DELFT UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Machine Learning Based Error Modeling for Surrogate Model in Oil Reservoir Problem

---

Author:
Jie HUANG

Advisors:
Prof. Dr. Ir. Hai Xiang LIN
MSc. Cong XIAO

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Applied Mathematics

February 13, 2019

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

EEMCS
Applied Mathematics

Master of Science

Machine Learning Based Error Modeling for Surrogate Model in Oil Reservoir Problem

by Jie HUANG

This thesis focuses on the construction and optimization of a prediction model for the errors resulting from a model order reduction (MOR) procedure in oil reservoir simulation. MOR is a numerical technique that projects the physical based model, which is also called the high-fidelity model (HFM), into a lower dimension by using matrix decomposition, such that the computational speed can be greatly increased. The reduced order model (ROM) is also known as surrogate model. Obviously, error occurs during the projection process. We want to estimate this error and predict it through building an error model, and to fortify the surrogate model by adapting a parameter estimation. In this thesis, three statistical methods will be adapted to our problem, including least absolute shrinkage and selection operator (LASSO) and two machine learning (ML) methods: long short term memory (LSTM) and fully-connected recurrent neural network (RNN). The training data is the error of the ROM, which is defined as the difference between the ROM values and HFM values. Efforts have also been made to improve the performance of the error model, including the pre-processing of the data, and several model optimization techniques. The model order reduction method here is a non-intrusive subdomain POD-RBF algorithm, which treats subsurface oil-water flow data by adapting domain decomposition (DD), radial basis function (RBF) and proper orthogonal decomposition (POD). The high-fidelity model is generated by Matlab reservoir simulation toolbox (MRST). The error is defined as the difference between the HFM data and the ROM data. Through the comparison of several statistical models, this error can be best predicted by an optimized traditional recurrent neural network.

Acknowledgements

First, I would like to thank my thesis supervisor Prof.Lin of the Apply Mathematics at TU Delft. He has always been nice and knowledgeable, and has taught me so much. He allowed me to follow my own plan, but steered me in the right direction whenever he thought I needed it.

I would also like to thank my second advisor Cong Xiao. He has helped me so much on the work. Without his passionate input and solid academic levels, the experiments could not have been so successful.

Finally, I must give my very sincere gratitude to my parents for providing me with living support and encouragement during my years of study and the process of writing thesis. This accomplishment would not have been possible without them.

. . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research problem

Oil is one of the most important sources in modern industry, and predicting the changes of an oil reservoir has always been of major interest. Owing to the uneven distribution of an oil reservoir and the complex geological structure, this becomes a hard task. Fortunately, modern computational science has provided a way of simulating the ongoing changes underground through a combination of several prospected data. Theses simulations are physical based, usually include the data of several key parameters in a large area, that are recorded continuously for decades. Obviously, the physical models can be extremely large and highly non-linear and, thus computational expensive. Naturally, research efforts have been made to accelerate the computation by linearizing the model, and reducing the number of the elements involved in the computation. One of the ideas is to apply model order reduction (MOR)[4], which means to project the physical based model, also known as the original high-fidelity model (HFM), into a lower dimension by using matrix decomposition, such then there will be fewer elements joining the computation. The reduced order model (ROM) is also called a surrogate model. However, error occurs during the linearization (if adapted), interpolation and projection process, and we wish to compensate for it. So the main concern of this thesis is to predict the error by building an error model, fortify the surrogate model with this error model, and to further estimate the accuracy of the fortified surrogate model by parameter estimation.

The method for error modelling is building statistical error models that concern little of the physics but much about the regular pattern of the previous data. It requires an input and an output data set, and in our case these data sets are the ROM errors at different time steps. The error is defined as the difference between a HFM pressure or saturation value, and a ROM pressure or saturation value that has been projected back into the full dimensional space. As the HFM values are always non-linear with the physical parameters, in most cases the errors are also non-linear, which is most suitable for statistical methods. The statistical methods require only the input and output variables whether they are linear or not, and can also achieve high accuracy, provided a sufficient large dataset. Therefore, statistical methods will be adapted to predict the error of ROMs in my work, and furthermore to operate a parameter estimation.

The error model should predict error for every meaningful point in a concerned area, thus it conforms the definition of the regression problem in machine learning. Hence the first method we considered is "least absolute shrinkage and selection operator" (LASSO)[38]. It is a regression analysis technique which performs variable selection and regularization to enhance the prediction accuracy of the statistical model. Previous works have applied LASSO on constructing error models of surrogate models[6], thus LASSO is used as a comparison in this thesis. In the training process of LASSO, data from all points and time steps are taken into one regression. It shows better result in the linear or weak-nonlinear cases. However, in most situations, errors are distributed in non-linear way, thus limiting the capability of LASSO.

Fortunately, the development of machine learning (ML) has provided some more generic solutions. ML algorithms build mathematical models of given data, also named as "training data", and proceed classifications or predictions without being specifically programmed for certain task. Recurrent neural network (RNN) and long short-term memory (LSTM) are two representative algorithms among ML methods that will be used in our situation, and a fully connected multilayer perceptron (MLP) technique is adapted. Previous work[6] have shown the feasibility of such algorithm as random forest[36], and through my work, the capability of RNN, LSTM and MLP will be presented.

A recurrent neural network (RNN) is a class of neural network that are especially powerful for dealing with data in a sequence. MLP means a class of supervised multilayer fully connected neural network, here it refers to the two layer fully connected RNN. In my work, this is used to train the data from a certain time step under different parameters. It shows the common pattern of the error under different geometric conditions, and by sampling from different timings, it could also show the error's changing trend as time goes. However, in order to predict the precise change on the timeline, LSTM comes first on the table.

Long Short-Term Memory (LSTM) is a time-recurrent neural network (RNN), the related research was first published in 1997[30]. LSTM is suitable for processing and predicting important events with long intervals in time series[35] owing to its special structure. It is used in this thesis to predict the future error under a certain control.

All these statistical methods have their own advantages and limitations for our cases. And they have to be carefully investigated and adjusted for different data cases. Thus different attempts have been made to optimize the valuable models in different experiments, including pre-processing the training data, adjusting neural network parameters and clustering the training set accordingly. Pre-processing includes cropping the data based on its current time step and parameter, transforming the data with targetive function, domain decomposition[19] and normalliztion. Different optomization procedures[34] such as stochastic gradient descent and conjugate gradient descent, and several loss functions such as absolute loss and quadratic loss are also taken into comparison. The parameter estimation is obtained by least square method[20].

1.2 Data source

The training data is the error between HFM variables and ROM variables. In the prediction problems, we try to predict the future error based on the previous experience, so the input of the training set would be the error at time step n, and the output is the error at time step $n + 1$. According to the need in the oil industry, pressure and saturation are the two most important measurable variables that can represent the production of an oil well. Therefore, these two variables will be used in the experiments in this thesis.

The high-fidelity model is generated by Matlab reservoir simulation toolbox (MRST), and the model order reduction method here is a non-intrusive subdomain POD-RBF algorithm introduced by Cong Xiao et al[5]. It treats subsurface oil-water flow data by adapting domain decomposition (DD), proper orthogonal decomposition (POD) and radial basis function (RBF). The main idea of this method is to perform high-fidelity training simulations, save 'snapshots' (also called state vectors) at time steps from these simulations, and then obtain a set of basis functions from these snapshots[16]. RBF interpolation is a data-based response surface method, and has been applied to reservoir problems[10] and fluid dynamics[24] successfully. Here it is combined with a domain decomposition (DD) technique and constructed as different local RBF interpolations according to local flow dynamics. The above procedure is the complete construction of a surrogate model for oil reservoir problem, and the non-ignorable error during the process is our major concern. Combination of this ROM with our error model will create a reliable and fully non-intrusive model that benefits the oil reservoir researchers.

1.3 Objective

The primary objective of this research is presented as follows:

To develop statistical models for predicting the errors caused by the order reduction of an oil reservoir simulation.

The primary objective can be further divided into following sub-objectives:

1.To develop an LSTM-based model that predicts the pressure and saturation error of the next step under a fixed permeability condition.

2.To develop an RNN-based model that predicts the pressure and saturation error of a new permeability under a fixed time step.

3.To develop an LASSO-based model that predicts the pressure and saturation error of the next time step.

4.To compare the aforementioned models and optimize the machine learning models by pre-processing and clustering the data, and selecting suitable optimizers.

