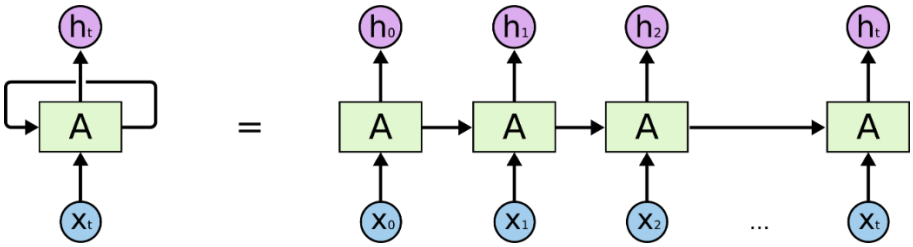


Figure 2.11: Expanded RNN architecture (reprinted from Colah 2015)

The meaning of each variable in this figure and its relationship with our applications are

$X_0, X_1, X_2, \ldots X_t$: input at t = 0, 1, 2, …, t (For example, injection rate of surrounding wells)

$h_0, h_1, h_2, \ldots h_t$: output at t = 0, 1, 2, …, t (For example, production rate of a target producer)

One of the problems of simple RNN models is that they may not capture long-term dependencies. For instance, if we would like to estimate the last word of the sentence, and if the clue to the answer exists in the very first word, we need to maintain the long-term memory. RNN is not suitable for this type of problem. **Figure 2.12** shows the idea of the RNN limitation. RNN usually cannot maintain the long past information because the information needs to go through the RNN unit every time step. The past information gradually decreases as we go further.

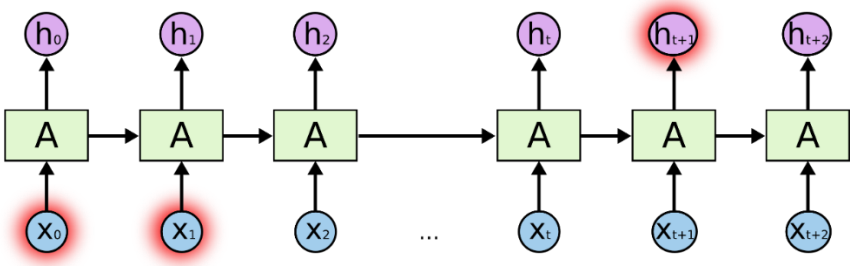


Figure 2.12: RNN limitation, long-term memory (reprinted from Colah 2015)

2.2.2 Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory networks (LSTM) can overcome the limitation of RNN. It is a special version of RNN and was introduced by Hochreiter et al. (1997). It is one of the most widely used network architectures for sequential data problems.

Figure 2.13 shows the visual representation of LSTM architecture. As it can be seen, LSTM has a more complex system that includes four functions in the unit. We explain those functions one by one.

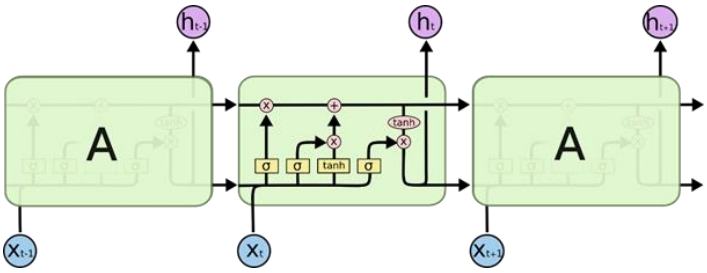


Figure 2.13: Architecture of LSTM (reprinted from Colah 2015)

One of the important elements of LSTM is the cell state. In **Figure 2.14,** we can see a horizontal line running at the top of the diagram. The cell state runs the entire chain (all timestep) with only some minor interaction. Then, it can transmit long past information (Colah 2015).

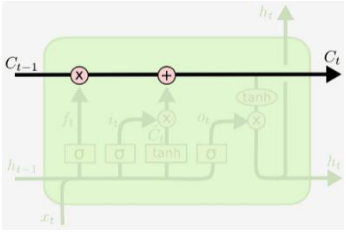


Figure 2.14: LSTM cell state (reprinted from Colah 2015)

LSTM has elements that remove or add the information to the cell state, which are called gates. In this section, the gates are described as a picture in **Figure 2.15**. The gate has a sigmoid activation function. As mentioned in this chapter, sigmoid has a range of 0 to 1. If the gate provides 0, it means that no information is allowed to go through the gate. If the gate gives the value of 1, all data is permitted through the gate (Colah 2015). In this section, we explain all the gates that we have in LSTM.

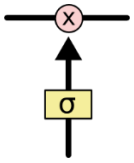


Figure 2.15: A Gate in LSTM (reprinted from Colah 2015)

The first step is to decide how much information we will maintain from the previous cell state, and this gate is named a forget gate. **Figure 2.16** provides the portion of the architecture.

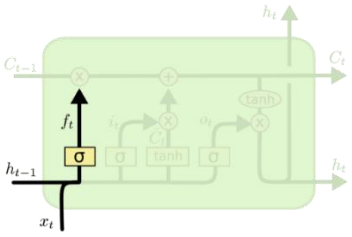


Figure 2.16: Forget gate in LSTM (reprinted from Colah 2015)

The equation of the forget gate is (Colah 2015).

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \tag{2.1}$$

Where,

σ : sigmoid function
h_{t-1} : hidden state of the last timestep
x_t : input of current timestep
W : weight
b : bias

The next step is to decide how much information goes through the cell state from the input of the current timestep and the hidden state of the last timestep. This gate is named an input gate. This step has two different layers: the sigmoid layer and the tanh layer (Colah 2015). **Figure 2.17** shows the visual representation of this computation, and equations are given below.

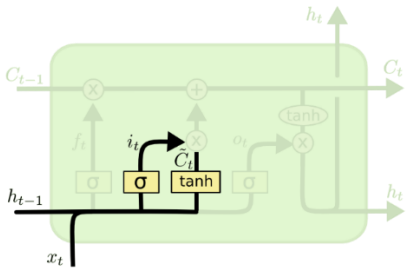


Figure 2.17: Input gate in LSTM (reprinted from Colah 2015)

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \tag{2.2}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right) \tag{2.3}$$

Where,

σ : sigmoid function
$\tanh$: Hyperbolic tangent
h_{t-1} : hidden state of the last timestep
x_t : input of current timestep
W : weight
b : bias

The computation result from the first and second steps is now gone through to the cell state. The new cell state is computed by simple multiplication and summation. **Figure 2.18** shows the visual representation of this computation.

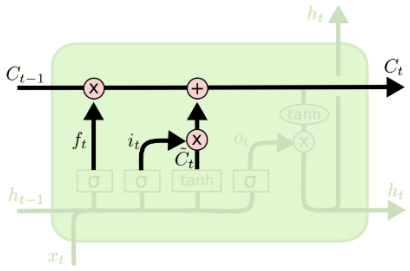


Figure 2.18: Update of cell state in LSTM (reprinted from Colah 2015)

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.4}$$

Where,

C_{t-1} : cell state of the last timestep
f_t : calculation result of forget gate
i_t : calculation result of input gate from sigmoid
$\tilde{C}_t$: calculation result of input gate from tanh

Finally, we compute the output of this unit. The output is based on the cell state, and it is filtered by the input of the current timestep and the hidden state of the last timestep (**Figure 2.19**).

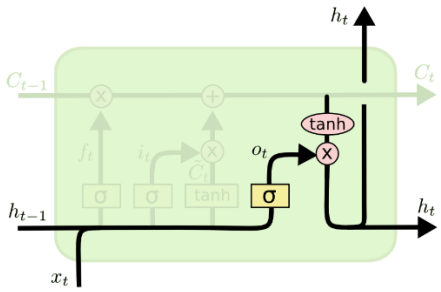


Figure 2.19: Output gate of LSTM unit (reprinted from Colah 2015)

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{2.5}$$

$$h_t = o_t \times \tanh\left(C_t\right) \tag{2.6}$$

Where,

σ : sigmoid function
$\tanh$: Hyperbolic tangent
h_{t-1} : hidden state of last timestep
x_t : input of current timestep
W : weight
b : bias

There are some different versions of specialized RNN. One of the most popular architectures is Gated Recurrent Unit (GRU), which was introduced by Cho et al. (2014).

In our application, we primarily use LSTM for developing the data-driven reservoir model and predicting the future production rate.

2.2.3 Neural Network Model Interpretation Method

Model interpretability is always preferred for any problem. Because the higher the interpretability of the machine learning model, the easier it is for someone to understand the underlying relationship between predictor variables and response variables (Molnar 2021). However, most of the robust machine learning models do not have suitable interpretability because there is a trade-off between model flexibility and interpretability (James et al. 2013). **Figure 2.20** provides the visual idea of the trade-off. Although the prediction performance of the Lasso and least square method are not very robust for practical problems, those models are simple and have better interpretability. On the other hand, the random forest, boosting, and neural networks are robust machine learning models in terms of prediction performance, but the interpretability is less than that of Lasso and least square method because of the model complexity.

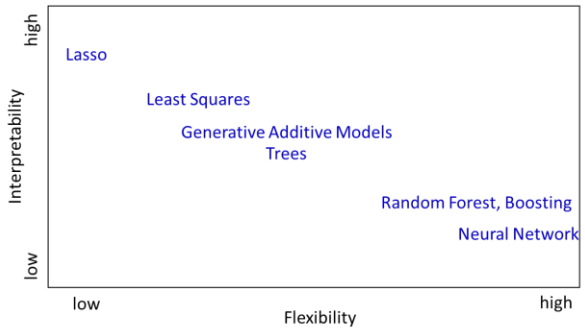


Figure 2.20: Trade-off between model flexibility and interpretability (reprinted from James et al. 2013)

The neural network models provide robust prediction performance, but it does not provide suitable model interpretability. It is better to use model agnostic interpretation methods for obtaining underlying relationships between predictors and response variables. This approach is separated from the models, and it can be applied to any type of machine learning model (Molnar 2021).

Breiman (2001) introduced a variable importance method called permutation feature importance for random forests, and Fisher et al. (2018) presented a model-agnostic version of the permutation feature importance method (Molnar 2021). Kareepadath Sajeev (2020) tested permutation feature importance for estimating the reservoir connectivity for a simple streak reservoir case. It showed reasonable agreement with the connectivity estimation from the streamline allocation factor. In our study, a similar feature importance method was tested for several synthetic cases and a real field case.

We measure the sensitivity of each injector with respect to the target producer by using perturbation variable importance. The procedure to estimate the sensitivity is as

follows: First, we train a neural network model using given datasets. Then, add some noise to one specific injector, and estimate the target production rate using the pre-trained model. This step is repeated for every injector, and we obtain the sensitivity for each injector. In our application, the predictor variables are injection rates, and response variables are the target producer's rates. Therefore, the variable importance result represents the reservoir connectivity between producers and injectors. **Figure 2.21** shows the idea of the variable importance method in our applications.

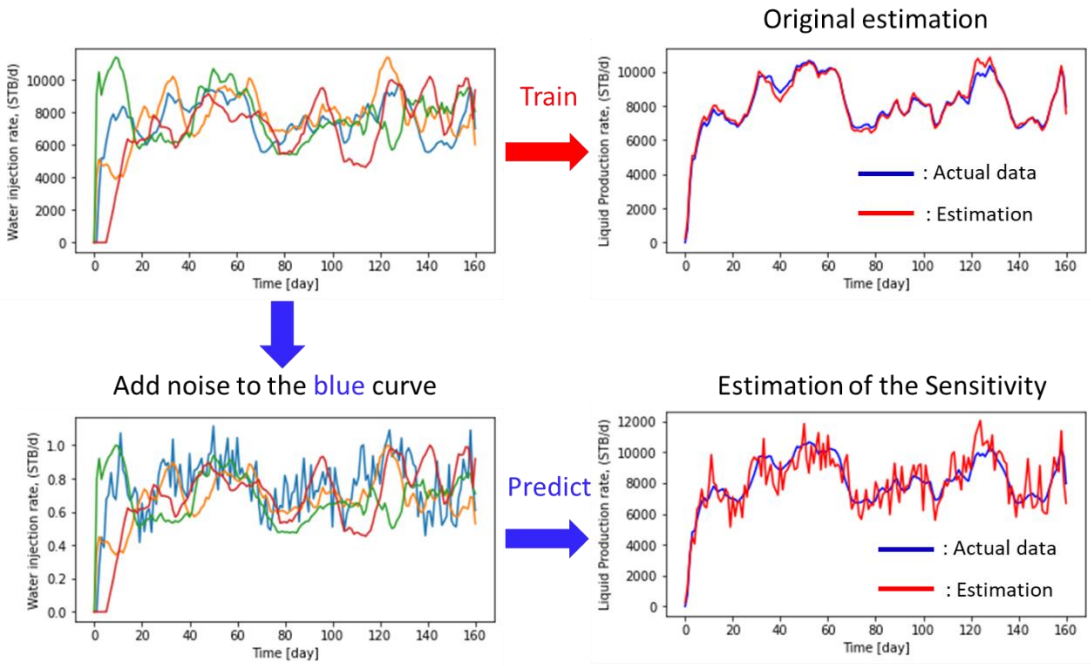


Figure 2.21: Variable importance method

2.3 Synthetic 2D Reservoir Application

In this section, we explain the workflow of the machine learning reservoir modeling approach using a simple 2D reservoir case.

2.3.1 Model Description

Figure 2.22 provides the permeability map of the synthetic reservoir. It is a channel reservoir with highly heterogeneous permeability and uniform porosity (permeability in the x-direction is identical to that of the y-direction). The grid dimension is 50 x 50 x 1, and the size of each grids is 50 x 50 x 10 [ft]. There are four injectors and nine producers in the reservoir. All injectors have continuous waterflood. The simulation was conducted using a commercial simulator (ECLIPSE).

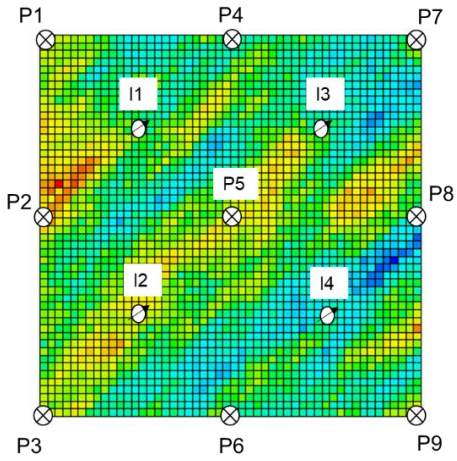


Figure 2.22: Synthetic 2D reservoir model (permeability map)

For all producers, we had bottom-hole pressure constraint, and for injectors, we had water injection rate constraint, and the fluid flow system was oil and water 2 phase flow problem. The production and injection histories of some selected wells are provided in **Figure 2.23**.

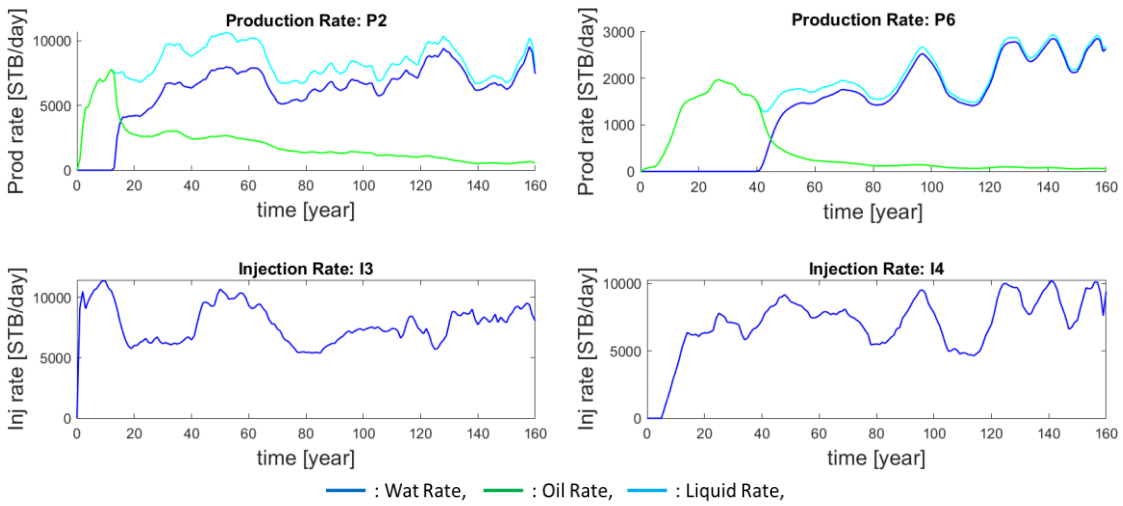


Figure 2.23: Well histories of selected wells

2.3.2 Training Neural Network Model

First, we made the sets of input data and output data. In this problem, we estimated the liquid production rate based on the water injection rates. We developed a neural network model for each producer. In this example, there are nine producers. Hence, we developed nine neural network models.

Before the training, we prepared the input and output data for the neural network model. All data were scaled to obtain the range 0 to 1 because it helps the model learn efficiently. Then, we took sliding windows for the input data. Then, the multiple timesteps of injection rates were used for estimating the production rate for every timestep. **Figure**

2.24 shows the idea of the sliding window. The horizontal line is the time axis. For each estimation, all red data are used for the estimation of the blue timestep. Therefore, we can consider the time lags between the injectors and producers by taking sliding windows. The window size is determined from the hyperparameter optimization before the training.

Figure 2.24: Sliding window (reprinted from Chou et al. 2016)

The LSTM model was used for this problem. We implemented it in Python using Keras and Tensorflow libraries. The training formulation is summarized in **Table 2.4**. We used a single-layer LSTM model. Mean absolute error (MAE) was minimized during the training, and we calibrated the weight and bias in the model. For an optimization algorithm, we used Adam optimizer, which is one of the most popular optimizers in deep learning (Kingma et al. 2015).

Table 2.4: 2D synthetic case training formulation

Neural Network Model	LSTM
Objective function	Mean absolute error (MAE)
Optimizer	Adam
Number of LSTM layers	Single-layer
Learning rate	0.01
Activation function (LSTM)	tanh
Number of NN parameters	~400000

In the hyperparameter optimization, we optimized three parameters, the number of nodes in the LSTM layer, window size, and the number of epochs (training iteration). We used the default of the learning rate and the activation function, which are shown in **Table 2.4**. First, we split the dataset into three parts: training, validation, and test dataset. The data split is given in **Figure 2.25**. In this figure, we use the dataset shown on the bottom for the hyperparameter optimization. We have 72 days of training data and 48 days of validation data. The grid search method was applied to optimize the hyperparameters. We selected the parameter that provided the minimum MAE.

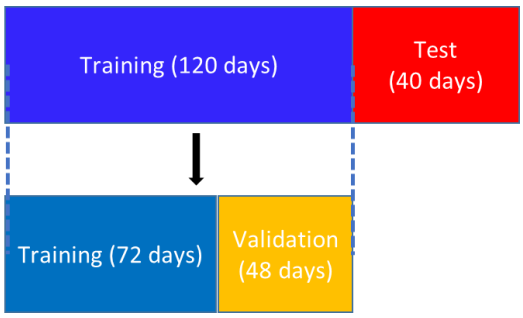


Figure 2.25: Data split for generating training, validation, and test dataset

We implemented hyperparameter optimization, model training, and test error estimation by following these steps.