1.4 Outline of this thesis

This thesis proceeds as follows. In Chapter 2 and Chapter 3, we present the background knowledge of the statistical methodology and the model order reduction. In Chapter 4, the development of the error model for reduced order model will be presented and compared, and this error model will be improved through pre-processing the data and inserting an optimizer. In Chapter 5, the error model will be applied to an application case through parameter estimation. In the last chapter, conclusions and discussion of future research will be presented.

Chapter 2

Introduction to the Surrogate Model

In this chapter, the background knowledge of the oil reservoir simulation and the construction of its surrogate model will be introduced. The first section describes the general way of forming an oil reservoir model, the equation that governs the oil and water flow, and the conditions that we have adapted in our experiments. The second section introduces the linearization and the process of order reduction of the physical based model which is described by the discretized form of a transformed governing equation.

2.1 Construction of the Physical Based Model

A simplified 2-dimensional reservoir model is represented as a horizontal layer of multiple grid cells[1]. At some locations in the reservoir area some injection wells are installed, while usually around them are some production wells. The decision on the placement of the wells is based on the knowledge about the oil reservoir. For every grid cell, permeability, porosity,initial saturation and initial pressure are defined. In this thesis, permeability is used as a control variable, while the other initial conditions are defined as constant. And the quantities of interest are the changing pressure and saturation along with time.

The model we took into experiments is formed by Matlab Reservoir Simulation Toolbox. It is a highly simplified model based on the assumptions that porosity is constant over time, compressibility is isothermal, the displacement of oil by water is immiscible and that there is no gravity affect[4].

A brief derivation of the equation that governs the oil-water flow is described below, for more details the readers are referred to [2].

Under the assumption that fluid density does not change in space and time, by combining the mass balance of each fluid and solid phase and Darcy's law, the equation governing oil-water flow can be written as:

$$\partial_t(\phi S_j) + \nabla(f(S_w)v + \phi S_j \partial_t d) = 0 \tag{2.1}$$

where S_j represents the saturation of phase j, j=o for oil or w for water. v_j stands for the Darcys velocity at j fluid phase, $\partial_t d$ for the velocity, with $d = [d_1, d_2]$ for the displacement. The relation between the j fluid phase velocity and the fluid pressure is determined by Darcys Law:

$$v_j = -\lambda_j(S_j)k(x)k(\phi)\nabla p_j \tag{2.2}$$

where $\lambda_j(S)$ is the phase mobility of fluid j, and $k(x)k(\phi)$ represents the permeability of the rock.

We take the quantities of interest as pressure p and water saturation S. A fully-implicit finite volume procedure is used to form a discrete representation of Eq.2.3. Defining $x = [p_o, S_w]$ as the state vector and u as the well control (in this thesis the permeability), the discrete system for the governing equation can be written as:

$$g(x^{n+1}, x^n, u^{n+1}) = A(x^{n+1}, x^n) + F(x^{n+1}) + Q(x^{n+1}, u^{n+1}) \tag{2.3}$$

Here g is the residual that should be derived to 0, n and $n+1$ are time steps, and A, F and Q are the discretized accumulation, flux and source/sink terms.

2.2 Construction of the Surrogate Model

Traditionally, the discretized residual equation is solved by a full-order simulator, and this can be computational expensive. So the following section will introduce a promising surrogate simulator[5], whose error prediction is the major concern of this thesis.

The surrogate model (or ROM) we applied here is a non-intrusive subdomain POD-RBF algorithm. Generally, ROMs can be intrusive or nonintrusive. Intrusive ROMs require participation of the HFM simulator's source code, while the nonintrusive ROMs require only the HFM outputs, such as the derivative matrices created during the pre-processing. The further 'test' runs (involving new sets of well controls) are operated outside the HFM simulator. Here the 'training' and 'test' are to describe the model order reduction process, and they are different from the training and test runs in machine learning technique that we are going to mention in the next section.

We now briefly introduce the subdomain POD-RBF formulation for oil-water systems, more details are introduced in[5]. The general idea is to perform the order reduction by doing proper orthogonal reduction (POD), then obtain the derivation items in the derived residual equation at any given time around a single point by interpolating radial basis function (RBF)

Before the model order reduction, the data is first employed a domain decomposition process as shown in Fig.2.1. The 2D or 3D physical domain for a dynamic model is denoted as Ω. The whole computational domain Ω is decomposed into several subdomains Ω^d accordingly, $d \in 1, 2, ..., S$ and on each subdomain the unknowns are calculated locally,e.g.,local pressure and saturation variables. In each subdomain Ω^d, the local state vectors are used to construct its local POD basis functions ϕ^d and the corresponding POD coefficient $\psi^{d,n+1}$ at the time step $n+1$. For each subdomain Ω^d, the reservoir dynamic model in the reduced subspace is modified to represent the underlying dynamic system associated with this subdomain Ω^d and its surrounding subdomains Ω^{sd} by considering the dynamic interaction between these subdomains, The well model represents the underlying dynamic system just associated with this subdomain Ω^d where the well is located in.

Suppose the current state is x^n, to determine x^{n+1}, we derive Eq.2.4 around the state (x^{i+1}, x^i, u^{i+1}). This gives

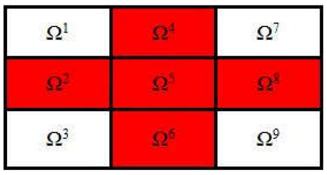

FIGURE 2.1: Illustration of domain decomposition in a 2-D case

$$g^{n+1} = g^{i+1} + \frac{\partial g^{i+1}}{\partial x^{i+1}}(x^{n+1} - x^{i+1}) + \frac{\partial g^{i+1}}{\partial x^{i}}(x^{n} - x^{i}) + \frac{\partial g^{i+1}}{\partial u^{i+1}}(u^{n+1} - u^{i+1}) \tag{2.4}$$

where $g^{n+1} = g(x^{n+1}, x^n, u^{n+1})$ and $g^{i+1} = g(x^{i+1}, x^i, u^{i+1})$.

To reduce the dimension of

$$R^{i+1} = \frac{\partial g^{i+1}}{\partial x^{i+1}} \tag{2.5}$$

A proper orthogonal decomposition is employed for linear order reduction. We represent the state vector x by a reduced state z and a basis matrix Φ:

$$x \approx \Phi z \tag{2.6}$$

where $z \in R^l$ is a low-dimensional variable, then the Φ matrix can be constructed by proper orthogonal decomposition (POD). It representes the high-dimension space by a set of orthogonal basis vectors, these are the singular vectors of the 'snapshot' matrices.

This gives

$$R_r{}^{i+1}(z^{n+1} - z^{i+1}) = -[(\frac{\partial A_r{}^{i+1}}{\partial x^i})_r(z^n - z^i) + (\frac{\partial Q_r{}^{i+1}}{\partial u^{i+1}})_r(u^{n+1} - u^{i+1})] \tag{2.7}$$

where

$$R_r{}^{i+1} = \Phi^T R^{i+1}\Phi, (\frac{\partial A_r{}^{i+1}}{\partial x^i})_r = \Phi^T[(\frac{\partial A^{i+1}}{\partial x^i})\Phi, (\frac{\partial Q^{i+1}}{\partial u^{i+1}})_r = \Phi^T(\frac{\partial Q^{i+1}}{\partial u^{i+1}}) \tag{2.8}$$

The gradient

$$R^{i+1} = \frac{\partial g^{i+1}}{\partial x^{i+1}} \tag{2.9}$$

is obtained by a radial basis function (RBF) interpolation into traditional POD scheme. $R^{n+1}(Z^{d,n}, Z^{sd,n+1}, u)$ denote a RBF interpolation function for the POD coefficient $Z^{d,n+1}$ at the time level $n+1$ for the subdomain Ω^d from the set $(Z^{d,n}, Z^{sd,n+1}, u)$. The vector u denotes the set of reduced parameter's coefficient for the full physical domain, which is the permeability k in our case. The RBF interpolation function is a linear combination of M radial basis functions in the form of

$$R^{d,n+1}(Z^{d,n}, Z^{sd,n+1}, u) = \sum_{j=1}^{M} \omega_j{}^{d,n+1} * \theta \left\|(Z^{d,n}, Z^{sd,n+1}, u) - (Z_j{}^{d,n}, Z_j{}^{sd,n+1}, u_j)\right\| \tag{2.10}$$

Here, ${\omega_j}^{d,n+1}$ is a weight coefficient vector. $\left\|(Z^{d,n}, Z^{sd,n+1}, u) - ({Z_j}^{d,n}, {Z_j}^{sd,n+1}, u_j)\right\|$ is a scalar distance by L_2 norm. θ is a series of radial basis functions based on different centres $(Z_j^{d,n}, Z_j^{sd,n+1}, u_j)$. The specific coefficient ${\omega_j}^{d,n+1}$ is determined as such to guarantee that the interpolation function $R^{d,n+1}$ at the current data points $(Z^{d,n}, Z^{sd,n+1}, u)$, matches the given data $Z^{sd,n+1}$ exactly.