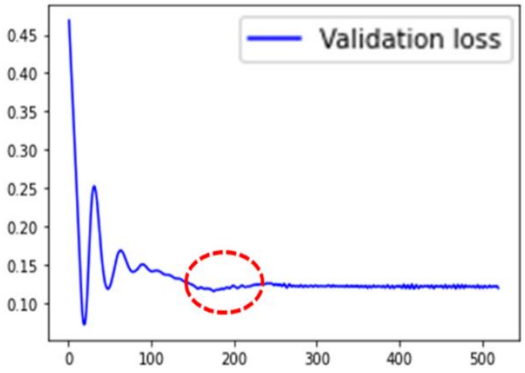
1. Fit the model using 72 days of training data with different hyperparameters
2. Assess the fitting performance using the validation dataset
3. Select the optimum hyperparameters that provide the best training performance.

4. Again, fit the model with optimized hyperparameters using the 120 days of training dataset.
5. Estimate the test error using the 40 days of the test dataset.

Figure 2.26 shows an example of hyperparameter optimization results.

MAE		Node size					
		30	50	100	200	300	500
Window size	1	2.04E-02	2.12E-02	2.05E-02	2.07E-02	2.09E-02	2.12E-02
	2	1.61E-02	1.74E-02	1.63E-02	1.62E-02	1.52E-02	1.59E-02
	5	2.21E-02	2.04E-02	1.99E-02	1.96E-02	2.05E-02	2.02E-02
	7	2.18E-02	2.44E-02	2.11E-02	2.36E-02	2.37E-02	2.34E-02
	10	3.56E-02	2.16E-02	2.69E-02	2.63E-02	2.40E-02	2.64E-02

Optimize the number of nodes



Optimize the number of epochs

Figure 2.26: Hyperparameter optimization (2D synthetic case)

The top table shows the MAE value for some different numbers of node and window size. It shows that 300 nodes and window size 2 are the optimum parameters that provide the best prediction performance. The bottom plot shows the MAE for the validation dataset against the number of epochs. As shown in the figure, if we continue the iteration, first,

the MAE value steeply decreases, then it becomes flat. The ideal number of epochs for this model is around 150~200. We followed this optimization procedure for all nine neural network models.

2.3.3 Prediction result

As explained in the last section, we estimate the liquid production rate based on the surrounding injector's histories. **Figure 2.27** provides the prediction results of all producers. The blue lines are the actual liquid rate, the red lines are the training result, and the green lines are the prediction result. We got reasonable matches for most of the producers. Therefore, it is confirmed that LSTM can roughly predict future liquid production rates for this simple reservoir case.

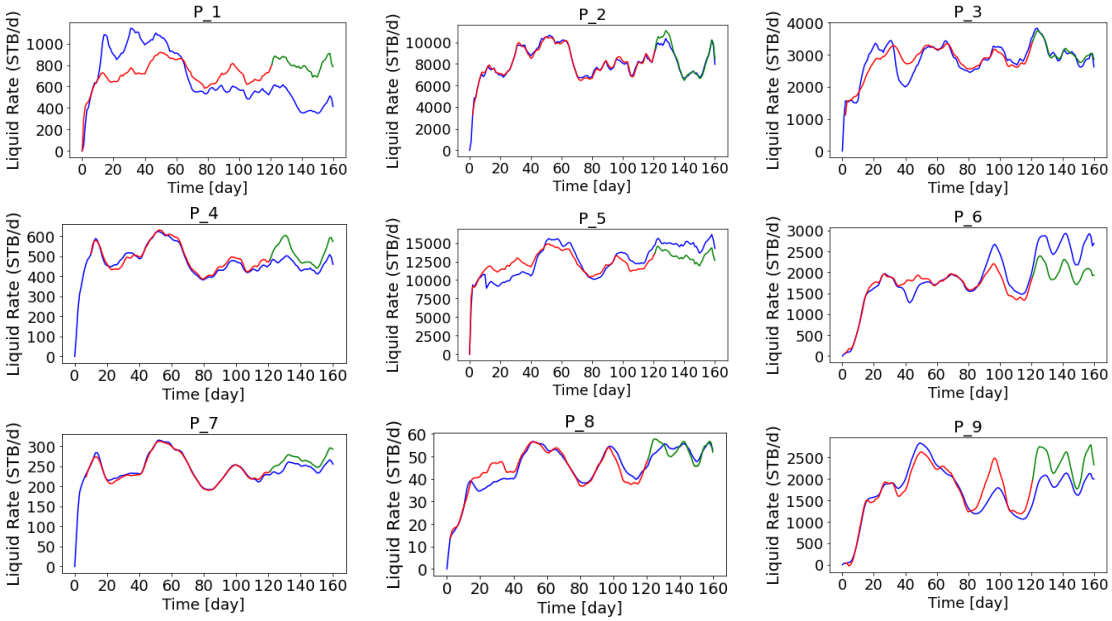


Figure 2.27: Liquid production rate forecasting (2D synthetic case)

There is a slight deviation for P1, P6, and P9. A couple of reasons for these disagreements can be considered as follows: For P1, the pressure support is dominated by I1. However, from the permeability map, we can see very high reservoir connectivity between I1 and P2 (**Figure 2.22)**. Hence, most of the injected fluid goes to P2 mainly. In such a case, it is not easy to capture the pressure signal at P1. For P6 and P9, the main pressure support is from I4. But we can see that I4 is completed in the low permeability formation (**Figure 2.22**). In that case, the signal of injection rate history is smoothed out, which makes it difficult to capture the injectors' signals from the producers' history.

2.3.4 Reservoir connectivity identification from variable importance

Then, we tested the reservoir connectivity identification from the variable importance. The procedure of variable importance is as follows. First, the neural network model is fitted for the entire history (160 days). Then, we add noise for one injector's rate and estimate the target production rate using the trained model. If the perturbed injector provides a large sensitivity value, it indicates that it has large connectivity with the target producer. **Figure 2.28** shows the reservoir connectivity map from LSTM variable importance and the allocation factors from streamline tracing. (We used Destiny, MCERI in-house software, for streamline tracing).

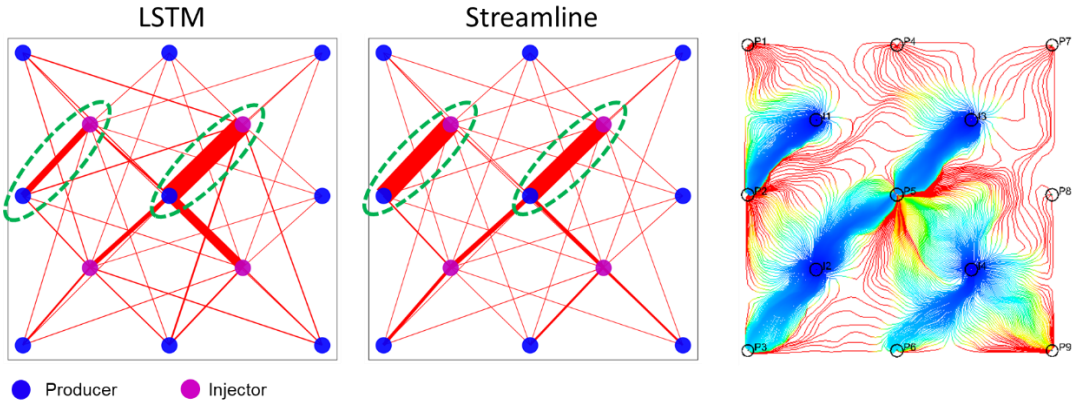


Figure 2.28: Reservoir connectivity from LSTM and streamline (2D synthetic case)

The left plot is from the LSTM variable importance, and the middle plot represents the allocation factors from the streamline tracing (SLN). The width of the lines corresponds to the sensitivity, which is computed as the root mean squared error between the actual liquid rate and the estimated rate based on perturbed injectors' data. As shown in the figures, there are two large connections around the center (indicated by green circles), and both LSTM and SLN captured it properly. Although there is a slight difference between LSTM and SLN, the overall trend of the connectivity matches correctly. It can be helpful diagnostic information for reservoir management, for instance, the rate allocation optimization.

2.3.5 Comparison with another Neural Network Architecture

We tested different neural network architectures for the same application. In this section, we tested Gated Recurrent Unit (GRU) and compared the liquid production rate forecasting performance with LSTM. The formulation and procedure to train the neural

network are the same with the LSTM application. Because of the simple feature of GRU, the number of neural network parameters is fewer than that of LSTM. The training formulation is summarized in **Table 2.5**. **Figure 2.29** shows the prediction results of nine producers.

Table 2.5: 2D synthetic case training formulation

Neural Network Model	GRU
Objective function	Mean absolute error (MAE)
Optimizer	Adam
Number of GRU layers	Single-layer
Learning rate	0.01
Activation function (GRU)	tanh
Number of NN parameters	~300000

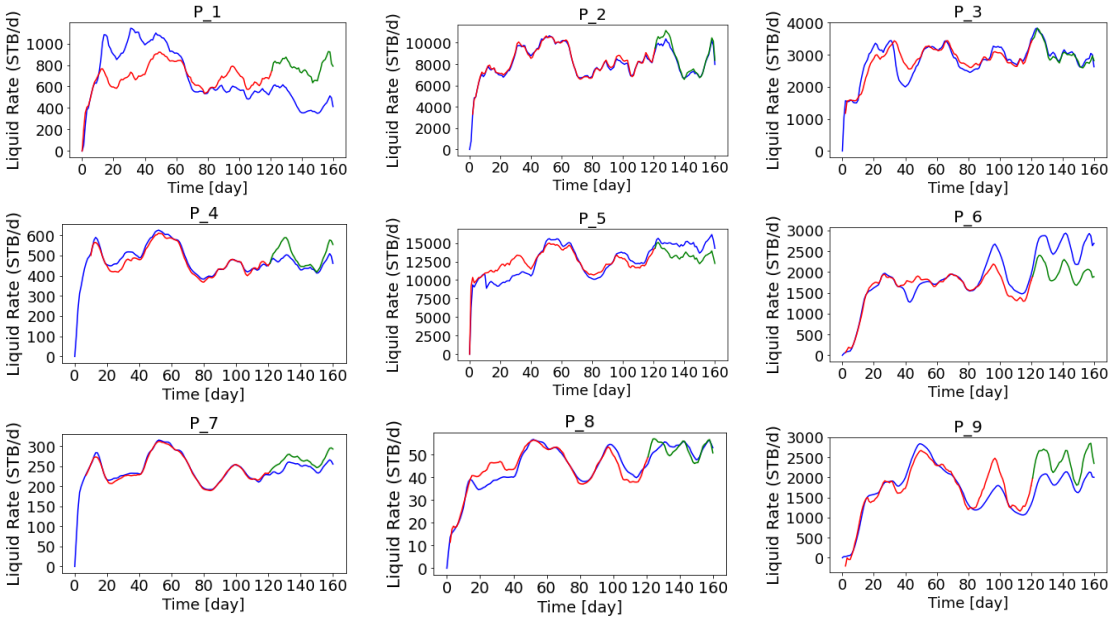


Figure 2.29: Liquid production rate forecasting from GRU (2D synthetic case)

As it can be seen, the prediction performance is very similar to that of LSTM, even if GRU contains a smaller number of parameters. Hence, for this specific case, GRU is preferred because a smaller number of parameters tend to prevent overfitting, and it is computationally efficient. However, if the problem becomes a more complicated system, it may be preferred to use more complex architecture.

2.4 Synthetic 3D large field case application

In this section, we extend the deep learning workflow to the large-scale 3D reservoir and confirm the ability of the LSTM data-driven reservoir modeling approach.

2.4.1 Model Description

Figure 2.30 shows the permeability map of a 3D synthetic reservoir. This case has 200 x 400 x 30 grid system and the entire size is 2000 x 4000 x 60 [ft]. In this reservoir, we have 22 producers and 23 injectors. The well placement is also given in **Figure 2.30**. We have heterogeneous permeability (permeability in the x-direction is identical to that of the y-direction) and uniform porosity. The simulation was conducted using a commercial simulator (ECLIPSE). For producers, we had bottom-hole pressure constraints, and for injectors, we had water injection rate constraints. There was continuous waterflood for all injectors, and the fluid flow system was oil and water 2 phase. The production and injection histories of some selected wells are provided in **Figure 2.31**.

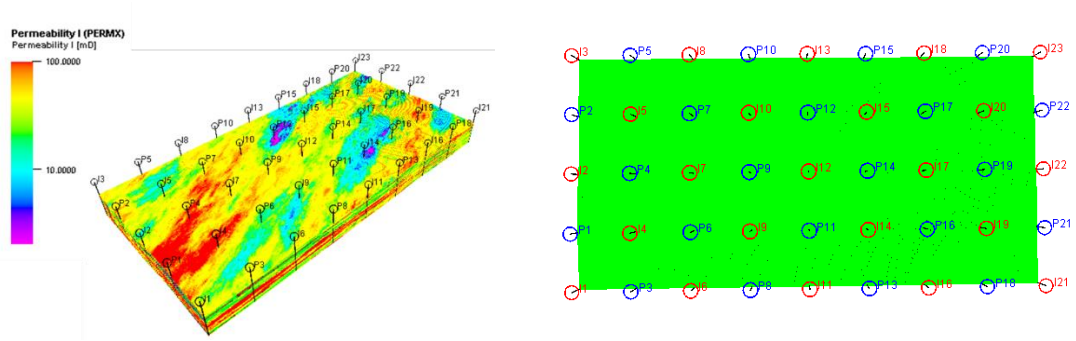
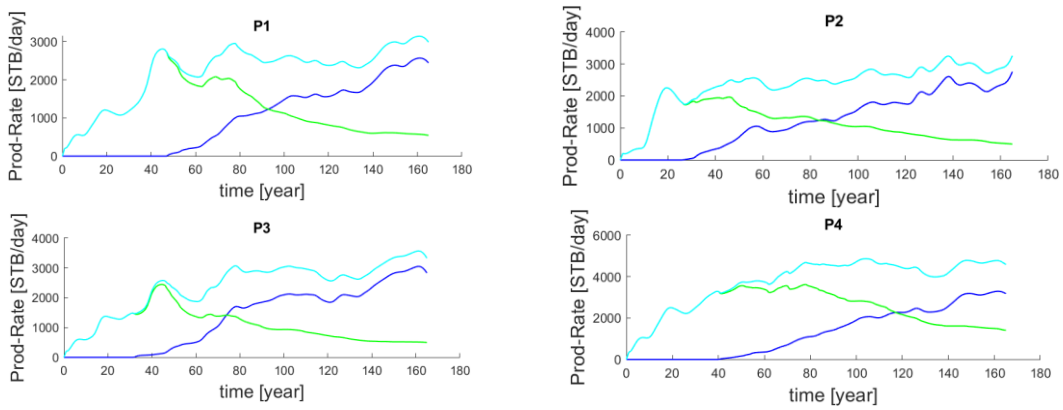
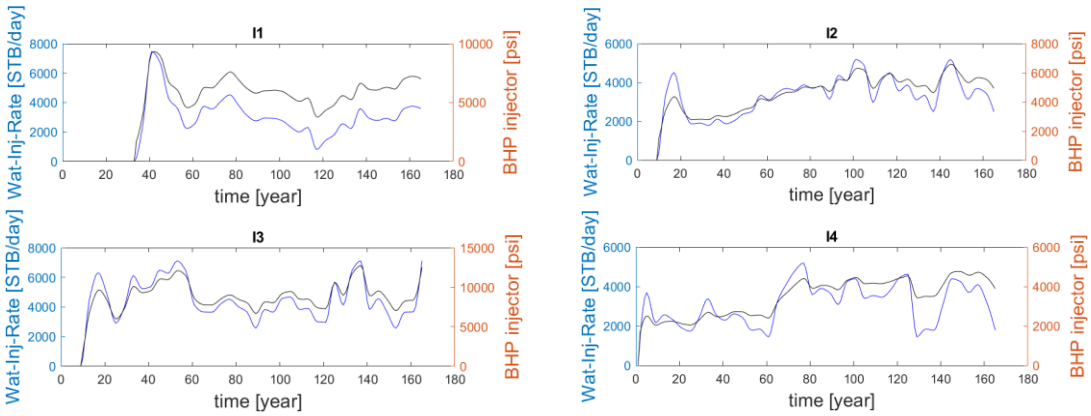


Figure 2.30: Reservoir model description for 3D synthetic case



Producer histories



Injector histories

: Wat Rate, : Oil Rate, : Liquid Rate, : BHP

Figure 2.31: Production and injection histories at selected wells

Our objective was to estimate the liquid production rate based on the surrounding injector's water rate and bottom-hole pressure and to identify the reservoir connectivity.

Table 2.6: Input and output data of the neural network model (3D synthetic case)

Input data	Water injection rate Injectors BHP
Output data	Liquid production rate

2.4.2 Training Neural Network Model

First, we made the sets of input data and output data. The reservoir model includes 23 injectors and 22 producers. Then, we needed to determine which injector's histories we include in the neural network for predicting the target producer's liquid rate. One of the simplest approaches uses search radius, and **Figure 2.32** represents the general idea. The search radius for 'P9' includes four injectors I7, I9, I10, and I12. In this case, the well rates of these four injectors are used to estimate the liquid production rate of P9. The length of the search radius needs to be determined by users. We can conduct numerical experiments for some different search radius values and determine the optimal one. In this synthetic case, the wells are equally spaced. Then, the length of the search radius in **Figure 2.32** is reasonable for this problem.

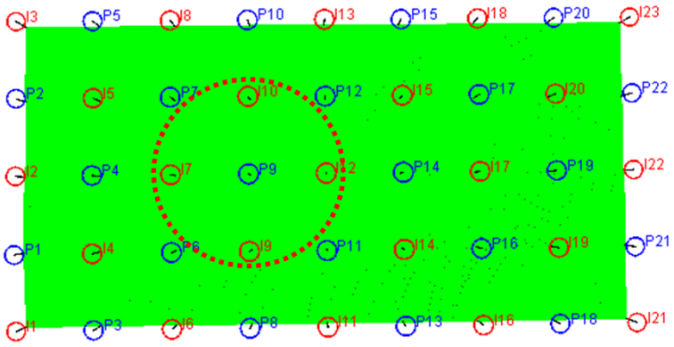


Figure 2.32: An example of search radius in synthetic case

Same as 2D synthetic case, the injection and production histories are scaled to obtain the range of 0 to 1 to help the neural network learn effectively. The LSTM model was used again, and it was implemented in Python using Keras and Tensorflow libraries. The training formulation is the same as the 2D synthetic application, which is summarized in **Table 2.7**. Hyperparameters were optimized using the grid search method same as the last case: number of LSTM nodes, window size, and number of epochs.