Among several radial basis functions, we chose the multiquadratic radial basis function for our problem, its general form is

$$\theta(l) = \sqrt[2]{l^2 + u^2}$$

in which l stands for the scalar distance from the origin

$$l = |(\mathbf{Z^{d,n}}, \mathbf{Z^{sd,n+1}}, \mathbf{u}) - (\mathbf{Z_j}^{\mathbf{d,n}}, \mathbf{Z_j}^{\mathbf{sd,n+1}}, \mathbf{u_j})|$$

After the whole order reduction process, the variables of the previous step $v^i = (x^n - x^i)$ and the current step $v^{i+1} = (x^{n+1} - x^{i+1})$ will be collected for constructing the error model, and the permeability $u(u^{n+1} - u^{i+1})$ (it does not change with the time) will be a major influencing factor to be considered. When the variable is pressure, v^i and v^{i+1} are represented by p^i and p^{i+1}, when the variable is saturation, they are represented by s^i and s^{i+1}.

Chapter 3

Statistical Methodology

The model order reduction process introduces error during the projection and interpolation operations. In order to fortify the ROM, we want to predict the error as precisely as possible. For this purpose, three statistical methods are considered, and three corresponding error models will be constructed, optimized and compared. One is a linear regression analysis method called LASSO, the others are two machine learning based methods, the traditional RNN and LSTM NN. In this chapter, background knowledge about these methods will be introduced.

3.1 LASSO

The first method I have applied is "Least absolute shrinkage and selection operator" (also LASSO). It is a commonly used linear regression technique[38], and have been adapted for error modelling with the combination of a machine learning based classification in [6].

Suppose a data set consists of N samples, each of them consists of p covariates. Let $x_i := (x_1, x_2, \dots, x_p)^T$ be the covariate vector at the state i and y_i be the output. Then the goal of LASSO is to solve

$$\min_{\beta_0,\beta} \left\{ \frac{1}{N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 \right\} \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t. \tag{3.1}$$

Here t is a parameter which determines the amount of regularisation. Let X be the covariate matrix

$$X_{ij} = (x_i)_j \tag{3.2}$$

and x_i be the i th column of X, this expression can be written into

$$\min_{\beta_0,\beta} \left\{ \frac{1}{N} \|y - \beta_0 - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq t. \tag{3.3}$$

where

$$\|\beta\|_p = \left(\sum_{i=1}^{N} |\beta_i|^p \right)^{1/p} \tag{3.4}$$

is the standard ℓ^p norm.

Since

$$\hat{\beta}_0 = \bar{y} - \bar{x}^T \beta, \tag{3.5}$$

so that

$$y_i - \hat{\beta}_0 - x_i^T\beta = y_i - (\bar{y} - \bar{x}^T\beta) - x_i^T\beta = (y_i - \bar{y}) - (x_i - \bar{x})^T\beta, \tag{3.6}$$

Now Eq.3.3 can be written in the Lagrangian form,

$$\min\left\{\frac{1}{N}\|y - X\beta\|_2^2\right\} \text{ subject to } \|\beta\|_1 \leq t. \tag{3.7}$$

where the relationship between t and λ depends on the data.

3.2 Recurrent Neural Network

In this study, two machine learning techniques have also been used in the experiment. One is the recurrent neural network (RNN).

RNN is made to deal with sequential data, such as time series. In the traditional neural networks, we assume that the inputs and outputs are independent from each other. RNNs are called recurrent because they proceed the same task for every case of a sequence, and the outputs depend on the previous computation steps. They have memory cells to capture values that have already been calculated. The basic RNN is constructed as a network of neurons (nodes) organized in an input layer, an output later, and one or more hidden layers. Each neuron is connected with all the other neurons in the next layer. Each connection has an real-time weight. Here is the structure of a typical RNN:

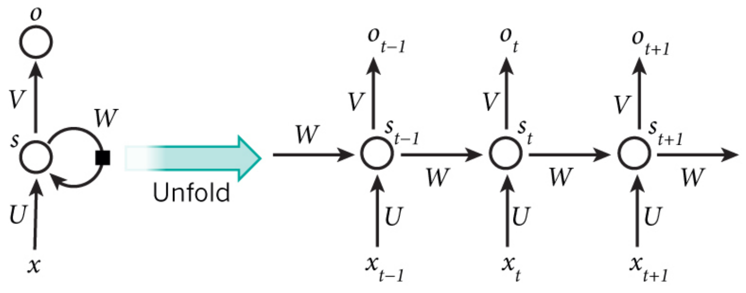

FIGURE 3.1: A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: soybean yield prediction, chapter: the full story, section: neural networks

The above figure shows a recurrent neural network being unfolded into a full network. The governing formulas of the computation happening in a RNN are as follows: $x_t$is the input at step t. s_t is the hidden state at step t. It is the memory of the network. s_t is computed based on its last hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. s_{-1} is typically determined to be 0. o_t is the output at step t.

3.3 Long Short-Term Memory

The disadvantage of RNN is that as the time steps increase, it fails to derive context from time steps which are much far behind. To solve this problem, long short-term memory (LSTM) is designed. It is a special kind of RNN that able to learn long-term dependencies.

The structure of an LSTM network is the same as an RNN, but the repeating module is more complicated. Unlike in RNN which often has a single tanh layer, it consists of an input gate, a memory cell, a forget gate and an output gate, as in Fig.3.2.

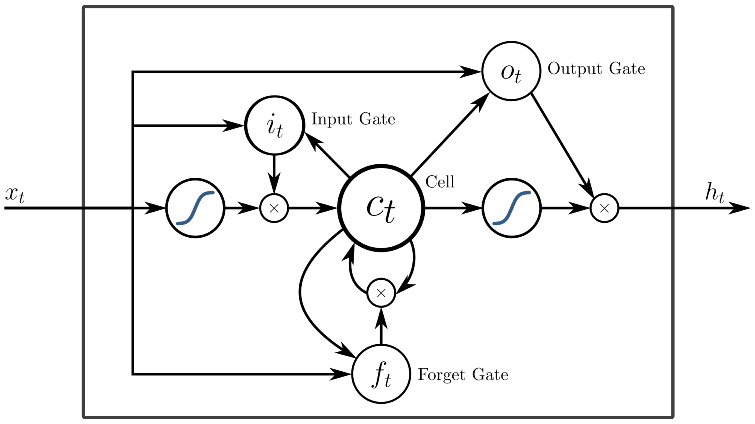

FIGURE 3.2: A LSTM module. Source: Wikipedia.

The LSTM cells store values for a time period by using an identity activation function for the memory cell. In this case, the gradient does not vanish when an LSTM network operates back-propagation.

The LSTM gates compute interactions using the logistic function. The input gate controls the range of new value that can be taken into the cell, the forget gate controls whether a value stays in the cell and the output gate determines if a value in the cell can be used to compute the output activation of the LSTM unit. Each gate has its own weight and bias. The weights of the connections between the gates are adjusted during the training. This is achieved by minimizing δ_t:

$$\delta_t = \partial E / \partial h_t \tag{3.8}$$

in which δ_t is the error (it is not the error of ROM) at step t, E is the cost function, and h_t is the output at step t. The minimization is performed by gradient descent, see Fig3.3.

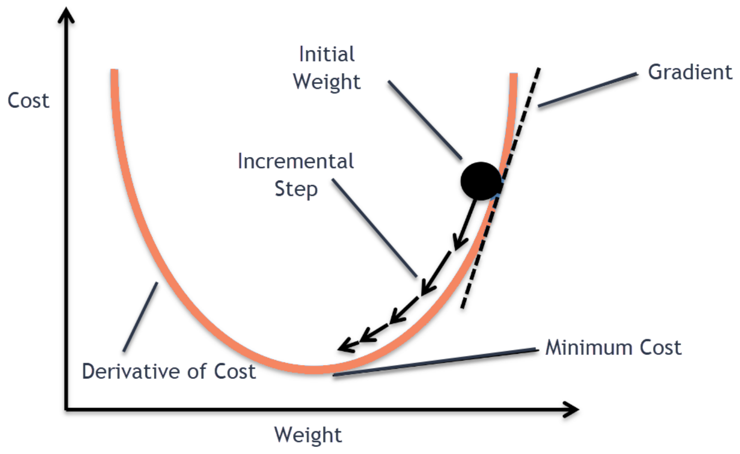

FIGURE 3.3: Minimize cost function by gradient descent. Source: https://www.imagenesmy.com

The selection of proper cost function and the gradient descent optimization method is explained in the next chapter.

Chapter 4

Error Modelling

This chapter describes the experimental results of statistical error models. The first step is to acquire the dataset by means of simulations with both HFM and ROM, here the same case as in [5] is used. Next, three regression models based on LASSO, RNN and LSTM will be constructed and optimized for our interested variables (pressure and saturation), their results will be reported and analyzed.

4.1 Data Acquisition

As mentioned in section 2.1, the derived governing equation is represented as

$$g^{n+1} = g^{i+1} + \frac{\partial g^{i+1}}{\partial x^{i+1}}(x^{n+1} - x^{i+1}) + \frac{\partial g^{i+1}}{\partial x^{i}}(x^{n} - x^{i}) + \frac{\partial g^{i+1}}{\partial u^{i+1}}(u^{n+1} - u^{i+1}) \tag{4.1}$$

in which the variables of the previous step $v^i = (x^n - x^i)$ are collected as the input set, and that of the current step $v^{i+1} = (x^{n+1} - x^{i+1})$ are collected as the output set. The permeability $U = (u^{n+1} - u^{i+1})$ (that does not change with the time) and time step t are two major influencing factors to be considered. When the variable is pressure, v^i and v^{i+1} are respectively denoted by p^i and p^{i+1}; and when the variable is saturation, they are represented by s^i and s^{i+1} As a regression problem, the variables along with the permeability U will be considered in the global domain for a lasting time.

The test model is based on a 2D synthetic model containing 9 wells on a 30 by 30 domain under 1000 different permeability conditions, and lasts for 200 time steps. So there are totally 900*200*1000 data for each variable. The construction proceeds as follows.

A 2D heterogeneous oil-water reservoir is considered with two-phase imcompressible flow dynamics. The reservoir contains 8 producers and 1 injector, which are labeled as P_1 to P_8, and I_1 respectively in Fig.4.1. The colour bar labels the permeability state in the field.

Next, an ensemble of 1000 Gaussian-distributed realizations of log-permeability is generated. We assume that the log-permeability fields are not conditioned to the permeability values at the well locations. The log-permeability fields and the corresponding porosity fields are described as the following:

$$\sigma_\beta = 5 \tag{4.2}$$

$$\mathbf{C}_\beta(x_{i1,j1}; y_{i2,j2}] = \sigma_\beta^2 e^{-[(\frac{|x_{i1}-x_{i2}|}{\chi_x})^2 + (\frac{|y_{i1}-y_{i2}|}{\chi_y})^2]} \tag{4.3}$$

$$\frac{\chi_x}{L_x} = 0.2, \frac{\chi_y}{L_y} = 0.2 \tag{4.4}$$

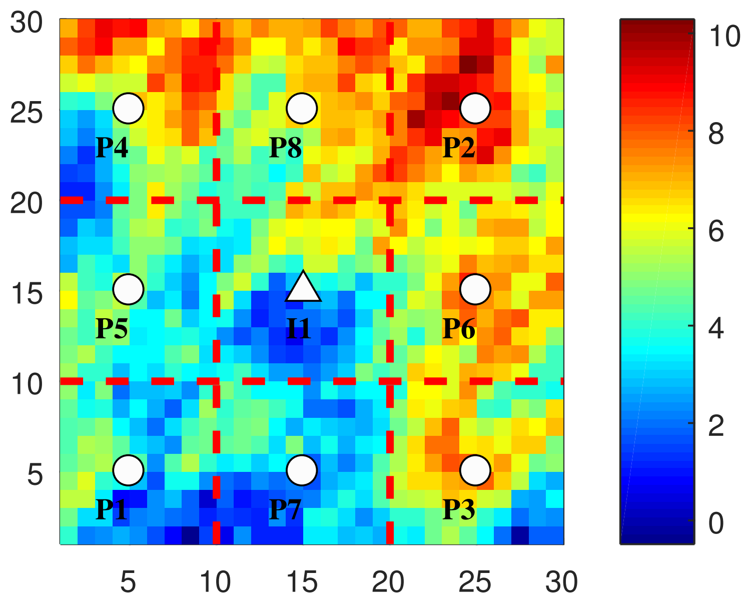

FIGURE 4.1: The well placement in the 2-D reservoir model

TABLE 4.1: Experiment settings using MRST for case 1

Description	**Value**
Dimensions	30 $\times$ 30 $\times$1
Grid cell size	10 $\times$ 10 $\times$ 10
Number of wells	8 producers, 1 injector
Fluid density	1014 kg/m^3, 859 kg/m^3
Fluid viscosity	0.4 mP·s, 2 mP·s
Initial pressure	30 MPa
Initial saturation	S$_o$=0.80, S$_w$=0.20
Connate water saturation	S_{wc}=0.20
Residual oil saturation	S_{or}=0.20
Corey exponent, oil	4.0
Corey exponent, water	4.0
Injection rate	200m^3/d
BHP	25MPa
History production time	5 year
Prediction time	10 year
Timestep	0.1 year
Measurement timestep	0.2 year

$$\boldsymbol{\phi} = 0.25(\frac{e^{\beta}}{200})^{0.1} \tag{4.5}$$

Here, σ_β is the standard deviation of log-permeability $\boldsymbol{\beta}$; C_β is the covariance of $\boldsymbol{\beta}$; $x_{i1,j1}$=(x_{i1},y_{j1}) denotes the coordinates of a grid block; χ_x (or χ_y) is the correlation length in x (or y) direction; and L_x (or L_y) is the domain length in x (or y) direction. The background log-permeability $\boldsymbol{\beta}_b$ is taken as the average of the 1000 realizations. The permeability field was parameterized using KL-expansion, resulting in 18 permeability patterns with $l_\beta = 18$ corresponding independent coefficients, which are used as a low-dimensional representation of the 900 grid blocks' permeability values.

Having reduced the parameter space, the next step is to reduce the reservoir model. The first step is to perform a set of training runs and take snapshots. Since the required number of training runs is unknown, the following procedure is introduced: (1) generate a sample coefficient vector by sampling from the set $\{-1, 1\}$, (2) run a high fidelity model simulation with these parameters, (3) extract snapshots and form the snapshot matrix, (4) compute a SVD of the snapshot matrix (5) repeat steps (1) to (4) until changes in the singular values are insignificant. This produced for this case a set of 15 training runs and 240 snapshots for pressure and saturation each. For each subdomain, two separate eigenvalue problems for pressure and saturation are solved using proper orthogonal decomposition.

Fig4.2, Fig4.3, Fig4.4, Fig4.5, Fig4.6 and Fig4.7 are the visual examples of the HFM value, ROM value and error distribution of pressure and saturation at time step 25, 50 and 150, with three different permeability conditions.

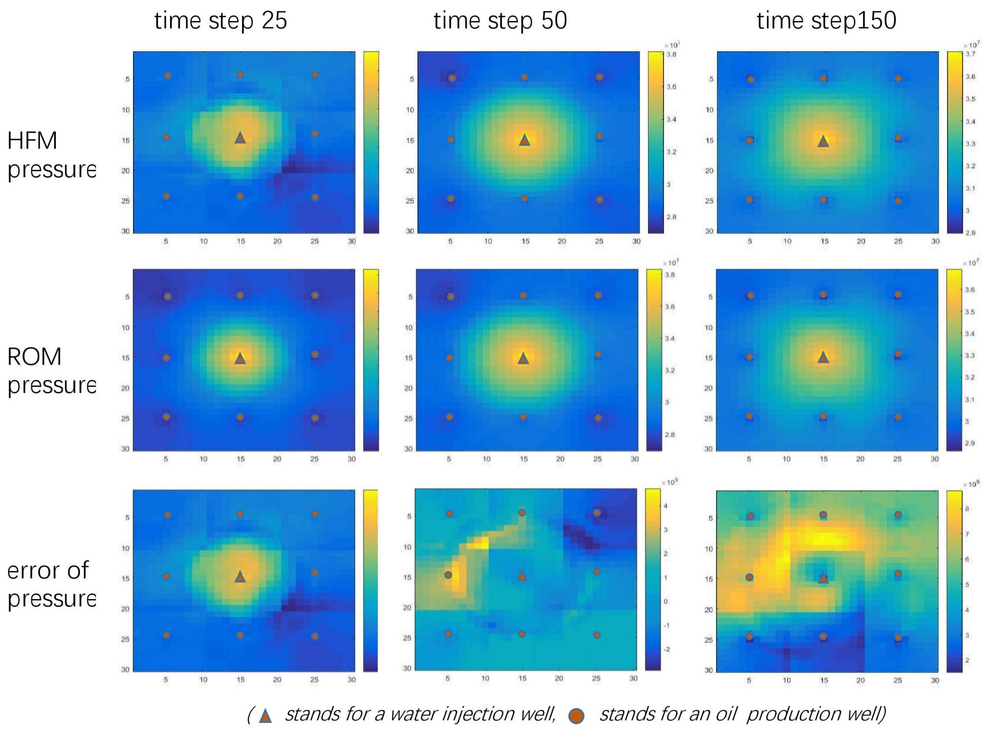

FIGURE 4.2: The pressure value of HFM and ROM and the pressure error at permeability condition U1