Table 2.7: Training formulation (3D synthetic case)

Neural Network Model	LSTM
Objective function	Mean absolute error (MAE)
Optimizer	Adam
Number of LSTM layers	Single-layer
Learning rate	0.01
Activation function (LSTM)	tanh
Number of NN parameters	~350000

2.4.3 Model Estimation Result

As explained in the last section, we estimated the liquid production rate based on the surrounding injector's histories. **Figure 2.33** provides the prediction results of selected wells. The blue lines are actual liquid rates, the red lines are training results, and the green lines are the prediction result. We obtained reasonable matches for most of the producers. For P20, there is a slight deviation, but it follows the trend of actual data.

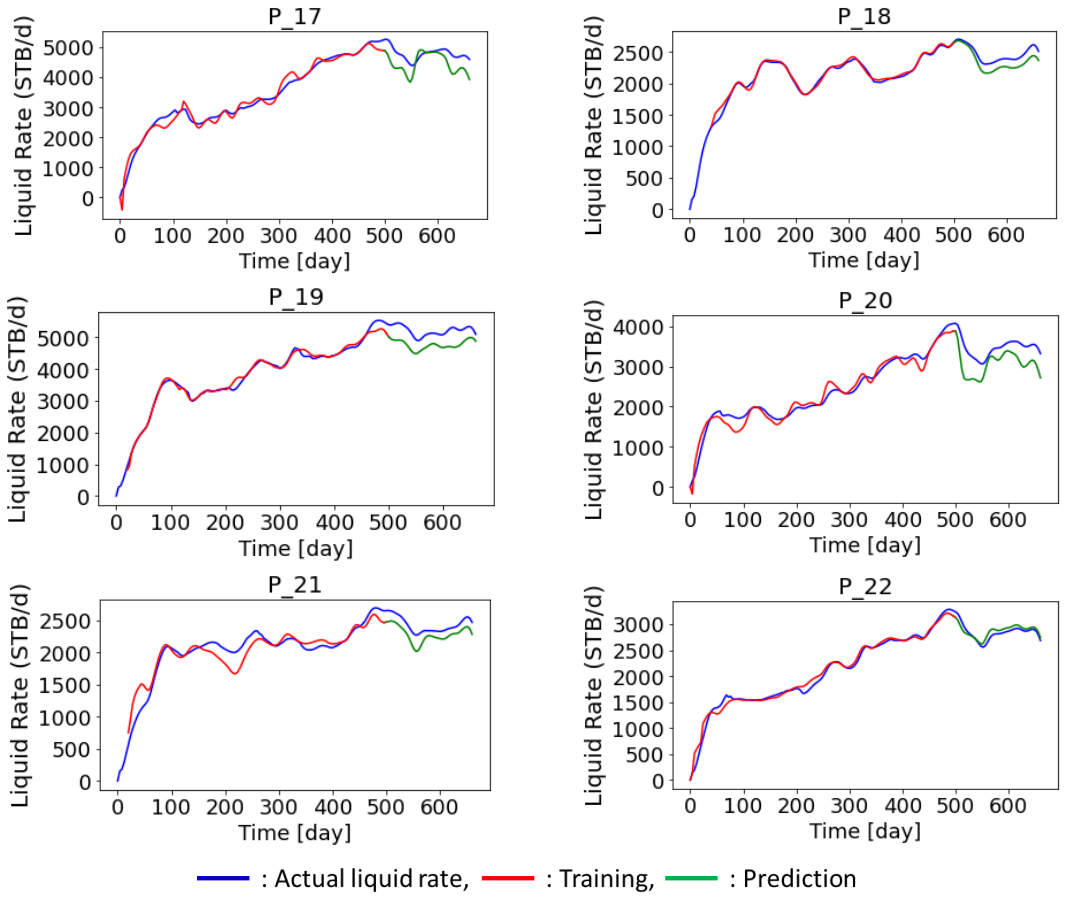


Figure 2.33: Liquid production rate forecasting (3D synthetic case)

Next, we estimated the reservoir connectivity using LSTM variable importance. **Figure 2.34** shows the reservoir connectivity map from LSTM variable importance and the allocation factor from streamline tracing. (from Destiny, MCERI in-house software).

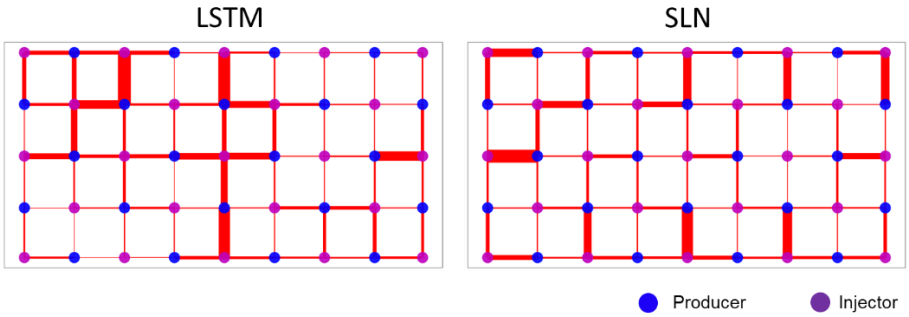


Figure 2.34: Reservoir connectivity from LSTM and streamline (3D synthetic case)

The left plot is from the LSTM variable importance, and the right plot shows the allocation factor from the streamline tracing (SLN). The width of the lines corresponds to the sensitivity, which is computed as the root mean squared error between the actual liquid rate and the estimated rate based on perturbed injectors' data. As it can be seen, there is some difference in the reservoir connectivity map between LSTM variable importance and streamline allocation factor. A couple of reasons for the disagreement is considered. First, the injection and production profiles were similar for several wells in the field. If there are the same injection profiles for different wells, it provides completely the same connectivity values with the target producer. Because the data-driven approach only looks at the well histories for identifying the connectivity. However, this is not true for reservoir

connectivity. Next, the producer and the injector relationship might not be captured based only on the well rate profiles. In high conductive channel formation, the fluctuation in the injection rate history can be identified in the target producer's rate history as signals. However, for a low permeability reservoir, the signals from the surrounding injectors may not be captured because the production rate profiles are smoothed out due to the low permeability. The 2D synthetic reservoir case, presented in the last section, is a channel reservoir with a highly heterogeneous permeability distribution. In such a case, it is easier to capture the reservoir connectivity because it has obvious correlations through channel formation between injectors and producers.

2.5 Application for a mature oil field involving large-scale WAG injection

In this section, we present the machine learning reservoir model for the mature oil reservoir field with CO_2 WAG operation. Our objective of this application was to predict the gas production rate based on the surrounding injector's histories and identify the reservoir connectivity using the trained neural network model. The reason for selecting the gas production rate as a response variable is because it is relatively easier to capture the relationship between producers and injectors. The injected gas tends to form a channel flow because the gas viscosity is very small, and gas saturation was almost zero at the initial condition. Then, the injected gas flows as if it is a tracer in this field.

2.5.1 Model Description

In this application, we decided to pick up a specific region in the reservoir, which has relatively long and continuous well histories. The quality and amount of data are the keys to the success of machine learning applications. **Figure 2.35** shows the entire field map with the selected region to apply the machine learning model approach. The red dots represent the injectors, and the blue dots are producers. The brown dots are the points of conversion wells that convert from producer to injector or injector to producer at a specific time. The size of the dots corresponds to the values of the cumulative gas rate. Large dots have a large amount of cumulative gas rate. **Figure 2.36** shows the gas production rates, producer's tubing head pressure, gas injection rates, and water injection rates for the wells in this region.

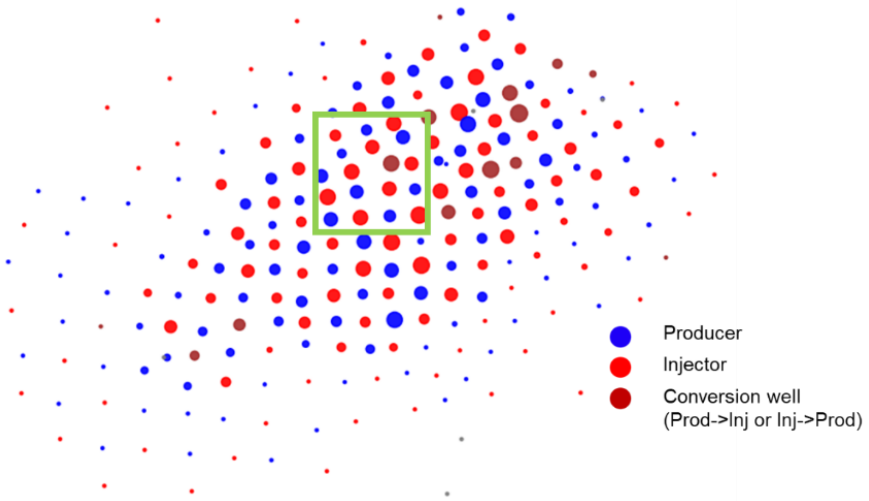


Figure 2.35: Full-field map with the test region (real field application)

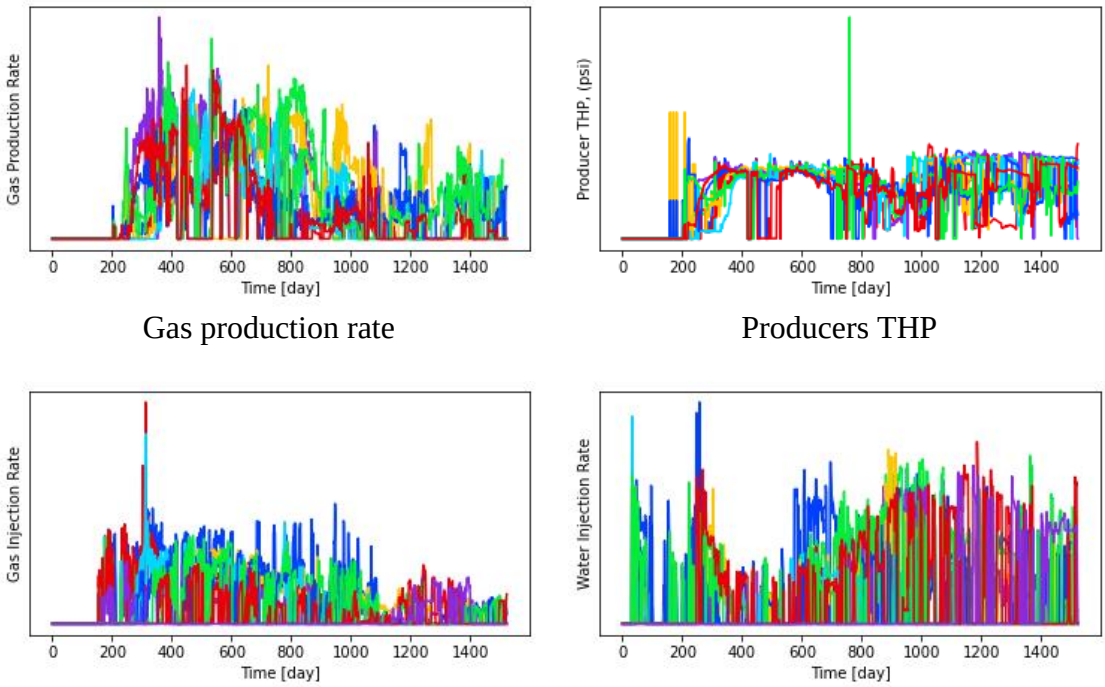


Figure 2.36: Well histories in the region of interest (real field application)

2.5.2 Training Neural Network Model

The green rectangle region has eight producers. Hence, we developed eight different neural network models in this application. As explained in the synthetic case application, we defined the search radius and determined which injectors we included for each neural network model. We need to use a reasonable length of the search radius and selected 1200 [ft] for this application.

The list of the input for the neural network model is given in **Table 2.8**. We tested various input data combinations, and we found that this combination gets the better result

for this specific problem. Since these are all field data, they contain some unrealistic values. As a data preprocessing, we removed all unrealistic well rates and scaled the well histories into 0 to 1. We used LSTM for this application in the Python libraries, Keras and Tensorflow. The training formulation was the same as the synthetic application, and it is summarized in **Table 2.9**.

Table 2.8: Input and output of the model (real field application)

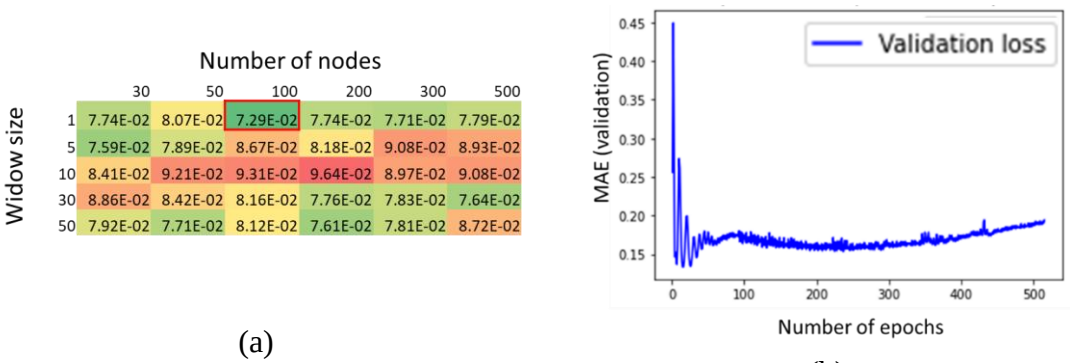
Input	Gas injection rate Cumulative gas injection rate Water injection rate THP of target producer
Output	Gas production rate

Table 2.9: Training formulation (real field application)

Neural Network Model	LSTM
Objective function	Mean absolute error (MAE)
Optimizer	Adam
Number of LSTM layers	Single-layer
Learning rate	0.01
Activation function (LSTM)	tanh
Number of NN parameters	~260000

We implemented hyperparameter optimization for three parameters: number of LSTM nodes, window size, and the number of epochs. The procedure for optimization is the same as the synthetic case. Please refer to the 2D synthetic case application for more detailed steps. We provide optimization results for one neural network model in **Figure**

2.37. **Figure 2.37** (a) shows the MAE values for different window size and the number of nodes. The result shows that window size 1 and 100 LSTM units are the optimal combinations. Next, the optimal number of epochs was determined for each neural network model. The neural network model was trained using the optimized node size and window size, and we tracked the MAE of the validation dataset with a different number of epochs. **Figure 2.37** (b) gives one example of a plot with MAE against the number of epochs. In this example, the optimal number of epochs is around 200~300. The very small number of epochs shows small MAE values, but it also fluctuates significantly. It indicates that the neural network model is still trying to capture the relationship between predictor variables and response variables.



(a) (b)

Figure 2.37: Hyperparameter optimization. (a): MAE with different window size and the number of nodes. (b): MAE against the number of epochs

2.5.3 Model Estimation Result

Figure 2.38 shows the prediction result for the eight producers in the test region.

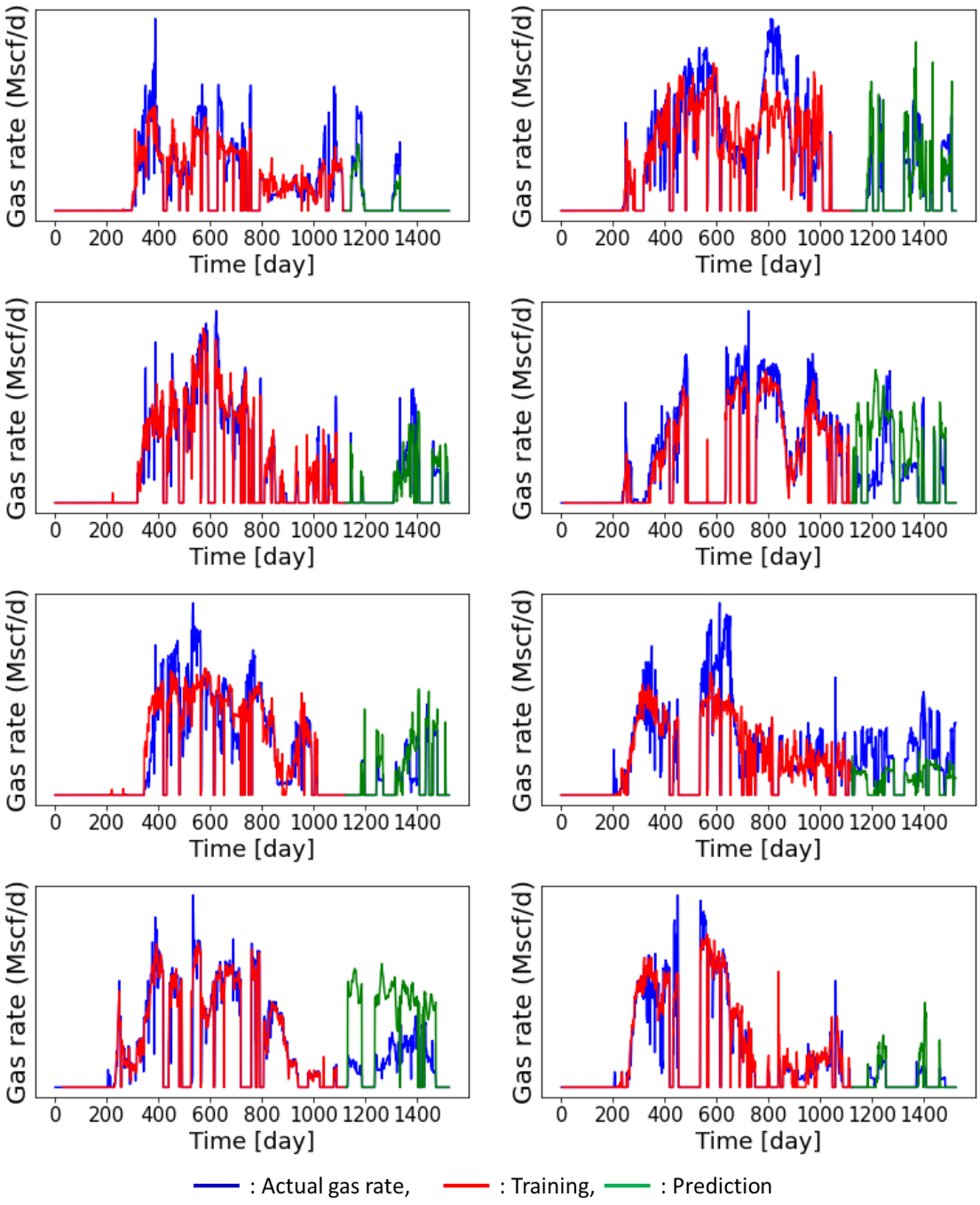


Figure 2.38: Gas production rate forecasting (field application)

The blue lines are actual gas production rates, and red lines are the fitting result of the training part, and the green lines are the prediction part. At some wells, we obtained reasonable matches with actual data. However, we can also see the deviation from actual gas production rates for some wells. The reasons for the disagreement are summarized as follows.

- The reservoir is much more heterogeneous than the synthetic case, and we may not see obvious signals that indicate the well pair connectivity in the well histories.
- The injection and production histories include noise because it is real field data.
- There were many workover operations during the history, and many wells were re-completed for a different reservoir zone frequently. This field has significant vertical transmissibility anisotropy. Then, once the well is re-completed in the different zone, it generates a new reservoir connection with the surrounding other wells.

Next, we estimated the reservoir connectivity using the LSTM variable importance. **Figure 2.39** provides two reservoir connectivity maps with search radius 1200 [ft] and 900 [ft].

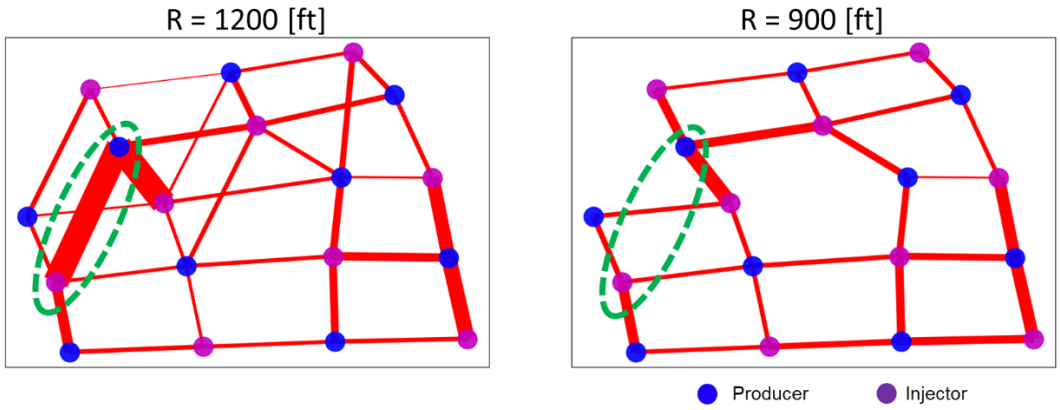


Figure 2.39: Reservoir connectivity map from variable importance

The width of the connection lines corresponds to the reservoir connectivity, which is computed as the root mean squared error between the actual gas rate and the estimated rate based on perturbed injectors' data. These connectivity maps provide some useful information for reservoir management, like rate allocation optimization. As it can be seen, the connectivity map from search radius 900 [ft] is missing the large reservoir connectivity, which is indicated by the green dashed circle. From this analysis, we can say that the result of variable importance is very sensitive to the length of the search radius. We need to investigate some optimization methods to determine the optimal search radius that gives a correct reservoir connectivity map.

CHAPTER III

APPLICATION OF 1D NETWORK MODEL

3.1 Methodology

This section presents the basic concept and the workflow of 1D network model. First, we mention how to construct the flow network grid system. And then, the strategy to map the flow network system into commercial reservoir simulation. Finally, a history matching algorithm and the rate allocation optimization methods are presented.

3.1.1 Constructing the Flow Network Grid System

Figure 3.1 provides an example of a flow network system. At the first point, we have well location information. There are four injectors at the edges and one producer at the center. In the flow network system, producers and injectors are connected by series of 1D grid cells. In this case, we have four producer injector pairs. Hence, four connections are built into the system. Every grid block has its grid volume, pore volume, transmissibility, and some other geologic properties. In this network grid system, each well is completed in the well cell. The bottom right figure is a zooming picture of the center producer. Because the center well is completed in the well cell, it allows flow communication between the connection and well also among each connection. Hence, it forms one-dimensional flow within each connection and multidimensional flow at the well cell. This is similar to the concept of streamline bundles. It forms one-dimensional flow within each streamline bundle, and there are many flow directions at the well point. It is

an important concept because it enhances the degree of freedom and the history matching quality.

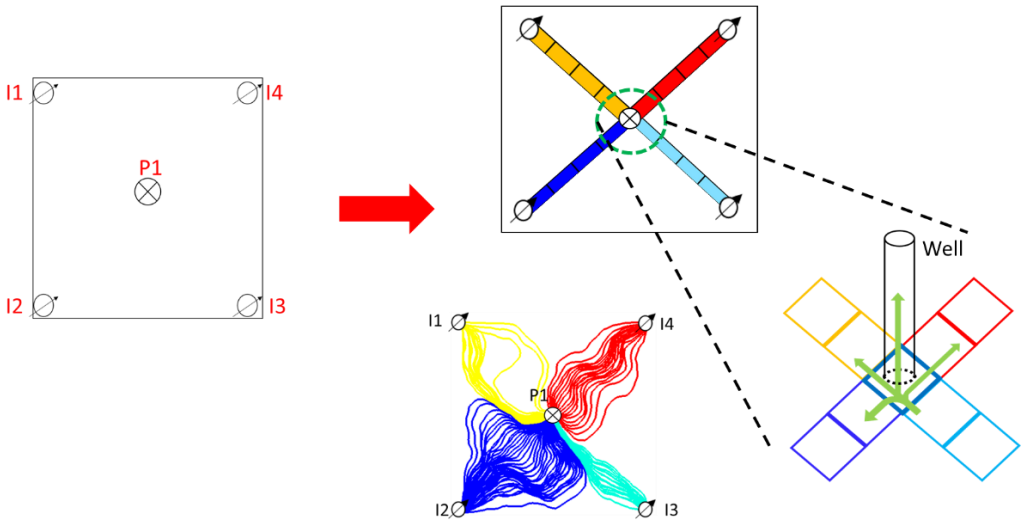


Figure 3.1: Flow network grid system

The generation of the flow network system is not dependent on geology, and it only depends on the well location. In case we have a large number of producers and injectors, not all injector producer pairs should be connected because it increases the number of grid cells and increases computational cost. It possibly connects very far injector producer pairs, even though it has little physical connection. We need some rules to generate a proper flow network system. The easiest way is to use a search radius and make the connections only within the search area. **Figure 3.2** shows an example that has 23 injectors and 22 producers. The search area for injector "I12" is represented at the center, which contains four producers. Then, "I12" has only four connections in this

example. The bottom right picture is the constructed flow connection map using the search radius. The length of the search radius needs to be pre-determined.

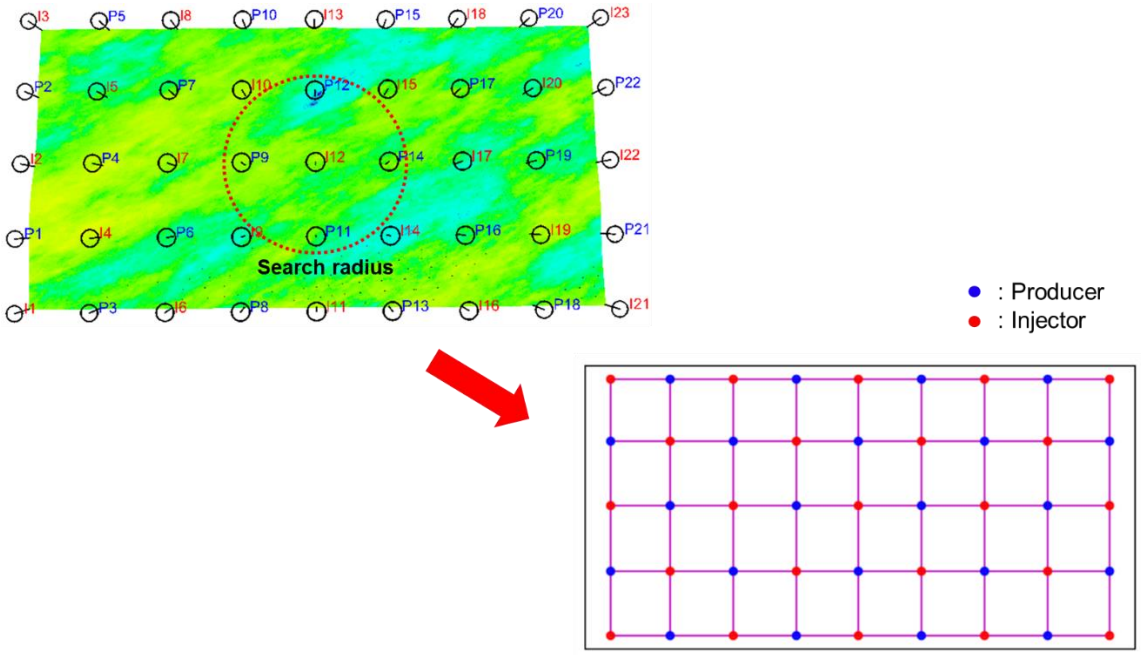


Figure 3.2: Flow network generation for multiple well pairs

If we have a reliable prior geologic model, the streamline tracing helps generate the flow network grid system. This procedure allows us to include some prior knowledge of the geologic model. The steps to generate a flow network using streamline are as follows.

1. Run the reservoir simulation only a single timestep with uniform injection rates and uniform production rates for all wells.

2. Trace streamlines based on the simulation.

3. Generate connections based on the number of streamlines for each injector producer pair.

3.1.2 Mapping the Flow Network into a Commercial Simulation Grid

The flow network grid system needs to be converted into a standard simulation format. Ren et al. (2019) provide a way to map the flow networks into a commercial simulator format. The flow network can be converted into a 2D cartesian grid, as shown in **Figure 3.3**. It is the same example in **Figure 3.1**. Each row of the 2D cartesian grid is corresponding to each connection. In these grids, the transmissibility in the vertical direction is always set to zero. Because each connection communicates only through the well cell, it should not flow directly from other connections. The small arrows on the producer represent non-neighbor connections (NNCs). It connects each network to the well cell. The number of grid blocks for each connection needs to be determined by the users. The decision is dependent on the problem that we solve, but Ren et al. (2019) suggested using ten grid blocks for each connection as a reasonable start point of history matching.

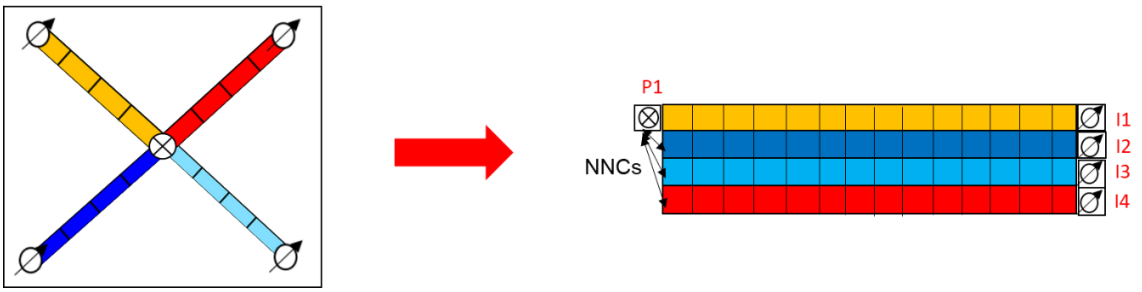


Figure 3.3: Conversion from flow network to 2D Cartesian numerical simulation grid

We can implement history matching and calibrate each grid cell's geologic properties based on this grid system. In many cases, we choose to calibrate pore-volume, permeability, and relative permeability. The calibrated properties can be used to estimate reliable information, like reservoir connectivity and oil in place. However, it is not expected to produce geologic properties comparable to laboratory experiment data or actual field data. Because the reservoir is now simplified using flow networks, and the calibration procedure tunes the parameter based on this network grid system.

3.1.3 History matching algorithm

We use ensemble smoother with multiple data assimilation for the model calibration method (ESMDA) (Emerick et al. 2013). This algorithm can deal with many parameters, such as the grid properties for large reservoir models. ESMDA is a specialized version of ensemble smoother (ES). ES assimilates all observed data only once. On the other hand, ESMDA assimilates all observation data multiple times with an inflated measurement error at each assimilation step. By assimilating the same data multiple times, the matching quality is significantly improved. This methodology is motivated by the fact that the single

data assimilation and multiple data assimilation are mathematically equivalent in linear-Gaussian cases. For n_m dimensional model parameter vector m and n_d dimensional observed data vector d_{obs}, the analyzed model parameter vector can be written as follows.

$$m_j^a = m_j^f + C_{MD}^f \left(C_{DD}^f + \alpha_i C_D \right)^{-1} \left(d_{uc,j} - d_j^f \right) \tag{3.1}$$

Where,
m : model parameter
C_{MD}^f : cross-covariance matrix between data and model parameters
C_{DD}^f : auto-covariance matrix of observed data and predicted data
C_D : covariance matrix of data measurement error
$d_{uc,j}$: vector of perturbed observation
d_j^f : Predicted observed data
j : index of ensembles
a : analysis
f : forecast
α : inflation coefficient

Based on the above equation, the observed data is assimilated, and model parameters are calibrated. The number of ensembles and inflation coefficient need to be determined before the history matching implementation. (For a detailed description of ESMDA, referred to Emerick et al. 2013.)

3.1.4 Injection Rate Allocation Optimization

We utilized the calibrated 1D network model for the rate allocation optimization. Two different approaches were applied for optimization. First, we applied finite volume-based flow diagnostics for computing the well allocation factor and generating the

reservoir connectivity map. The second approach was the streamline-based rate allocation optimization.

The finite-volume-based approach is an alternative to streamlines for obtaining flow diagnostics information, and this approach is already used for many applications (Shahvali et al. 2012; Shook et al. 2009; Møyner et al. 2014). The main advantage of this approach is the applicability to unstructured grids and ease of implementation (Shahvali et al. 2012). Flow network grid system has a lot of NNCs, as is mentioned in the last section. In such a case, the finite volume-based flow diagnostics method is convenient. We applied it to both a synthetic case and a large-scale CO_2 WAG injection reservoir application.

We implement the rate allocation optimization by using a streamline-based approach. Tanaka et al. (2017) proposed a fast and robust derivative-free workflow that improved economic values by optimizing rate allocation using streamline-based methods. This workflow requires much less computational cost than population-based techniques such as genetic algorithm (GA) and provides comparable optimization performance with GA (Tanaka et al. 2017). We used this optimization workflow for the calibrated flow network model.

3.2 Synthetic Application

3.2.1 Model Description

We used the 2D synthetic case to verify the 1D network model's capability. **Figure 3.4** shows the 2D synthetic reservoir case, which has four producers at the edges and one injector at the center. The reservoir model is discretized by 50 x 50 x 1, and the size of the reservoir is 3000 x 3000 x 100 [ft]. It has heterogeneous permeability and uniform porosity. Throughout the well history, the injector has continuous water floods.

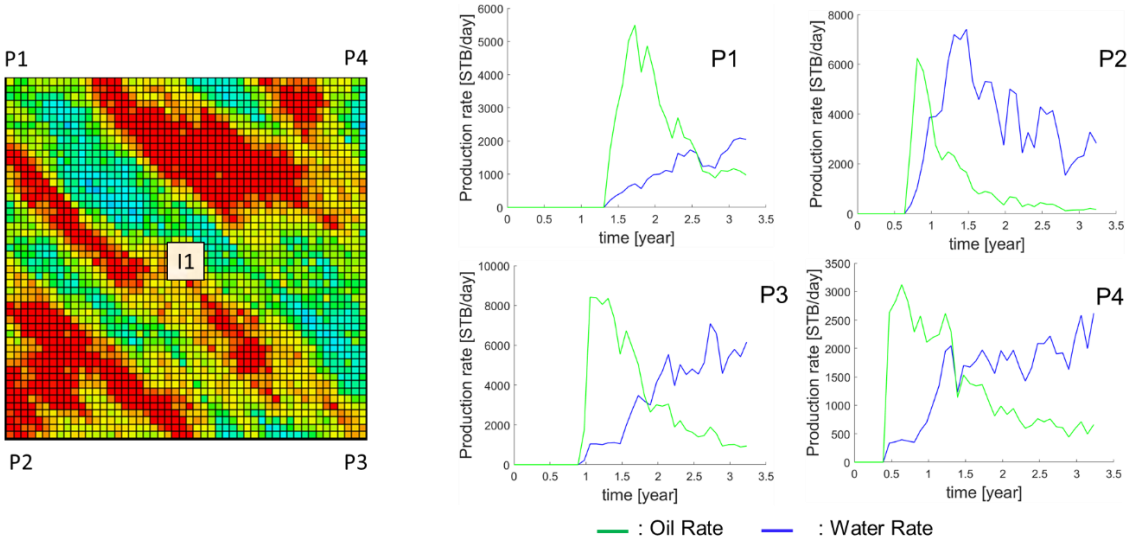


Figure 3.4: Permeability map of 2D synthetic reservoir and observed data

3.2.2 Flow Network Grid Generation and History Matching Formulation

First, we need to generate the flow network grid system. The network and the 2D simulation grid system are described in **Figure 3.5**.

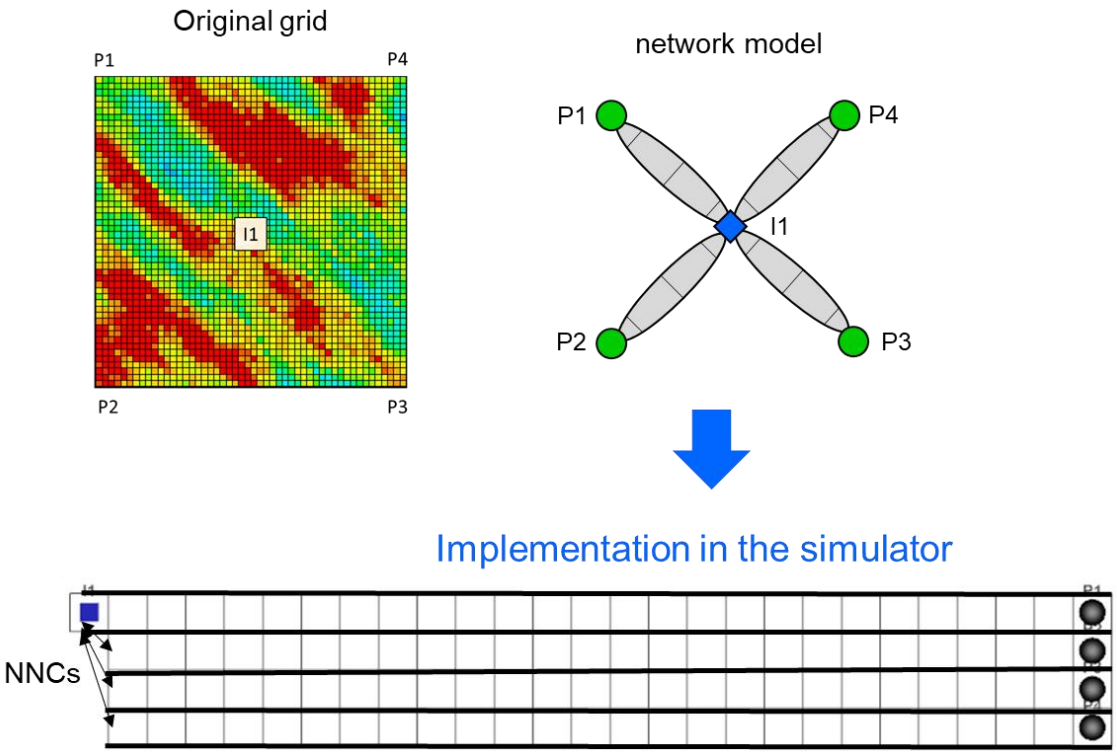


Figure 3.5: The network and 2D simulation grid for the synthetic case

Four networks connect each producer injector pair. In the numerical simulation, the grid system looks like the picture at the bottom. The simulation was conducted using the liquid rate constraint for producers and the water injection rate constraint for the injector. In this problem, the oil production rate and water production rate were integrated, and we

calibrated the permeability multipliers and pore volume multipliers. For the initial condition of permeability and pore volume, we assigned a uniform value, roughly the average of properties in the system. The ranges of parameters were determined from a few numerical experiments so that the ensemble results in a realistic range of production rate. The pore volume and permeability were computed as following equations.

$$k = k_0 \exp\left(mult_{perm}\right) \tag{3.2}$$

$$PV = PV_0 \exp\left(mult_{PoreV}\right) \tag{3.3}$$

Where,

k : permeability
k_0 : base permeability
$mult_{perm}$: permeability multiplier
PV : pore volume
PV_0 : base pore volume
$mult_{PoreV}$: pore volume multiplier

In **Figure 3.6**, the runtime comparison between the full geologic model and flow network model is provided. It shows that the network model reduced the simulation cost by around 40 percent. The CPU time reduction might not be significant for this case because the 2D synthetic was a simple case, and it is not very expensive. However, for large reservoirs with high heterogeneity, the number of grid blocks increases, and the CPU time reduction becomes more significant.