In the examples, saturation errors have shown a ringed shape, while on the inside and outside of this ring, the errors are extremely small. This is because the saturation data is first small around the water injection well and larger around the oil production wells, and as the extraction proceeds, the equilibrium between oil phase and the gas phase breaks, the the saturation gradually decreases around the oil production well. During the POD-RBF training, each step is based on the result of its last step. Thus, the error of a current step is the accumulation of the error of all the previous steps. According to the observation, on this 'ring', the errors at the outer edge are always positive, while at the inner edge they are negative. Therefore, after the accumulation, the current errors inside the ring are offset with their last step's error and become very small, remaining only a ring of the same size as

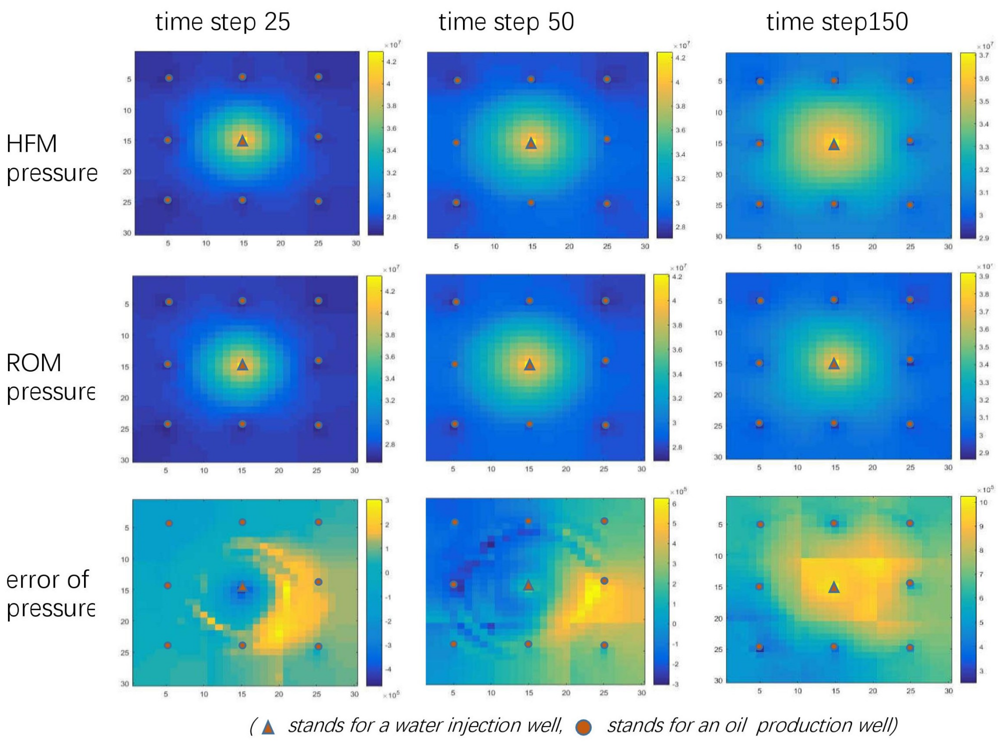

FIGURE 4.3: The pressure value of HFM and ROM and the pressure error at permeability condition U2

how much the saturation drop has diffused in one time step. This special shape has caused some extra optimization in the error modelling, as will be described in the next section.

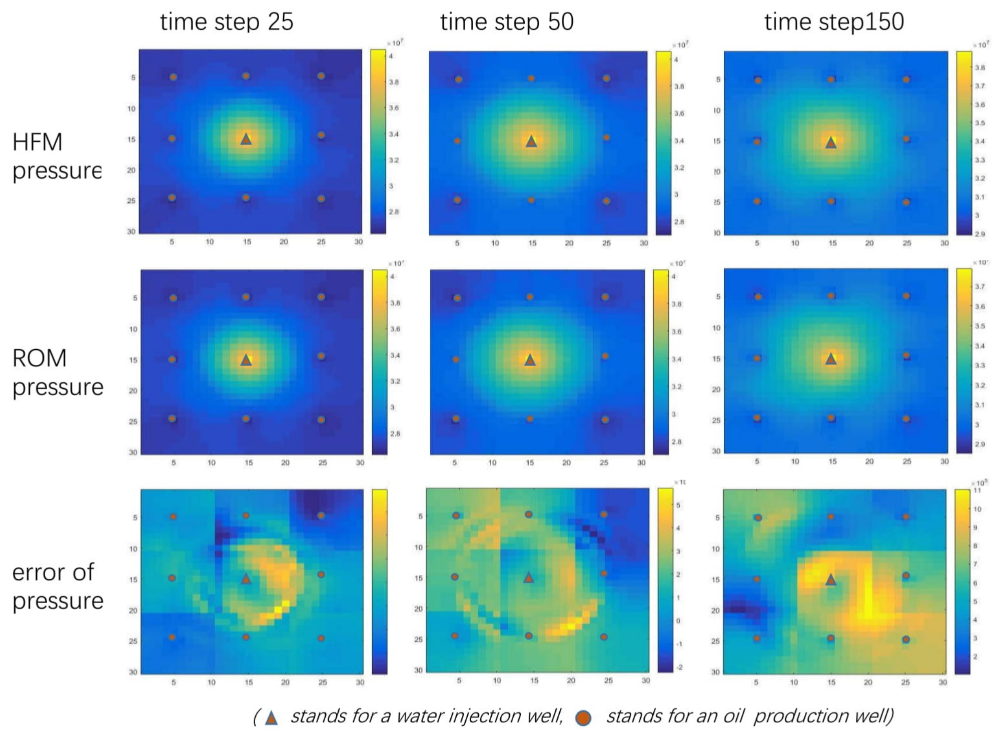

FIGURE 4.4: The pressure value of HFM and ROM and the pressure error at permeability condition U3

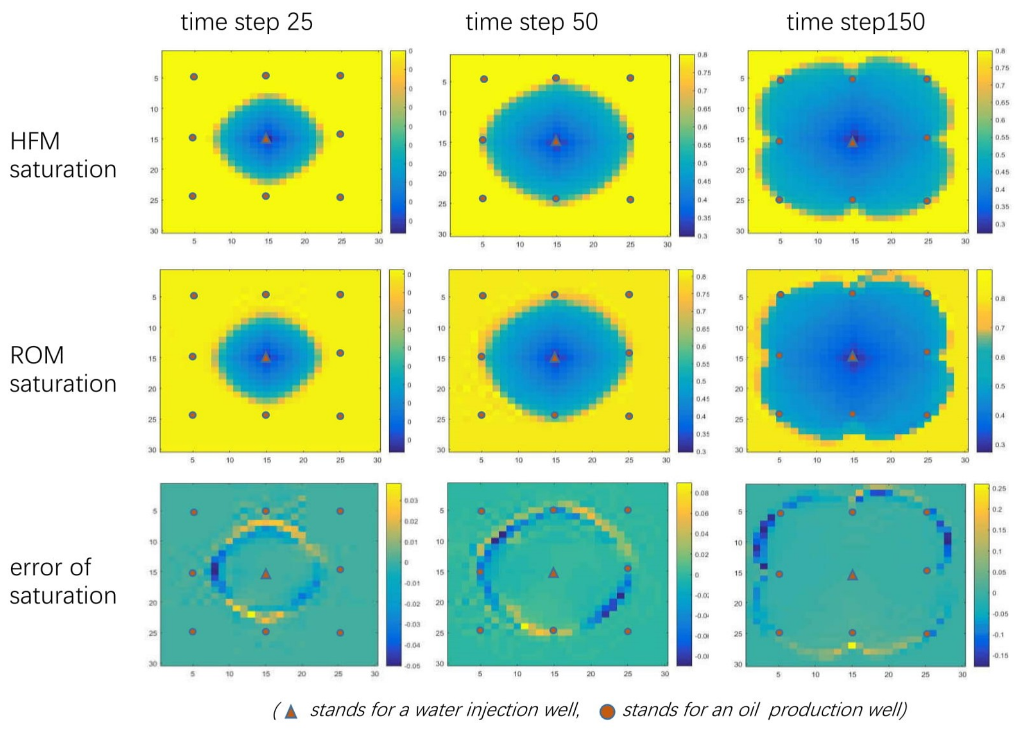

FIGURE 4.5: The saturation value of HFM and ROM and the saturation error at permeability condition U3