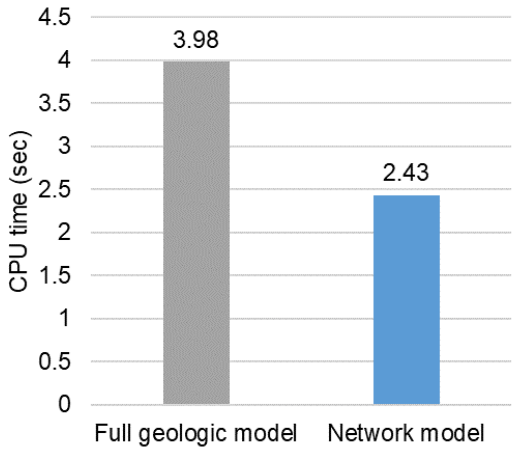


Figure 3.6: Run time comparison between full reservoir model and 1D network model for 2D synthetic case

3.2.2 History Matching Results

We used ESMDA for the history matching, and the matching results are provided in **Figure 3.7** (oil production rates). The red dots are the observed data, light blue lines represent the initial ensemble, and dark blue lines represent the updated ensemble. For all producers, ESMDA provides a suitable match, and we obtained a reliable 1D network model.

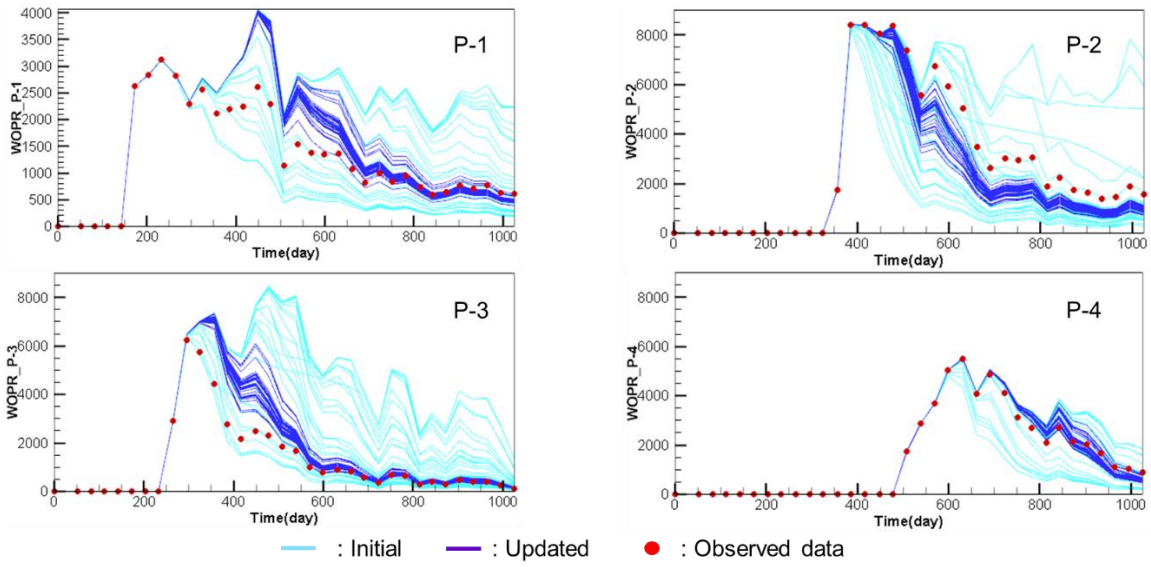


Figure 3.7: History matching result, oil production rate (2D synthetic case)

In order to confirm the reliability of this model, we tested the prediction performance for the oil production rate, and the results are given in **Figure 3.8**. The dark blue lines are representing model estimation, and red dots are observed data. The vertical dash line divides it into two timeframes. The production history until the dashed line was integrated into the model and used for calibration. The rest of it was used for validating the prediction performance. For all producers, reasonable matching quality was obtained, and the reliability of prediction performance was confirmed from this experiment.

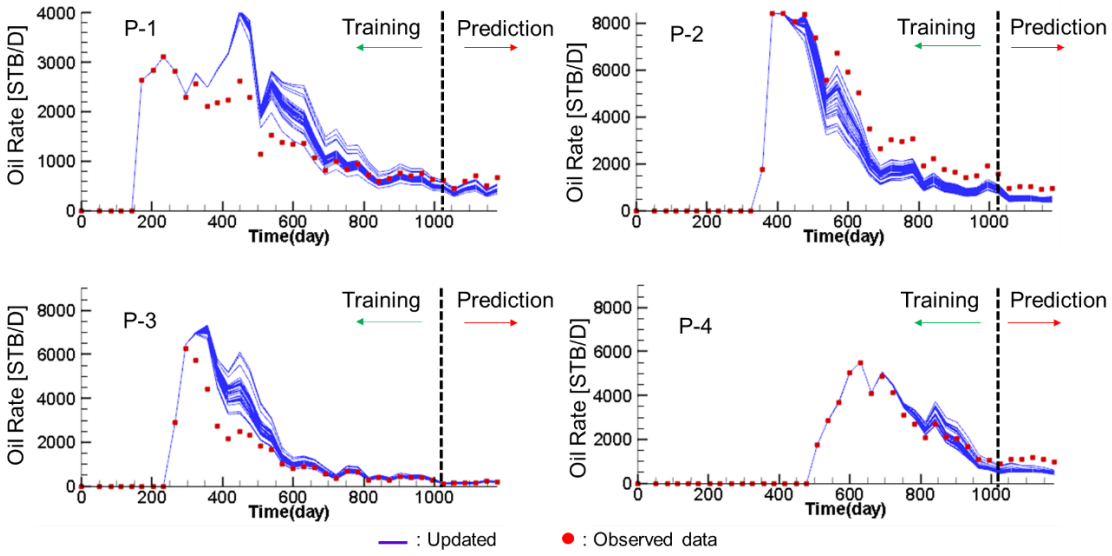


Figure 3.8: Prediction result for oil production rate (2D synthetic case)

Next, we analyzed the reservoir connectivity using finite-volume-based flow diagnostics. Using the calibrated network model, we ran a single timestep simulation with water injection and uniform bottom-hole pressure constraint for all producers. Then, compute well allocation factors from streamline time of flight calculation for each injector producer pair. We implemented the same procedure but using the original full reservoir model, and those two results are compared in **Figure 3.9**. The right plot is from the network model, and the middle one is from the original full reservoir model. Although there is a slight deviation from the original model, the network model still captures the major reservoir connectivity trend.

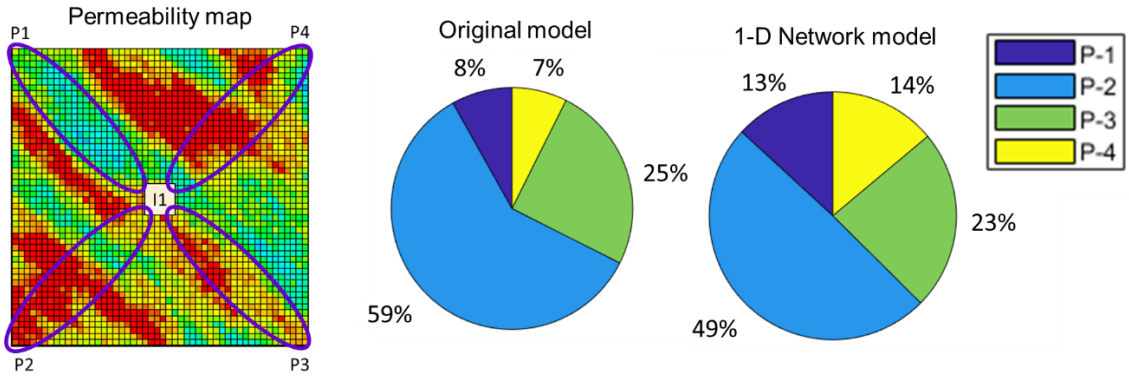


Figure 3.9: Reservoir connectivity for 2D synthetic case

3.3 Application of a Mature Oil Field with CO_2 WAG Injection

We applied the 1D network model to the mature oil field with CO_2 WAG injection. The calibrated model was used for flow diagnostics and streamline-based injection rate allocation optimization.

3.3.1 Model Description

The full field model has 2.5 million active cells, and the fluvial reservoir has significant permeability contrasts at varying length scales, ranging from 0.5 to 35,000mD. It is categorized as a highly heterogeneous reservoir. As mentioned, this field has CO2 injection, and we need to use compositional simulation to capture the multiphase multicomponent fluid flow accurately. **Figure 3.10** provides the permeability map of the full reservoir field. The field has more than 100 producers and injectors, totally more than 200 wells are completed. The production and injection histories are recorded after the WAG injection started. Because of the number of grid cells and the model complexity, the

full reservoir model is very expensive, and it takes more than 20 days for a single simulation. Then, history matching and optimization are not feasible.

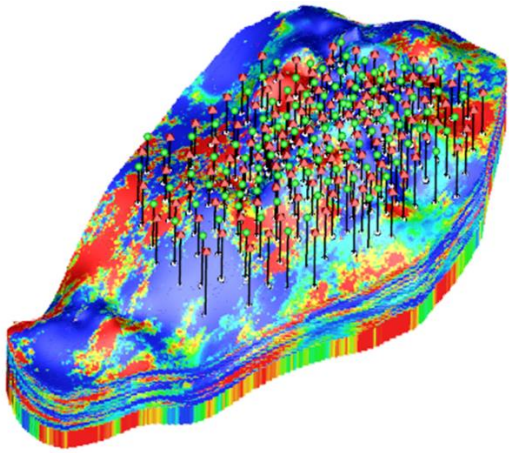


Figure 3.10: Permeability map of the mature oil field involving CO_2 WAG injection

3.3.2 Flow Network Grid Generation and reservoir model initialization

First, we need to generate the flow network grid system and assign the initial conditions of the simulation. Because we have a prior model for this reservoir field, we decided to use streamlines and to include some prior information into the model. The steps for the network grids generation and assignment of initial conditions are as follows.

1. Run a single-step full-field simulation with uniform injection rates for all injectors and uniform production rates for all producers using the prior model.

2. Trace streamlines for the single-step simulation and determines partition and well pair maps from the tracer concentration information.

3. Determine the 1D network grid system based on the well pairs from the streamline tracing.

4. Compute the average of geologic properties and grid solutions within each well pair partition.

5. Within each well pair, assign the averaged value of geologic properties as initial model parameters and averaged solutions as initial conditions of the simulation.

In this way, the 1D network model can effectively include the prior information, and it is helpful to obtain suitable history matching performance. **Figure 3.11** shows streamline tracing maps with the time of flight, and **Figure 3.12** provides partition maps of the full-field model from tracer concentration. For the streamline tracing, the "Destiny" (MCERI in-house software) was used. **Figure 3.13** is the flow network grid system of the entire field.

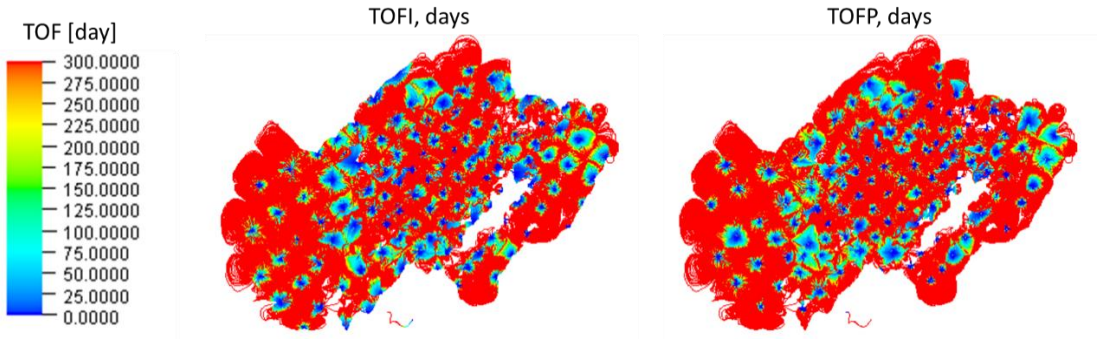


Figure 3.11: Streamline tracing maps (time of flight from injector and producer)

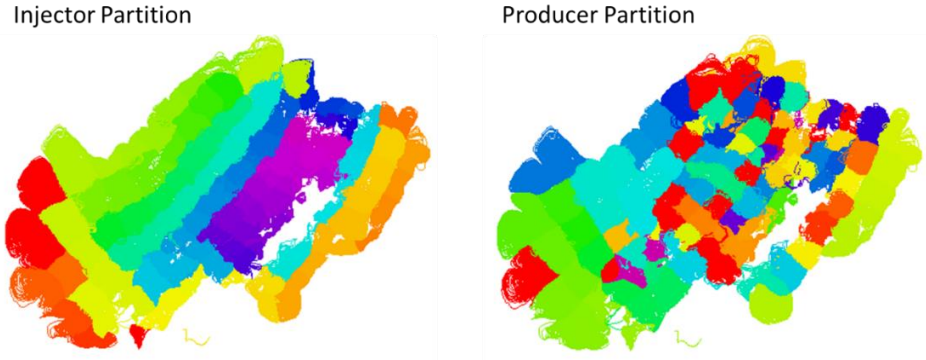


Figure 3.12: Partition map of the mature oil field with CO_2 WAG injection

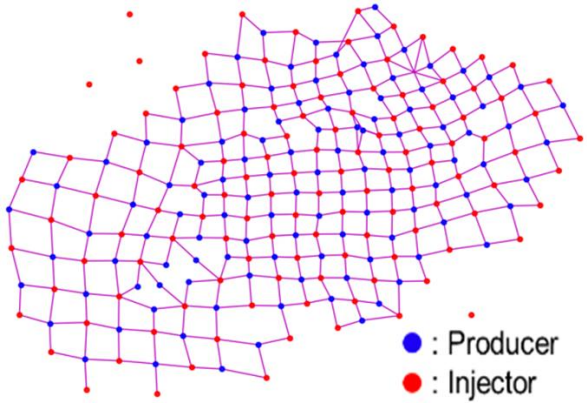


Figure 3.13: Network grid system of the mature oil field with CO_2 WAG injection

The dimension of the flow network grid system is provided in **Table 3.1**. We set four connections for each well pair for capturing a complex multiphase multicomponent flow system.

Table 3.1: Flow Network dimension

Number of connections	424 x 4
Grid each connection	5
Number of grids	8480

3.3.3 History Matching Formulation

In the reservoir simulation, we had liquid production rates constraint (Oil + Water) for producers, and we had water and gas (CO2) injection rates constraint for injectors. We calibrated pore volume multipliers, the permeability multipliers, and the relative permeability model parameters. The pore volume and permeability were computed using the multipliers as follows.

$$k = k_0 \exp\left(mult_{perm}\right) \tag{3.4}$$

$$PV = PV_0 \exp\left(mult_{PoreV}\right) \tag{3.5}$$

Where,

k : permeability
k_0 : base permeability
$mult_{perm}$: permeability multiplier
PV : pore volume
PV_0 : base pore volume

$mult_{PoreV}$: pore volume multiplier

The relative permeability model was based on the Brooks-Corey correlation (Brooks et al. 1964) and was augmented by the Stone correlation (Stone 1973) to compute three-phase relative permeability. The equations of this relative permeability model are as follows.

$$k_{rw} = k_{rwE} S_w^{n_w} \tag{3.6}$$

$$k_{row} = k_{rowE} \left(1 - S_w\right)^{n_{ow}} \tag{3.7}$$

$$k_{rg} = k_{rgE} S_g^{n_g} \tag{3.8}$$

$$k_{rog} = k_{rogE} \left(1 - S_g\right)^{n_{og}} \tag{3.9}$$

$$S_w = \frac{s_w - s_{wc}}{1 - s_{orw} - s_{wc}} \tag{3.10}$$

$$S_g = \frac{s_g}{1 - s_{org} - s_{wc}} \tag{3.11}$$

The three-phase oil relative permeability is given by Stone correlation.

$$k_{ro} = k_{rowE} \left\{ \left(\frac{k_{row}}{k_{rowE}} + k_{rw} \right) \left(\frac{k_{rog}}{k_{rowE}} + k_{rg} \right) - k_{rw} - k_{rg} \right\} \tag{3.12}$$

Where,

n_w : Water relative permeability exponent

n_{wo} : Oil-water relative permeability exponent

n_g : Gas relative permeability exponent

n_{og} : Oil-gas relative permeability exponent

k_{rwE} : Water relative permeability endpoint

k_{rgE} : Gas relative permeability endpoint

We determined the ranges of the permeability model parameters based on previous work in this field. **Table 3.2** summarizes the list of the history matching parameters. The ranges of permeability and pore volume were determined based on a few numerical experiments to obtain reasonable ranges of production histories. We have multiphase production rate histories of oil, water, and gas for about two years as observed data.

Table 3.2: List of history matching parameters and their ranges

Parameter		Parameterization Level	Low	Base	High
Permeability Multipliers		Grid level	-1.0	0.0	1.0
Pore Volume Multipliers		Grid level	-2	-0.5	1.0
Relative Permeability Model Parameters	n_w	Global	1.5	3.0	5.0
	n_{wo}		1.2	2.0	5.0
	n_g		1.5	2.0	5.0
	n_{og}		1.2	2.0	4.0
	k_{rwE}		0.2	0.5	1.0
	k_{rgE}		0.5	0.8	1.0

3.3.4 History Matching Result

Before we provide the history matching result, we would like to show the CPU time comparison between the full 3D reservoir model and the 1D network model. **Figure 3.14** presents the comparison of the number of grid blocks and CPU time. A full 3D model takes more than 20 days for a single simulation. On the other hand, the 1D network model takes just 45 minutes. Therefore, we can implement history matching and optimization in a realistic time frame.

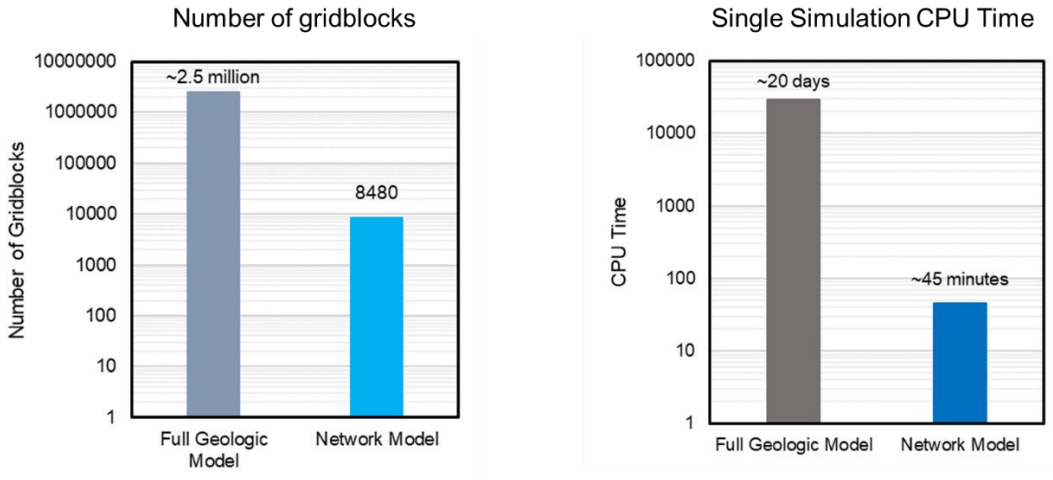


Figure 3.14: CPU time comparison between 3D full reservoir model and 1D network model

By using ESMDA for history matching, we integrated multiphase production rate histories and calibrated model parameters. **Figure 3.15** shows the matching result of the oil production rate, and **Figure 3.16** shows the matching performance of the gas

production rate. We can see that the history matching was properly implemented and obtained reasonable agreements with observed data. The 1D network model shows some large spikes for some wells, even if the observed data does not show that trend. Because the flow network system actively simplifies the reservoir model using 1D grid connections. Although this is not a very accurate model, it reduces the CPU time significantly so that the history matching and optimization can be done in a realistic timeframe.

One of the other possible reasons for the deviation is that there are many workover operations in this field. There is severe vertical permeability anisotropy, and the different layers have minimal communication with each other. Therefore, once the completion zone is changed, it generates an entirely new reservoir connection between producers and injectors. It is not easy to capture this operation because the network grid is determined before the history matching implementation. This problem may be resolved by changing the well pair connection during the history matching. Also, there are lots of missing well histories near the field edges, and it is one of the reasons for the disagreement. However, for the overall result, we obtained reasonable matching quality, and this calibrated 1D network model can be utilized for some diagnostics and optimization of reservoir management.

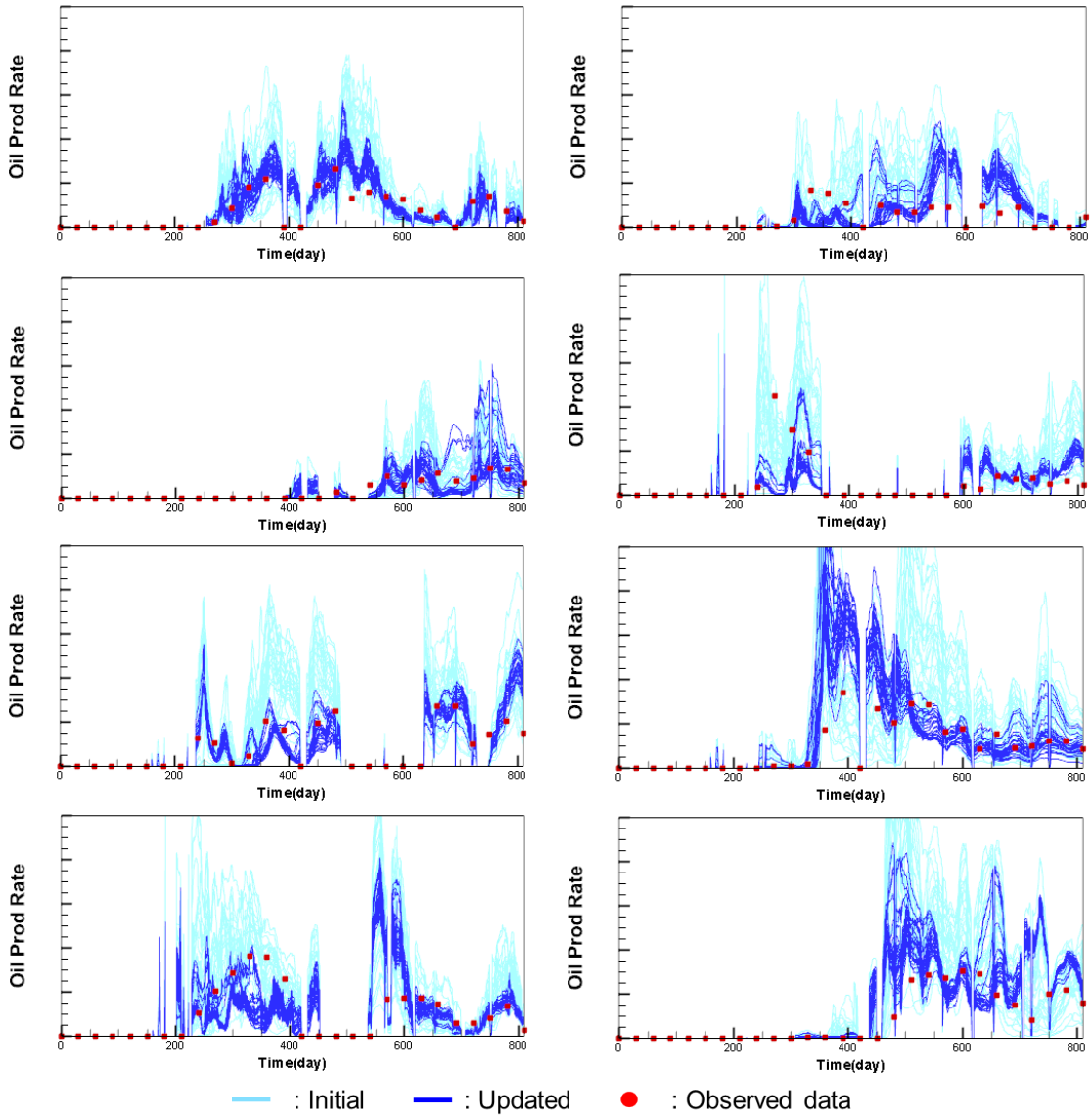


Figure 3.15: History matching result, the oil production rate for selected wells (real field application)

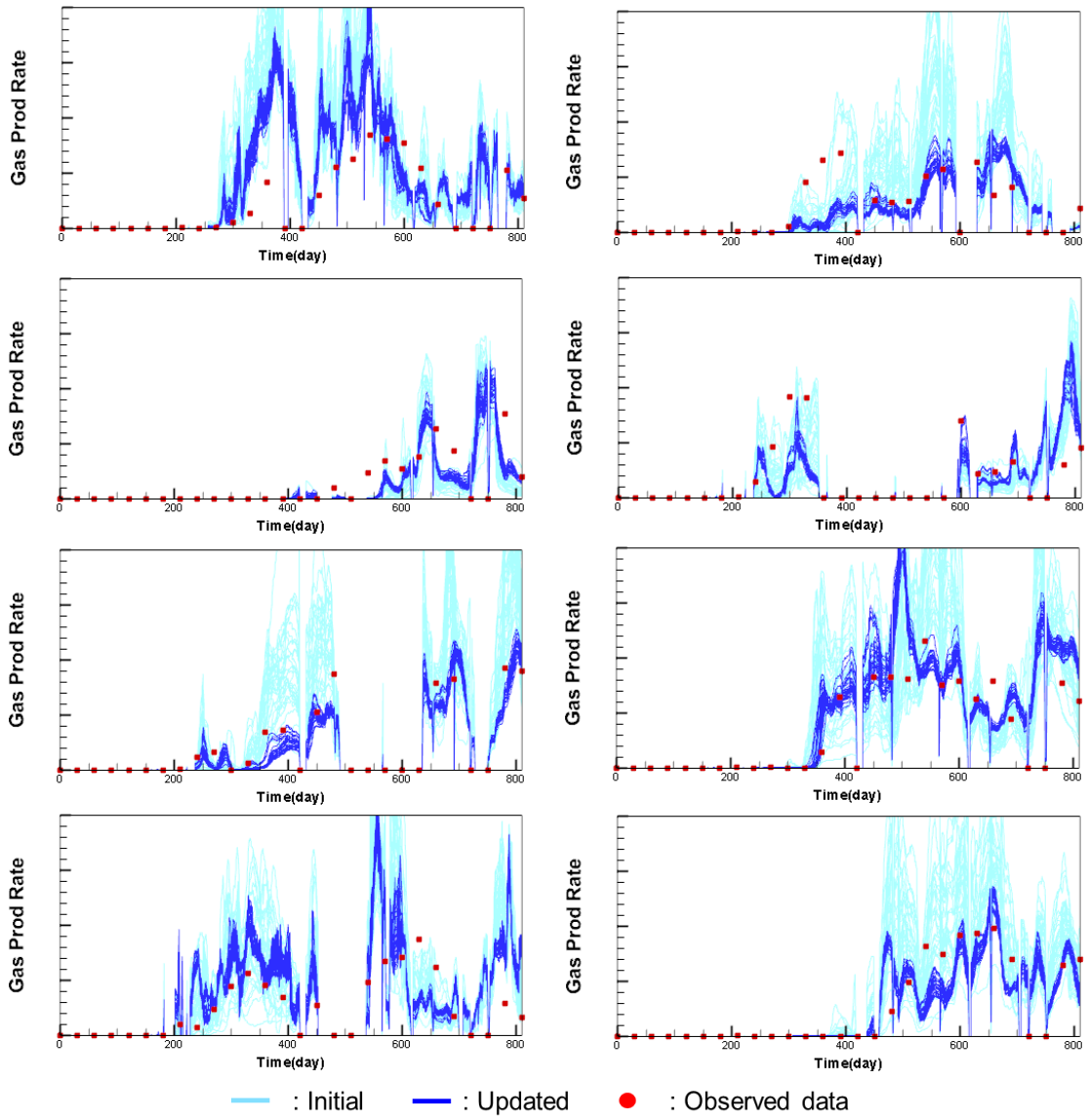


Figure 3.16: History matching result, the gas production rate for selected wells (real field application)

Using the calibrated network model, we predicted the oil and gas production rate. **Figure 3.17** and **Figure 3.18** show the predicted oil production rate and gas production rate of selected wells. The vertical dash line divides the production history. Until this dashed line, the production histories were utilized for model calibration. After

this dashed line, the data was used to assess the prediction performance, and it is about three months.

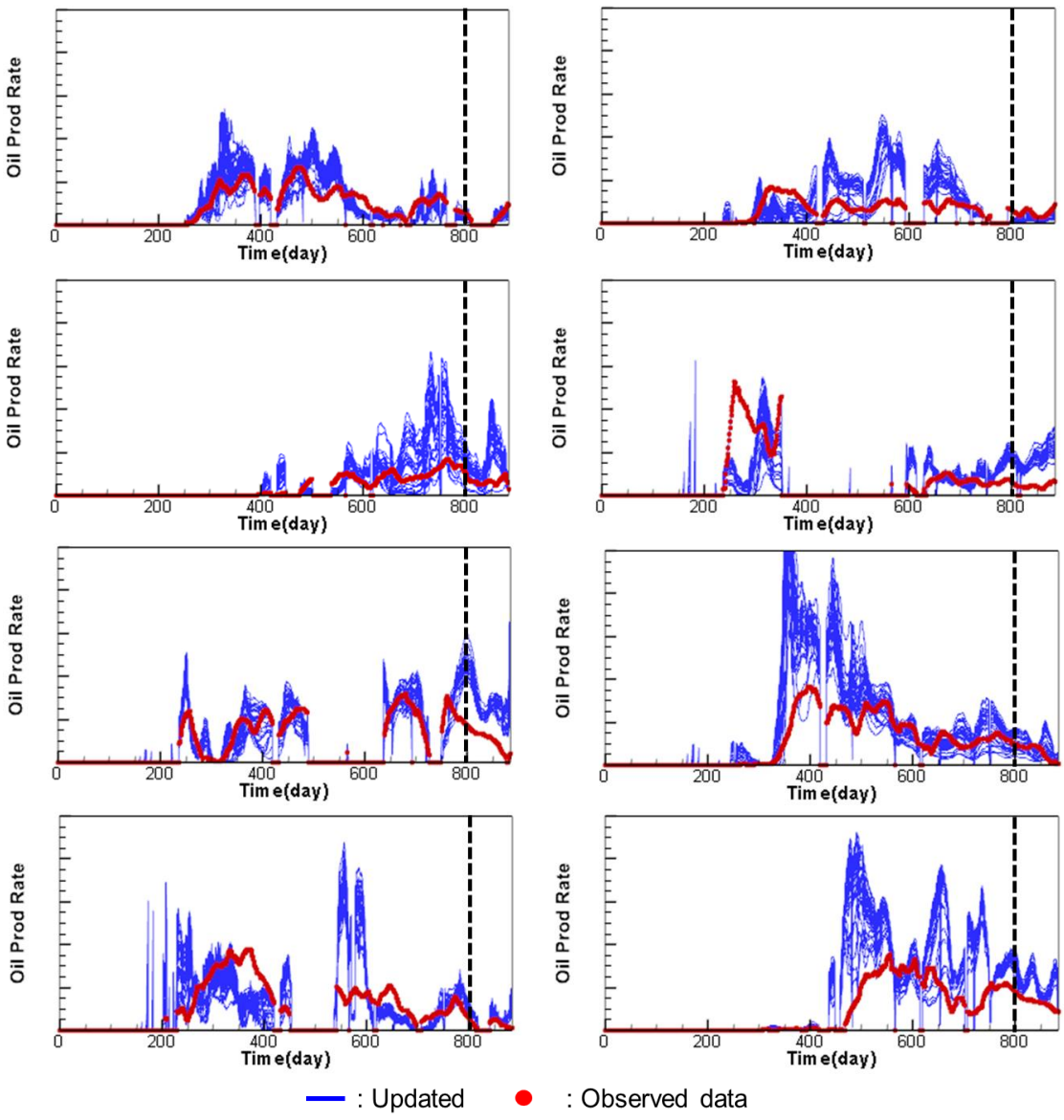


Figure 3.17: Prediction of oil production rate (real field application)

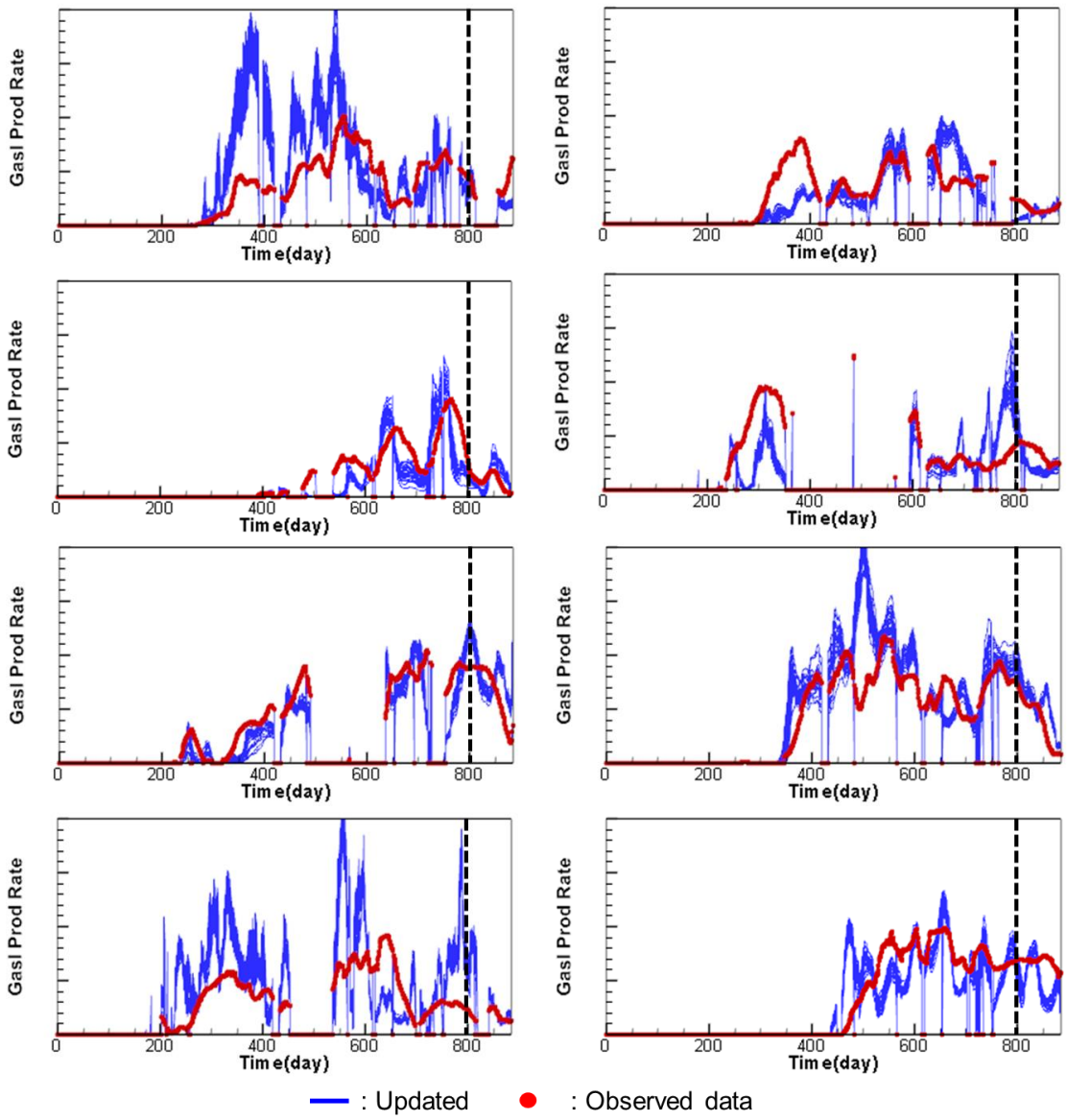


Figure 3.18: Prediction of gas production rate (real field application)

Next, we plotted the saturation transition during the 1D network model simulation. **Figure 3.19** shows the oil saturation distribution change with time. We can see that the oil is depleted mainly from the center of the reservoir. From this map, we can find connections that have a relatively large amount of remaining oil.