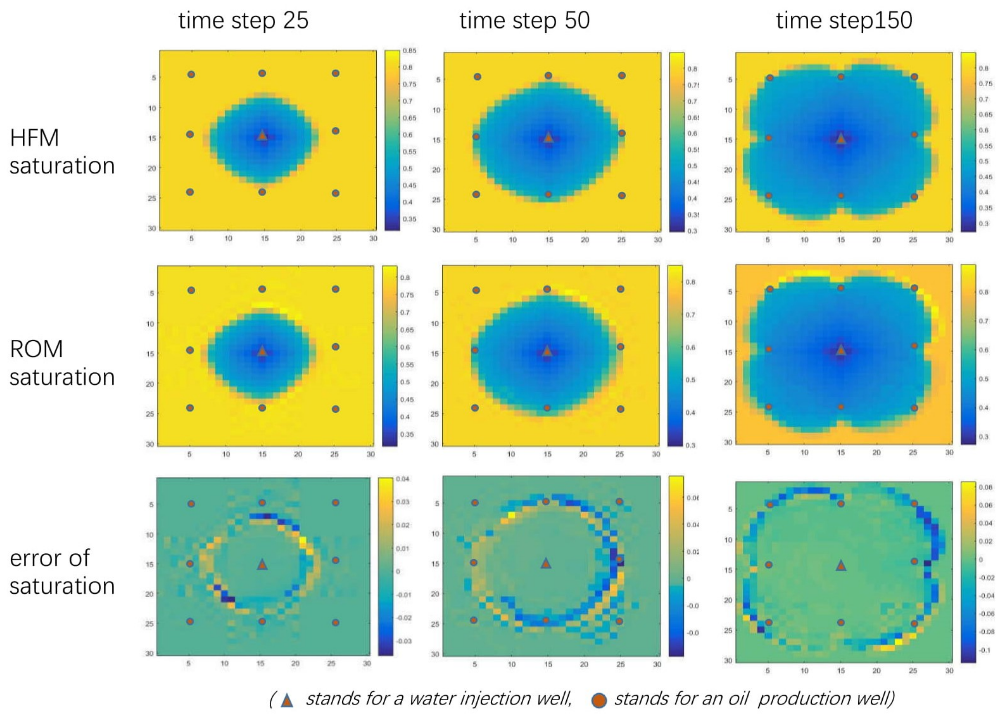

FIGURE 4.6: The saturation value of HFM and ROM and the saturation error at permeability condition U3

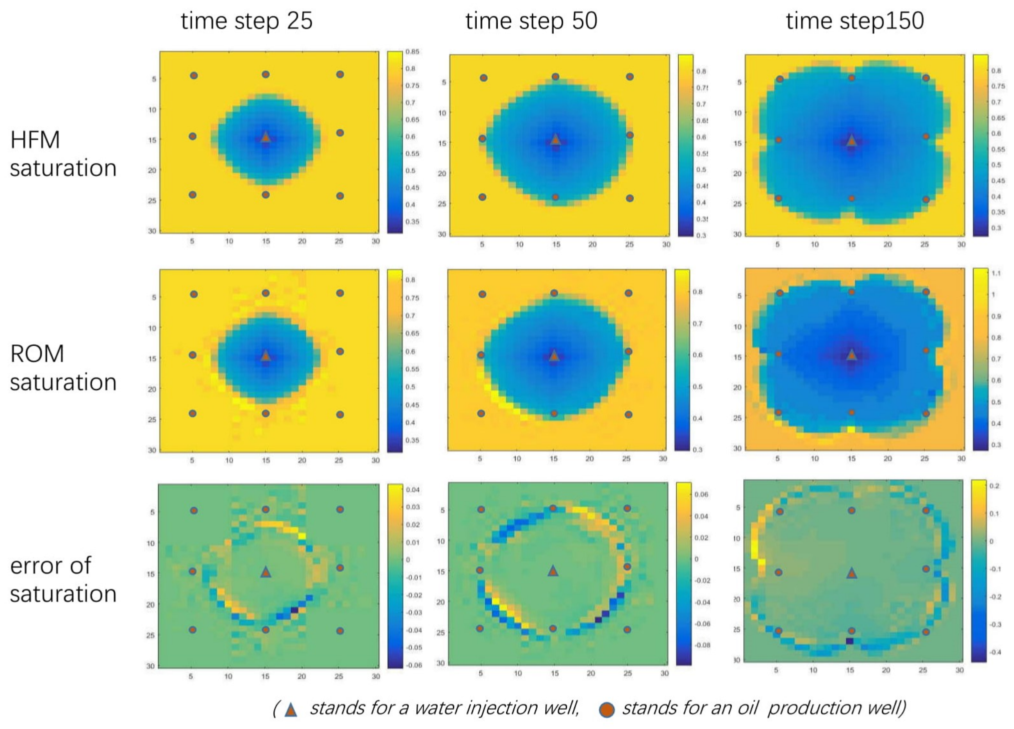

FIGURE 4.7: The saturation value of HFM and ROM and the saturation error at permeability condition U3

4.2 Error Prediction

The following section shows the test result after training the original input and output data, and the test result after transforming data with trigonometric function $f(X) = arctan(X)$. X is the input and output data of training and test.

The quality of test results are measured by cross-validated mean square error (MSE) and the linear regression between the test data and the predicted data.

4.2.1 Error Prediction Based on LASSO

The regularized least-squares regression using LASSO requires all samples into the experiment, of which 900*200*500 becomes the training set and the rest form the test set. The goal is to predict future time steps' value based on the previous time steps, and all permeability conditions are taken into training. The following shows the MSE and the coefficient fit process after training the data:

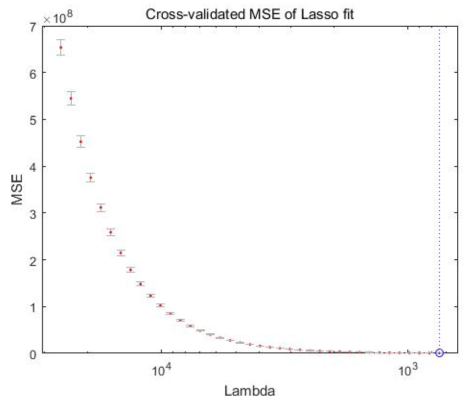

FIGURE 4.8: Cross-validated Mean square error after LASSO regression of pressure data

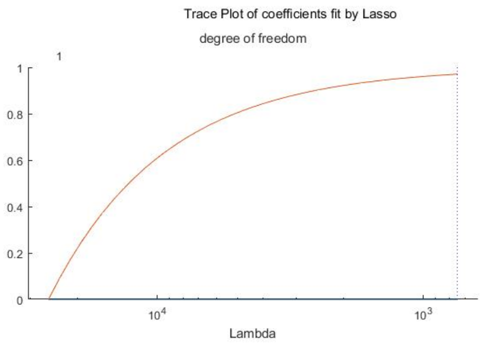

FIGURE 4.9: Trace of the fitted coefficient that corresponds to a Lambda set for pressure data

Here lambda is the penalty coefficient in

$$\min \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_0 \right\} \tag{4.6}$$

.

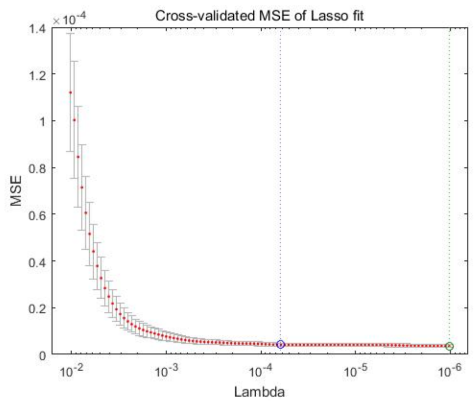

FIGURE 4.10: Cross-validated mean square error after LASSO regression of saturation data

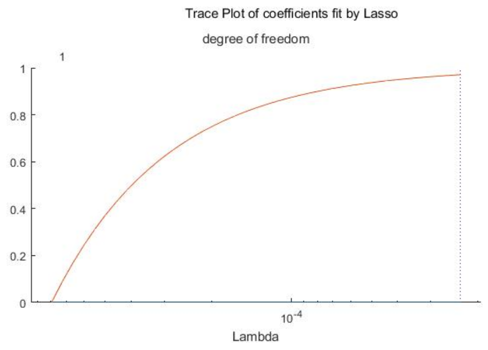

FIGURE 4.11: Trace of the fitted coefficient that corresponds to a Lambda set for saturation data

4.2.2 Error Prediction Based on LSTM

Based on the construction of LSTM network, we would train the data of one permeability at a time. This means the input and output data contain 900 geographic information at each time step for exactly 199 time steps. Its outstanding ability of predicting the next time step is just as obvious as its limitation: it needs a lot of independent training based on different permeability conditions (u) to learn the commonality of data under different u sets.