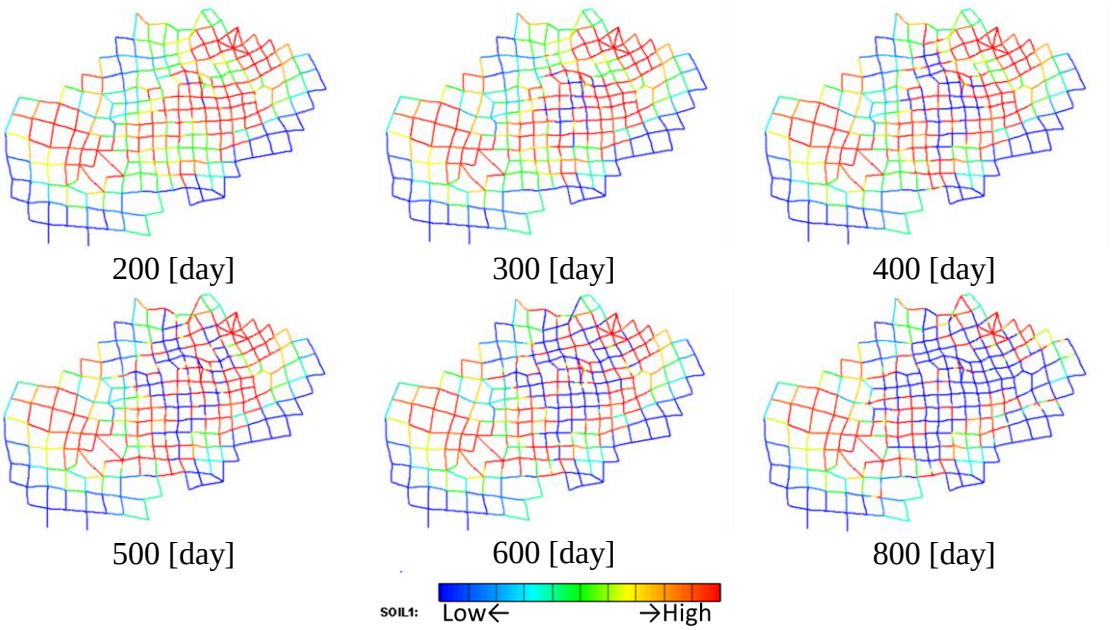


Figure 3.19: Oil saturation distribution for different time steps

Finally, we plotted the gas saturation distribution, which is provided in **Figure 3.20**. At 200 [day] of history, there is almost no storage gas in the field. The gas saturation gradually increases mainly around the center because the CO2 injection started from the center region. From this map, we can see which direction or connection the gas injection goes, and this information is useful for optimizing gas injection allocation.

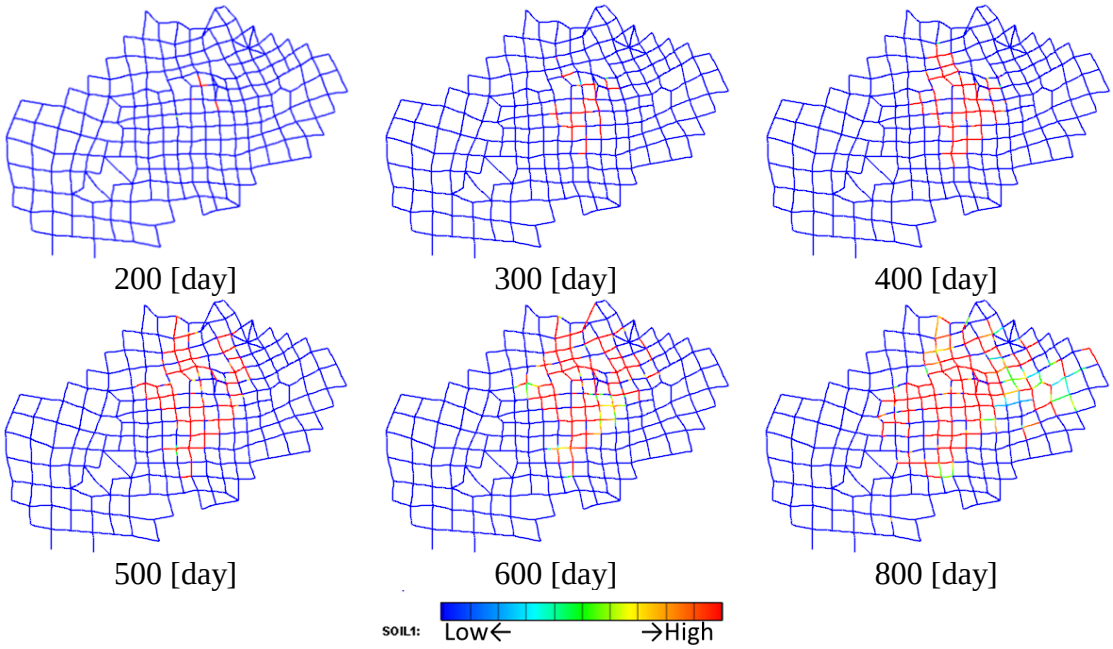
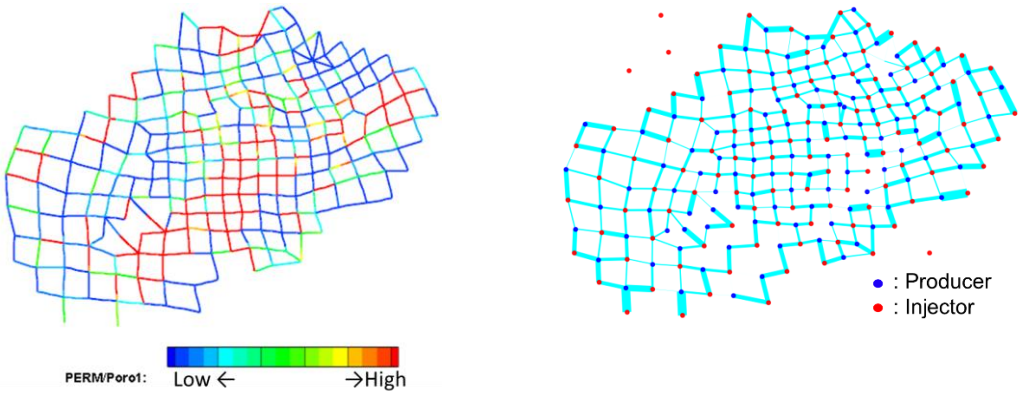


Figure 3.20: Gas saturation distribution for each time step

3.3.5 Reservoir Connectivity Identification

After the history matching, we estimated the reservoir connectivity using the calibrated 1D network model. **Figure 3.21** shows two different reservoir connectivity maps. Picture (a) plots the value of the permeability divided by the porosity of each grid cell. The large values (drawn by red) have large reservoir connectivity, and the small values (drawn by blue) have small connectivity. Picture (b) is a reservoir connectivity map from the well allocation factor. We compute the well allocation factor for each injector using the finite-volume-based flow diagnostics. In picture (b), the width is corresponding

to the value of well allocation factor of each injector. Then, from this plot, we can see how much injected fluid goes in which direction in the reservoir.



(a) (b)

Figure 3.21: Reservoir connectivity map using a history matched model. (a): permeability/porosity map. (b): well allocation factor map

3.3.6 Streamline-based rate allocation optimization

Next, we implemented the streamline-based rate allocation optimization using the history matched 1D network model. We followed the algorithm presented by Tanaka et al. (2017) and implemented using "Destiny" (MCERI in-house software). In this section, we briefly explain the algorithm and workflow of the rate allocation optimization.

The first step is tracing the streamlines and compute the value of the hydrocarbon volume along each streamline. In our application, we trace the streamlines for the history matched 1D network model.

$$\Lambda_\ell = q_\ell \sum_{node} \frac{\Delta\tau}{\rho_e} \sum_{\alpha=oil,gas} S_\alpha b_\alpha R_\alpha \tag{3.13}$$

Where,

q_ℓ : flow rate assigned to single streamline

ρ_e : effective density

$\Delta\tau$: time of flight within the node

R_α : the price of fluid per unit volume

b_α : inverse of formation volume factor

S_α : phase saturation

This equation represents the maximum possible profit from the single streamline. Then, predictive NPV along streamline can be estimated as follows:

$$r_\ell = q_\ell \sum_{node} \frac{\Delta\tau}{\rho_e} \sum_{\alpha=oil,wat,gas} \left(S_\alpha b_\alpha R_\alpha\right)\left(1+d\right)^{-\tau_{ey} f_\alpha^{-1}} + q_\ell b_{\alpha i} R_{\alpha i} \sum_{t_i}^{t_e} \Delta t \left(1+d\right)^{-t_{ey}} \tag{3.14}$$

Where,

f_α : fractional flow

t_i : arbitrary estimation start time

t_e : end of the reservoir lifetime

τ_{ey} : time of flight from producer

t_{ey} : remaining reservoir life ($=\left(t_e - t_i\right)/365$)

d : discount rate

$R_{\alpha i}$: the price of injection fluid per unit volume

$b_{\alpha i}$: inverse of formation volume factor of injection fluid

From the expected NPV value, we can estimate the well pair efficiency as follows:

$$e_{ip,NPV} = \frac{r_{ip}}{\Lambda_{ip}} = \sum_{N_{\ell,ip}} \frac{r_{\ell}}{\Lambda_{\ell}} \tag{3.15}$$

By using the well pair efficiency, the connection flow rates are updated as follows:

$$q_{ip}^{ite+1} = q_{ip}^{ite}\left(1-\eta+2\eta e_{ip}\right) \tag{3.16}$$

Where,

q_{ip}: connection flow rate
ite: iteration index
η: fraction of the allowable rate changes per iteration

We iterate these steps until it reaches the convergence criteria or the maximum number of iterations. For more details, please refer to Appendix A and Tanaka et al. (2017).

In the application for the mature oil reservoir, we use the history matched 1D network model. Hence, we trace the streamlines for the calibrated network grid system. The field injection rate and WAG injection cycle were fixed. The detailed description is provided in **Table 3.3**.

Table 3.3: Formulation of rate allocation optimization

Optimization Period	150 [day]
Field Liquid Production Rate	Fixed
Field Gas Injection Rate	Fixed
Field Water Injection Rate	Fixed
WAG Injection Cycle	Water = 10 [day] Gas = 20 [day]

The injection rate and production rate allocation were optimized, and we compared the oil recovery factor between the optimized case and the base case. The base case had uniform rates for all producers and injectors. **Figure 3.22** provides the well rates of the base case.

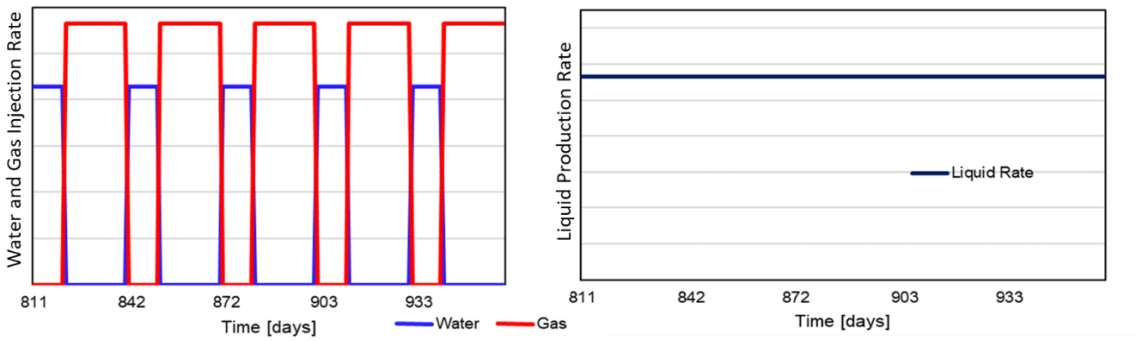


Figure 3.22: The base case for streamline-based rate allocation optimization

The optimization result is provided in **Figure 3.23**. The optimized case gets better sweep efficiency and provides roughly 2.0% larger oil recovery factors. Therefore, it was verified that the 1D network model workflow could be used for rate allocation optimization for a large real field with CO_2 WAG injection. In this application, the WAG injection cycle was fixed, and we only optimized the rate allocation. However, we can further extend to optimize the WAG injection cycle time for each well by using population-based methods like genetic algorithm.

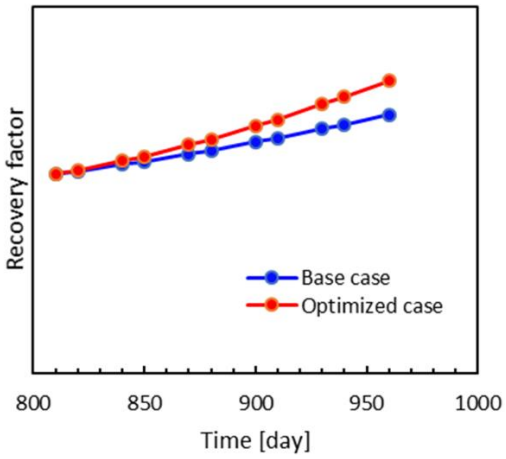


Figure 3.23: Oil recovery factor of the base case and optimized case

CHAPTER IV

SUMMARY AND PATH FORWARD

4.1 Application of Recurrent Neural Networks

In Chapter II, we presented the machine learning data-driven reservoir model using LSTM. The LSTM was fitted to the production rates based on the surrounding injector's rate histories. The trained LSTM networks were utilized for predicting the future production rate. Then, the LSTM variable importance method was tested for identifying the reservoir connectivity between producers and injectors. We applied this workflow to the 2D synthetic case, 3D large field-scale reservoir, and a mature oil field involving large-scale CO_2 WAG injection.

For the 2D synthetic case, the LSTM data-driven reservoir modeling worked well. We predicted the liquid production rate from the surrounding injector's water rates, and the overall prediction performance was reasonable. The reservoir connectivity identification from LSTM variable importance was also reasonably matched with the streamline allocation factor.

For the 3D large field-scale reservoir case, the LSTM model was fitted to the liquid production rate based on the surrounding injector's bottom-hole pressure and water injection rates. The prediction performance was reasonable for most of the producers. The reservoir connectivity was estimated by LSTM variable importance, and there is some

difference in the reservoir connectivity results between LSTM and streamline allocation factor.

For the mature oil field application, we fitted the LSTM networks to the gas production rates based on the surrounding injector's gas rates, water rates, cumulative gas rates, and producer's tubing head pressure. We obtained reasonable prediction results for some wells, but it still needs to be improved.

4.1.1 Consideration from the Recurrent Neural Networks Application

LSTM Performance for Production Rate Forecasting

We obtained reasonable prediction performance for synthetic cases, but the result of the field application needs to be improved. There are some possible reasons for the disagreement in the prediction result. First, the mature oil field is a highly heterogeneous reservoir, and the relationships between producers and injectors may not be clear. Second, we have many workover operations during history. There is severe vertical permeability anisotropy in this field, and once the perforation is re-completed in the different vertical zones, it possibly creates completely new reservoir connections with other wells. Third, the raw data includes much noise, and it might affect the prediction performance.

Reservoir Connectivity Identification from LSTM Variable Importance

It is a challenging task to identify the reservoir connectivity using the LSTM variable importance. Although it worked for the 2D synthetic waterflooding case, we did not obtain suitable results for the 3D waterflooding case. There are several possible reasons for this disagreement. First, there were similar well rate profiles for many producers in the field. Because the data-driven approach only looks at the well rate histories for identifying the reservoir connectivity, if the well rate profiles are very similar, it provides a similar value of reservoir connectivity regardless of the other information like the well location and geologic properties. Second, the producer and injector might not be highly connected in this reservoir. The 2D synthetic reservoir had highly conductive channel formation, and the injector's rate fluctuations could be identified in the target producer's history as signals. Then, it is relatively easier to capture the reservoir connectivity for highly heterogeneous channel reservoirs. However, for low permeability reservoirs, the signals from the surrounding injectors cannot be captured because the production rate profiles are smoothed due to the low permeability formation. Probably, the gas injection case is slightly easier to obtain the reservoir connectivity because the injected gas tends to form channel-like flow paths due to its low viscosity. Finally, we also found that the reservoir connectivity result was very sensitive to the length of the search radius. Then, we need some proper way to determine the search radius or other constraints before the neural network model calibration.

4.1.2 Path forward for Machine Learning Study

For obtaining suitable prediction performance for the real field application involving large-scale CO_2 WAG injection, we need to investigate a method to account for the workover operation. The simplest way is to use the well histories only after the workover operation. However, in this case, the amount of data is significantly reduced, and the prediction performance may get worse. The other possible option is transfer learning. The basic idea of transfer learning is to reuse the pre-trained neural network model for similar problems. We slightly modify the pre-trained neural network model by using the newly obtained dataset (Chollet 2018). In our application, we first train the model using the well histories before the workover. Then, we modify the trained model using the well histories after the workover operation. In this way, we can effectively use the entire dataset. This approach may work if the workover is not very often and if all the re-completion operations are recorded. In the field application, the workovers are very often, and they do not record all of it. Then, probably, we need to seek other approaches.

Next, we recommend using a more sophisticated version of neural network models. Oliva et al. (2017) presented a novel neural network architecture suitable for time series problems called Statistical Recurrent Unit (SRU). It has a simple and un-gated architecture and contains a comparable number of parameters to LSTM. They demonstrated that SRU overperformed LSTM and GRU for many applications. Li et al. (2019) developed an interesting neural network architecture called long and short-term Time-series networks (LSTNet). This architecture is a combination of RNN and CNN. The study showed that

LSTNet provided more stable prediction performance than other ordinary neural network models, LSTM and GRU.

The real field data includes noise in the well histories, and it is better to remove it as data preprocessing. However, if we use a smoothing algorithm, it possibly removes small signals, which indicates the relationship between producers and injectors. It is a delicate problem, but it would improve the prediction performance if we can denoise the data.

Finally, we need to investigate some strategies to determine the optimal search radius for the application of reservoir connectivity identification using variable importance.

4.2 Application of a Physics-Based Data-Driven Model

In Chapter III, we presented the application of 1D network model for 2D synthetic reservoir case and a real mature oil field involving large-scale CO_2 WAG injection. The flow network system was generated based on the well location, and the network model was calibrated using ESMDA. The calibrated 1D network models were utilized to identify the reservoir connectivity and for streamline-based rate allocation optimization.

For the 2D synthetic reservoir case, we obtained suitable history matching results and prediction performance. The well allocation factors were compared between the calibrated network model and the true reservoir model, and it showed great agreement.

For the application of the mature oil field involving CO_2 WAG injection, the 1D network model was constructed for the entire field. The simulation CPU time for the 1D network model was three orders of magnitude faster than the original 3D reservoir model. The history matching performance was reasonable, and we obtained good prediction results for many wells. The reservoir connectivity was estimated by computing the well allocation factor. Finally, we implemented the streamline-based rate allocation optimization for the entire field, and it provided improved oil recovery factors from the base case.

4.2.1 Consideration of the Physics-Based Data-Driven Model

The 1D network model provided promising history matching performance. For some producers, the prediction result did not match with the actual data. One of the possible reasons for the disagreement is that there are many workovers in the field, and it possibly generates new reservoir connections through different layers. Because the flow network grid system is constructed before the history matching, it is not easy to capture the new reservoir connection fluid flow during the history matching. This problem may be resolved by changing and calibrating the network connections during the history matching.

The entire workflow of history matching and the rate allocation optimization is very useful for many other fields involving large-scale EOR. It significantly reduces the simulation time and provides a robust model calibration and rate allocation optimization method.

4.2.2 Path Forward for the Physics-Based Data-Driven Model

The history matching quality can be improved by considering the workovers during the histories by calibrating the network connections. In the ESMDA procedure, it is required to compute the inverse of the data covariance matrix. If the amount of data is very large, the inverse calculation can be very expensive and requires huge memory. Then, it is recommended to apply some data dimensionality reduction methods like spectral decomposition. It will enhance the computational speed and the history matching quality by incorporating a large amount of data.

In the rate allocation optimization, the WAG injection cycle time was fixed for the entire field. If we can extend to optimize the injection cycle time, it would further improve the oil recovery from the reservoir.

REFERENCE

Albertoni, Alejandro, and Larry W. Lake. 2003. "Inferring Interwell Connectivity Only From Well-Rate Fluctuations in Waterfloods." *SPE Reservoir Evaluation & Engineering* 6 (01):6-16.

Alhuthali, Ahmed, Adedayo Oyerinde, and Akhil Datta-Gupta. 2007. "Optimal Waterflood Management Using Rate Control." *SPE Reservoir Evaluation & Engineering* 10 (05):539-551.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1):5-32.

Chen, B. 2021. "7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2." accessed April 9. **https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6**.

Cheng, H., V. Vyatkin, E. Osipov, P. Zeng, and H. Yu. 2020. "LSTM Based EFAST Global Sensitivity Analysis for Interwell Connectivity Evaluation Using Injection and Production Fluctuation Data." *IEEE Access* 8:67289-67299.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Y. Bengio. 2014. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation."

Chollet, FranCois. 2018. Deep Learning with Python. Edited by Toni Arritola, Jerry Gaines, Aleksandar Dragosavljevic and Tiffany Taylor. Shelter Island, NY: Manning Publiccations Co.

Chou, Jui-Sheng, and Ngoc-Tri Ngo. 2016. "Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns." *Applied Energy* 177:751-770.

Colah. 2015. "Understanding LSTM Networks." accessed April 10. **https://colah.github.io/posts/2015-08-Understanding-LSTMs/**.

Du, S., R. Wang, C. Wei, Y. Wang, Y. Zhou, J. Wang, and H. Song. 2020. "The Connectivity Evaluation Among Wells in Reservoir Utilizing Machine Learning Methods." *IEEE Access* 8:47209-47219.

Emerick, Alexandre A., and Albert C. Reynolds. 2013. "Ensemble smoother with multiple data assimilation." *Comput. Geosci.* 55:3–15.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. "Model class reliance: Variable importance measures for any machine learning model class, from the" rashomon" perspective." *arXiv preprint arXiv:1801.01489* 68.

Fujita, Yusuke, Akhil Datta-Gupta, and Michael J. King. 2016. "A Comprehensive Reservoir Simulator for Unconventional Reservoirs That Is Based on the Fast Marching Method and Diffusive Time of Flight." *SPE Journal* 21 (06):2276-2288.

Fumo, David. 2017. "A Gentle Introduction To Neural Networks Series — Part 1." accessed April 9. **https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc**.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Comput.* 9 (8):1735–1780.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An introduction to statistical learning : with applications in R, *Springer texts in statistics*. New York: Springer.

Kareepadath Sajeev, Shyam. 2020. Application of Deep Learning for Understanding Dynamic Well Connectivity. Thesis.

Kingma, Diederik P., and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." *CoRR* abs/1412.6980.

Laochamroonvorapongse, R., C. S. Kabir, and Larry W. Lake. 2014. "Performance assessment of miscible and immiscible water-alternating gas floods with simple tools." *Journal of Petroleum Science and Engineering* 122:18-30.

Li, Yuanjun, Ruixiao Sun, and Roland Horne. 2019. "Deep Learning for Well Data History Analysis." *SPE Annual Technical Conference and Exhibition*.

Lutidze, G. 2018. Stellnet - Physics-Based Data-Driven General Model For Closed-Loop Reservoir Management: University of Tulsa.

Madurai Elavarasan, Rajvikram, G. M. Shafiullah, Kannadasan Raju, Vijay Mudgal, M. T. Arif, Taskin Jamal, Senthilkumar Subramanian, V. S. Sriraja Balaguru, K. S. Reddy, and Umashankar Subramaniam. 2020. "COVID-19: Impact analysis and recommendations for power sector operation." *Applied Energy* 279:115739.

Molnar, Christoph. 2021. "Interpretable Machine Learning." Last Modified June 14, accessed April 12. **https://christophm.github.io/interpretable-ml-book/**.

Møyner, Olav, Stein Krogstad, and Knut-Andreas Lie. 2014. "The Application of Flow Diagnostics for Reservoir Management." *SPE Journal* 20 (02):306-323.

Murray, Rodney L., Reese S. Hopkins, and Douglas K. Valentine. 2020. "Network Optimization Models at Greater Kuparuk Area Using Neural Networks and Genetic Algorithms." *SPE Annual Technical Conference and Exhibition*.

Nwachukwu, Azor, Hoonyoung Jeong, Michael Pyrcz, and Larry W. Lake. 2018. "Fast evaluation of well placements in heterogeneous reservoir models using machine learning." *Journal of Petroleum Science and Engineering* 163:463-475.

Oliva, Junier B., Barnabás Póczos, and Jeff Schneider. 2017. "The Statistical Recurrent Unit." *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research.

Osako, Ichiro, and Akhil Datta-Gupta. 2007. "A Compositional Streamline Formulation With Compressibility Effects." *SPE Reservoir Simulation Symposium*.

Perlato, Andrea. 2020. "Bias Variance Trade-Off." accessed April 10. **https://www.andreaperlato.com/theorypost/bias-variance-trade-off/**.

Ren, Guotong, Jincong He, Zhenzhen Wang, Rami M. Younis, and Xian-Huan Wen. 2019. "Implementation of Physics-Based Data-Driven Models With a Commercial Simulator." *SPE Reservoir Simulation Conference*.

Sayarpour, M., E. Zuluaga, C. S. Kabir, and Larry W. Lake. 2009. "The use of capacitance–resistance models for rapid estimation of waterflood performance and optimization." *Journal of Petroleum Science and Engineering* 69 (3):227-238.

Shahvali, M.., B.. Mallison, K.. Wei, and H.. Gross. 2012. "An Alternative to Streamlines for Flow Diagnostics on Structured and Unstructured Grids." *SPE Journal* 17 (03):768-778.

Sharma, Sagar. 2017. "Activation functions in neural networks." Last Modified 2017, accessed April 12. **https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6**.

Shook, G. Michael, and Kameron M. Mitchell. 2009. "A Robust Measure of Heterogeneity for Ranking Earth Models: The F PHI Curve and Dynamic Lorenz Coefficient." *SPE Annual Technical Conference and Exhibition*.

Tanaka, Shusei, Dongjae Kam, Jiang Xie, Zhiming Wang, Xian-Huan Wen, Kaveh Dehghani, Hongquan Chen, and Akhil Datta-Gupta. 2017. "A Generalized Derivative-Free Rate Allocation Optimization for Water and Gas Flooding Using Streamline-Based Method." *SPE Annual Technical Conference and Exhibition*.

Thiele, Marco R., and Rod. P. Batycky. 2003. "Water Injection Optimization Using a Streamline-Based Workflow." *SPE Annual Technical Conference and Exhibition*.

Tian, Chuan, and Roland N. Horne. 2019. "Applying Machine-Learning Techniques To Interpret Flow-Rate, Pressure, and Temperature Data From Permanent Downhole Gauges." *SPE Reservoir Evaluation & Engineering* 22 (02):386-401.

Yousef, Ali A., Pablo H. Gentil, Jerry L. Jensen, and Larry W. Lake. 2006. "A Capacitance Model To Infer Interwell Connectivity From Production and Injection Rate Fluctuations." *SPE Reservoir Evaluation & Engineering* 9 (06):630-646.

Zhang, Yanbin, Jincong He, Changdong Yang, Jiang Xie, Robert Fitzmorris, and Xian-Huan Wen. 2018. "A Physics-Based Data-Driven Model for History Matching, Prediction, and Characterization of Unconventional Reservoirs." *SPE Journal* 23 (04):1105-1125.

Zhao, Hui, Zhijiang Kang, Xiansong Zhang, Haitao Sun, Lin Cao, and Albert C. Reynolds. 2016. "A Physics-Based Data-Driven Numerical Model for Reservoir History Matching and Prediction With a Field Application." *SPE Journal* 21 (06):2175-2194.

Zhao, Hui, Lingfei Xu, Zhenyu Guo, Qi Zhang, Wei Liu, and Xiaodong Kang. 2020. "Flow-Path Tracking Strategy in a Data-Driven Interwell Numerical Simulation Model for Waterflooding History Matching and Performance Prediction with Infill Wells." *SPE Journal* 25 (02):1007-1025.

APPENDIX A

STREAMLINE-BASED RATE ALLOCATION OPTIMIZATION

In this section, we summarize the streamline-based rate allocation optimization algorithm. Rate allocation optimization for multiphase flow systems is typically challenging because of the problem complexity and uncertainty of reservoir properties. Tanaka et al. (2017) provided a fast and robust derivative-free workflow that improves economic values by optimizing rate allocation using a streamline-based approach. Population-based optimization approaches such as Genetic Algorithm or Particle Swarm Optimization usually require many simulations. On the other hand, the streamline-based approach requires a very small number of simulations. Tanaka et al. (2017) showed that the proposed streamline-based approach gave comparable optimization performance to the population-based derivative-free techniques.

Several streamline-based rate allocation optimization workflows were established previously. Thiele et al. (2003) presented an optimization workflow that improves the oil cut for each well pair. Alhuthali et al. (2007) provided an optimization approach that equalizes the breakthrough time. However, these methods have some limitations. The breakthrough time approach may not work after the breakthrough. In contrast, the oil cut improvement approach works after a breakthrough. However, it may not be easy to apply it if a clear breakthrough cannot be detected (Tanaka et al. 2017).

The proposed approach estimates the expected economic performance, such as Net-Present-Value (NPV) using flow diagnostics information from streamline tracing. The expected NPV is used for rate reallocation in an iterative manner (Tanaka et al. 2017).

In the proposed approach, first, we need to calculate the streamline-related properties using static reservoir properties. Streamlines are flow trajectories associated with the volume of flowing fluid. The pore volume of each streamline is calculated as follows (Tanaka et al. 2017):

$$PV_\ell = \sum_{i=1}^{node} \frac{q_\ell}{\rho_{e,i}} \Delta\tau_i \tag{A.1}$$

Where,

q_ℓ : flow rate assigned to single streamline

ρ_e : effective density, dimensionless

$\Delta\tau$: time of flight within the node

The concept of effective density was presented by Osako et al. (2007) for considering the effect of compressibility of the fluid. For gas injection cases, this concept gets more important because of the highly compressible fluid injection. Once the pore volume is determined, the value of hydrocarbon in place along streamline becomes:

$$\Lambda_\ell = q_\ell \sum_{node} \frac{\Delta\tau}{\rho_e} \sum_{\alpha=oil,gas} S_\alpha b_\alpha R_\alpha \tag{A.2}$$

Where,

R_α : the price of fluid per unit volume

b_α : inverse of formation volume factor

S_α : phase saturation

This equation represents the maximum possible profit from the single streamline. Then, predictive NPV along streamline can be estimated as follows:

$$r_\ell = q_\ell \sum_{node} \frac{\Delta\tau}{\rho_e} \sum_{\alpha=oil,wat,gas} \left(S_\alpha b_\alpha R_\alpha\right)\left(1+d\right)^{-\tau_{ey} f_\alpha^{-1}} + q_\ell b_{\alpha i} R_{\alpha i} \sum_{t_i}^{t_e} \Delta t \left(1+d\right)^{-t_{ey}} \qquad \text{(A.3)}$$

Where,

f_α : fractional flow

t_i : arbitrary estimation start time

t_e : end of the reservoir lifetime

τ_{ey} : time of flight from producer

t_{ey} : remaining reservoir life ($=\left(t_e - t_i\right)/365$)

d : discount rate

$R_{\alpha i}$: the price of injection fluid per unit volume

$b_{\alpha i}$: inverse of formation volume factor of injection fluid

This equation estimates the NPV of a single streamline for an arbitrary period. In this equation, the producer contribution is considered in the first term, and the injector contribution is estimated in the second term. The production contribution is computed using the phase volume and travel time to the producer. The calculation of injector contribution uses physical time $t_e - t_i$. The discount term of the production contribution needs adjustment using fractional flow because a highly viscous or near irreducible phase

will have only a small contribution. Once the expected NPV is computed for all streamlines, we can estimate the NPV for field level, well pair level, and well level.

$$r_f = \sum_{N_\ell} r_\ell, \; r_{ip} = \sum_{N_{\ell,ip}} r_\ell, \; r_w = \sum_{N_{\ell,w}} r_\ell \tag{A.4}$$

Where,

N_ℓ: number of streamlines in the field

$N_{\ell,ip}$: number of streamlines for the injector producer pair

$N_{\ell,w}$: number of streamlines for target producer

From the expected NPV value, we can estimate the well pair efficiency as follows:

$$e_{ip,NPV} = \frac{r_{ip}}{\Lambda_{ip}} = \sum_{N_{\ell,ip}} \frac{r_\ell}{\Lambda_\ell} \tag{A.5}$$

The NPV-based efficiency $e_{ip,NPV}$ is a useful indicator of well pair connection performance to optimize the rate allocation. If it is 1.0, it says a large amount of hydrocarbon in place and easily transports. This value can be negative if the injection cost is higher than the production contribution. Thiele et al. (2003) presented a well allocation factor approach (WAFs), which computes the well pair efficiency as follows:

$$e_{ip,WAF} = \frac{Q_{o,ip}}{Q_{t,ip}} \approx \sum_{N_{\ell,ip}} \frac{f_{o,\ell}}{N_{\ell,ip}} \tag{A.6}$$

Where,

$Q_{o,ip}$: oil flow rate at surface condition:

$Q_{t,ip}$: total fluid flow rate at surface condition

$f_{o,\ell}$: the fractional flow of oil

Alhuthali et al. (2007) presented a method for equalizing the breakthrough time (EqAT). The efficiency can be computed as follows:

$$e_{ip,EqA} = \frac{\tau_{ip}}{\tau_{fm}} = \frac{N_\ell \sum_{N_{\ell,ip}} \tau_\ell}{N_{\ell,ip} \sum_{N_\ell} \tau_\ell} \tag{A.7}$$

Where,

τ_{ip} : average time of flight for well pair

τ_{fm}: average time of flight for field

These two streamline-based rate allocation approaches try to minimize the variance of the efficiency and optimize the allocation. More concretely, minimizing $\mathrm{var}(e_{ip,EqA})$ is equivalent to equalize the breakthrough time, and minimizing $\mathrm{var}(e_{ip,WAF})$ leads to improve the oil cut. As mentioned, these approaches have limitations. For example, EqAT may not work after a breakthrough, and WAFs do not work before a breakthrough. On the other hand, the proposed NPV estimation approach is applicable for both before and after breakthroughs. However, in some cases, the objective is not to maximize the NPV but maximize the early time production or balance injection front propagation. In these situations, the WAFs are more profitable. From this point of view, general efficiency was defined by combining three efficiency measures:

$$e_{ip} = \left(e_{ip,NPV}\right)^\alpha \left(e_{ip,EqA}\right)^\beta \left(e_{ip,WAF}\right)^\gamma \tag{A.8}$$

The hyperparameters α , β , γ are determined by users based on the objective of the problem. By using the well pair efficiency, the connection flow rates are updated as follows:

$$q_{ip}^{ite+1} = q_{ip}^{ite}\left(1-\eta+2\eta e_{ip}\right) \tag{A.9}$$

Where,

q_{ip} : connection flow rate
ite : iteration index
η : fraction of the allowable rate changes per iteration

Generally, implementing this computation with a small value of η and a large number of iterations lead to the success of rate allocation optimization. The summary of the workflow is presented in **Figure A.1**. First, we run the simulation until the target timestep. Then, run the short time simulation where we are going to optimize the rate allocation. Trace streamlines and evaluate NPV for each well pair. Based on the NPV value, compute the efficiency of each well pair and update the connection flow rate. After that, we modify the connection flow rate to satisfy the well level constraint, such as maximum flow rate or pressure drawdown. Then, again, scale the connection flow rate to satisfy the field level constraint. This procedure will be repeated until it converges or it reaches the maximum number of iterations. These steps are repeated for every timestep where we want to optimize the rate allocation.

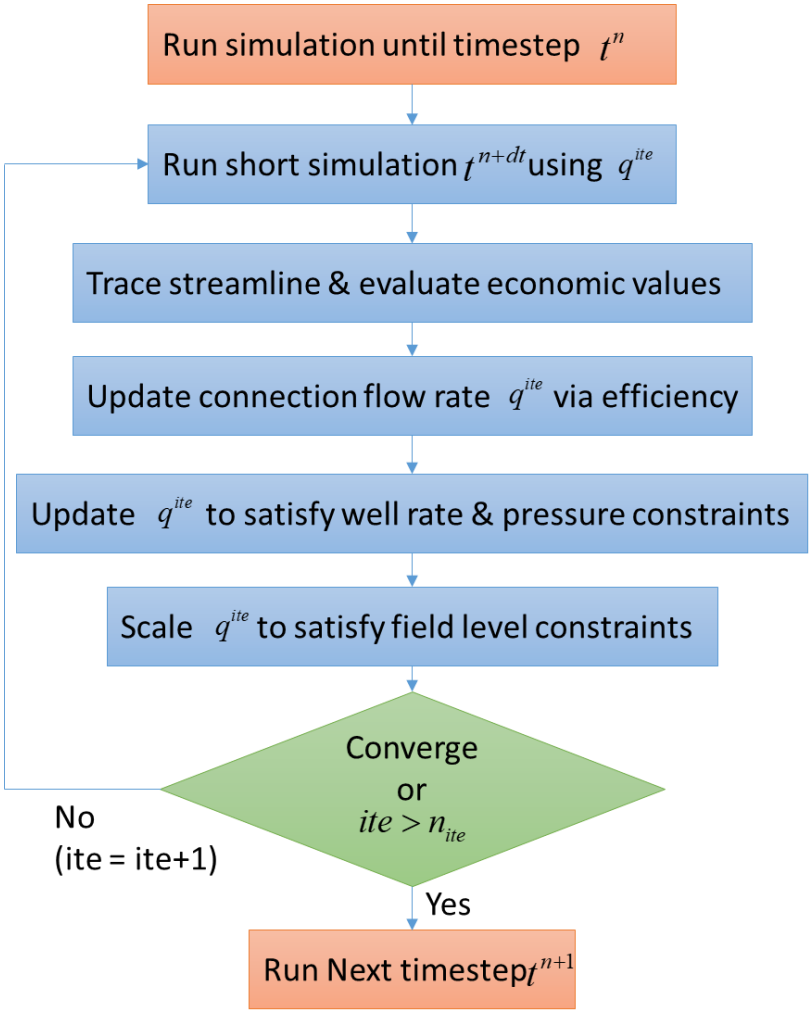


Figure A.1: The flowchart of streamline-based rate allocation optimization workflow