The loss function is defined as the quadratic loss function. If the target is T, then a quadratic loss function is

$$\lambda(x) = C(T-x)^2 \ \lambda(x) = C(T-x)^2 \tag{4.7}$$

for some constant C; the value of the constant makes no difference to a decision, and can be ignored by setting it equal to 1. This loss function can enhance the weight of extreme values, and this kind of values appear a lot in the early time steps.

The training set occupies 70 percent of the data, as the following 15 percent set to be validation and the rest as test group. There are 50 neurons per hidden layer and at each round one time steps of data will be sent for correlation. The optimization method is conjugate gradient descent. Although stochastic gradient descent is most popular among all the converging approaches in NN training, conjugate gradient descent technique is more suitable in this error model. The reason is that, owing to the highly simplification of the governing equation, the original model shows weak nonlinearity, as the ROM is a linear case, the results in the weak nonlinearity of the error data. What's more, the error data shows high convexity and sparsity, because the ROM has an accuracy of over 90 percent (in general), and the error mainly occurs around the wells, see Fig.4.7. This is a favourable case for conjugate gradient method. Besides, in the subdomain POD-RBF process, each subdomain is trained with its direct surroundings, this means its elements are closely related with those of its neighbour domains, but unrelated with the further subdomains. A major shortage of using conjugate gradient descent for machine learning problems is that, the data is often non-convex and it always converges to a local optimal solution, but under this circumstance, this problem is unlikely to happen. Experiments also proved that conjugate gradient descent has achieved fast and accurate convergence.

The rest hyperparameters are not the main focus of this thesis, so the details are given directly in the Tab.4.2:

TABLE 4.2: The hyperparameters in LSTM structure and experiment

Description	**Value**
Size of samples	900
Number of samples	199
Training set	70%
Validation set	15%
Test set	15%
Batch size	900,1800
Loss function	Mean squared error
Training algorithm	conjugate gradient
Number of layers	2
Neurons per layer	900
Dropout rate in input	0
Dropout rate in recurrent connections	20%
Dropout rate in output	20%
Input time intervals	1,2

And the training results are presented in the form of least MSE and linear regression, as shown in Tab.4.3 and Tab.4.4.

TABLE 4.3: The result in LSTM training of pressure

Input intervals	**1**	**2**
MSE in U1	52441260510.3609	73699422417.9125
MSE in U2	61072675774.1735	107206004385.5059
Regression of all in U1	0.81441	0.50721
Regression of all in U2	0.82974	0.61893

In order to give an intuitive idea of the training process and the result of not only global data set but also the training, validation and test sets separately, here are some examples. In the regression figures, values R measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship. The horizontal coordinate stands for the target values, and the vertical coordinate stands for the predicted values. In the MSE figures, the horizontal coordinate is the epoch number, and the vertical coordinate shows the converge process of MSE.

TABLE 4.4: The result in LSTM training of saturation

Input intervals	1	2
MSE in U1	0.00074212	0.00104291
MSE in U2	0.00092031	0.00119702
Regression of all in U1	0.33167	0.30075
Regression of all in U2	0.44884	0.35771

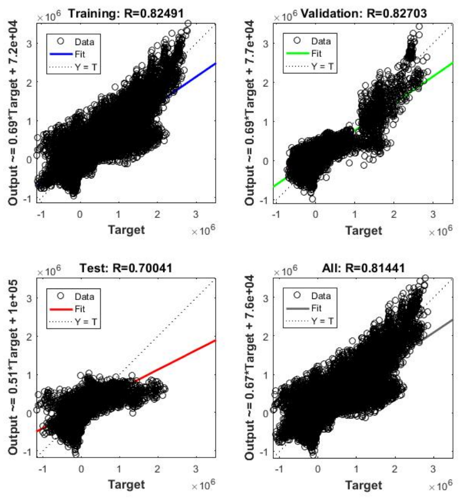

FIGURE 4.12: Regression of LSTM predicted pressure data at permeability condition U1, input interval=1

The training set of LSTM NN has 200 samples of 900 elements, this is a small size in NN training, and for the 2 layer case, could barely cause over-fitting. As the above results demonstrated, when the time interval equals to 1, it gives better prediction than when it is 2. The reason could be that: on one hand, the training is under-fitting, that means when the time interval is 2, there are 500 iterations for one epoch, only half times of when the time interval is 1, and it is not enough to learn the data; on the other hand, the time dependence of data maybe not strong enough, so the influence of under-fitting cannot be countervailed by importing more time intervals in one batch.

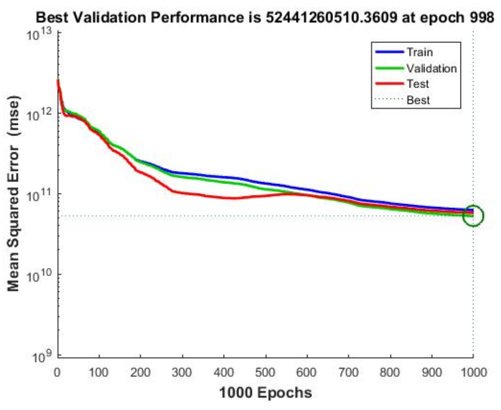

FIGURE 4.13: MSE of LSTM predicted pressure data at permeability condition U1, input interval=1

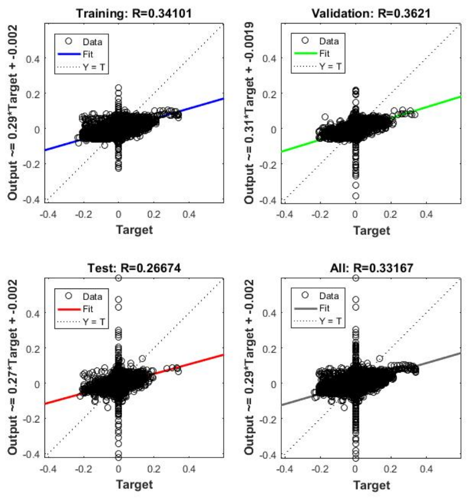

FIGURE 4.14: Regression of LSTM predicted saturation data at permeability condition U1, input interval=1

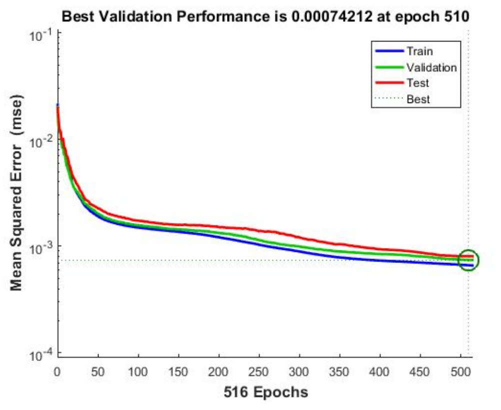

FIGURE 4.15: MSE of LSTM predicted saturation data at permeability condition U1, input interval=1

4.2.3 Error Prediction Based on RNN

The following are the predictions made by a tradition RNN and its full-connected case. The full-connected NN is represented as multi-layer perceptron (MLP). It predicts the error model at a fixed timing when various of permeability conditions occur. The ideal error model would be able to predict the error at both new time step and of new permeability, but in this thesis, this cannot be achieved at the same time, hence the following models would focus on the prediction of new permeability conditions. Again, The training set takes 70 percent of the data, 15 percent are set to be validation and the rest as test group. The training method is conjugate gradient with a quadratic loss function. The following table presents all the details in the structure and experiment.

TABLE 4.5: The hyperparameters in RNN structure and experiment

Description	**Value**
Size of samples	900
Number of samples	1000
Training set	70%
Validation set	15%
Test set	15%
Batch size	900
Loss function	Mean squared error
Training algorithm	conjugate gradient
Number of layers	2
Neurons per layer	500
Dropout rate in input	0
Dropout rate in recurrent connections	1 or 0
Dropout rate in output	0
Input series intervals	1
Time step	25, 50, 150

When the forget rate is 1, all previous information is lost. And when it is 0, the system becomes fully connected, and all the previous parameters are preserved. In this case, a RNN of same input and output sizes is no different from a fully connected NN of multi perceptron layer, and we call it multilayer perceptron or MLP in the following content as a distinguish. In Tab.4.6 and Tab.4.7 we present the linear regression and least MSE of the test results. The time steps are indicated as T1, T2, T3.

TABLE 4.6: The result in RNN and MLP training of pressure

Dropout rate	**1**	**0**
MSE in T1	58354850271.3609	46652505633.6624
MSE in T2	51016490007.1735	35288804315.8402
MSE in T3	50374629649.1735	19128638465.2276
Regression of all in T1	0.62338	0.78613
Regression of all in T2	0.69192	0.81177
Regression of all in T3	0.69263	0.97133

In order to give an intuitive idea of the training process and the result of training, validation and test data sets separately, Fig.4.16, Fig.4.17 Fig.4.18 and Fig.4.19

TABLE 4.7: The result in RNN and MLP training of saturation

Input intervals	**1**	**2**
MSE in T1	0.000013925	$1.6079*10^{-5}$
MSE in T2	0.000027604	0.000022741
MSE in T3	0.000826487	0.000323765
Regression of all in T1	0.48262	0.73444
Regression of all in T2	0.53528	0.78362
Regression of all in T3	0.58391	0.83477

are some examples. In the regression figures, R values measure the correlation between outputs and targets. When R value is 1, it means a close relationship, while 0 means a random relationship. The horizontal coordinate stands for the target values, and the vertical coordinate stands for the predicted values. In the MSE figures, the horizontal coordinate is the epoch number, and the vertical coordinate shows the converge process of MSE.:

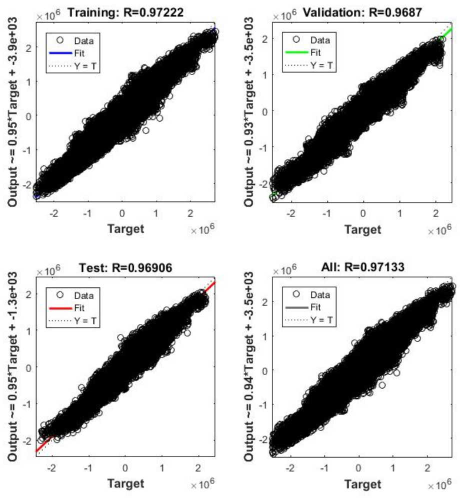

FIGURE 4.16: Regression of MLP predicted pressure data at time step 150

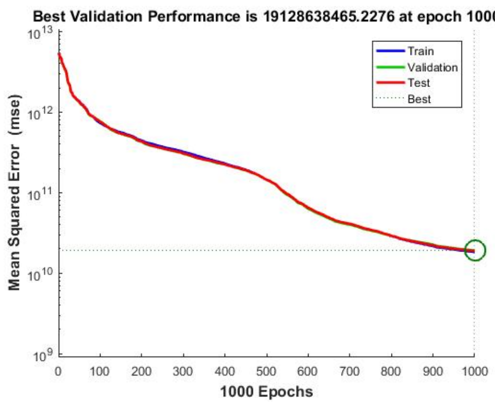

FIGURE 4.17: MSE of MLP predicted pressure data at time step 150

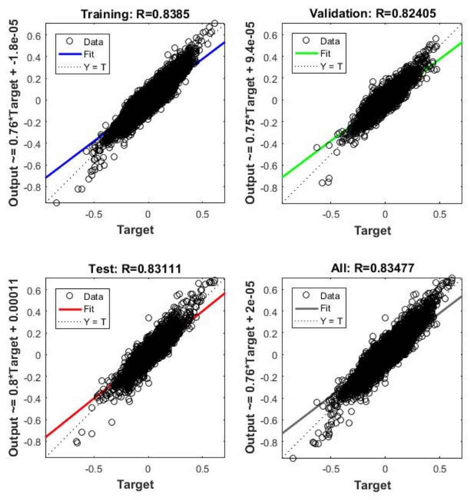

FIGURE 4.18: Regression of MLP predicted pressure data at time step 150

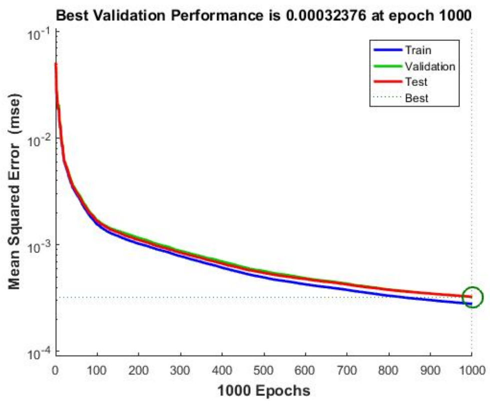

FIGURE 4.19: MSE of MLP predicted saturation data at time step 150

4.3 Performance Evaluation

The results in the previous section 4.2 have shown that, LASSO learns to reproduce the output with a fast convergence, which should not occur in real industry, due to the non-linearity of the difference between a nonlinear HFM and a linear ROM. The only good explaination is that, the HFM data itself shows high linearity, and this could be the result of the highly simplified model.

As for the machine learning techniques, it turns out there is a huge disparity between pressure and saturation. In both methods, pressure always gives better result than saturation, and it even reaches a maximum of 97 percent correlation with the test data. After analysing the original error data, we have noticed that this is because of the different error location and distribution. The pressure is measured in the production well, and its errors locate on most grids of the field through out all time steps. The saturation is measured in the water injection well, which is the single well in the centre (in this case). According to observation, the saturation error takes longer time to 'spread' to the edge of the field (see Fig.4.7), leaving many grids with zero error. The large amount of zeros affect the machine learning training process by misleading it to neglect the larger values, and assigning too much weight to the blank parts. Besides, the saturation values have a small standard deviation, and 68.3 percent of the values are distributed in the range of one standard deviation, while 95 percent of them are distributed with two times standard deviation. Under this circumstance, the values within two time's standard deviation are given much more weight than the out ranged ones. This also causes the penalty of some rare extreme values.

Experiments on multiple neural network parameters have also been conducted. It is surprising that for LSTM training, the result has better accuracy when the time interval is one, compared to the situation when it is two. This may suggest that the output values are time independent, or the sample number is so small that the influence of increasing the epoch number is larger than the influence of enlarging the epochs.

And by comparing all three methods, we are pity to find that only LASSO can predict the error based on different permeability conditions and timings, but it only functions for the linear cases and weak nonlinear cases, and the practical cases are often highly nonlinear. LSTM predicts the future error of one given permeability condition based on the trained samples of previous time step under the same permeability condition. When it meets a new permeability condition, its coefficients becomes obsolescent and its usefulness becomes limited. The MLP training was performed at one certain time step for many permeability conditions, and it gives good predictions for predicting errors of a new permeability condition. But to observe the changes through out the whole timeline, it requires n times of trainings, n represents the number of the time step. Considering the fact that in a MLP system, the features in a matrix are extracted more effectively and are fully inherited by the next state, more features are taken into training while less information is lost. And since the data set here is not time dependent, the superiority of LSTM is pretty weak. Thus even though MLP does not train the data as a sequence, its prediction accuracy is even better. For now this seems to be the most promising method, and further optimization will be concerned on the MLP training.

4.4 Model Optimization

In this section, several ways of optimizing the MLP model will be presented, including the preprocessing of the data, and the design of loss function and training method. The purpose is to reduce the number of zeros in the saturation error, to reduce the weight of the zero values while increasing the weight of non-zero values, and to reduce the influence of weak correlated values.

4.4.1 Transformation

According to statistic, the saturation values have a small standard deviation, and 68.3 percent of the values are distributed in the range of one standard deviation, while 95 percent of them are distributed with two standard deviation. Under such circumstance, the values within two standard deviation are given much more weight than the out ranged ones. However, it happens to be the out ranged values that matters the most. Therefore, we hope to raise the standard deviation by 'balancing' the values in the data set. A transformation by trigonometric function $f(X) = arctan(X)$ is adopted for preprocessing saturation data, where X represents the training and test data. Through transforming, the difference of near zero values can be enhanced, and the abnormal values are shrinked, leading to a fair weight decay, then the values are more likely to be preserved instead of being penalized. Some examples of the transformed result are given by Fig.4.20, Fig.4.21, Fig.4.22 and Fig.4.23:

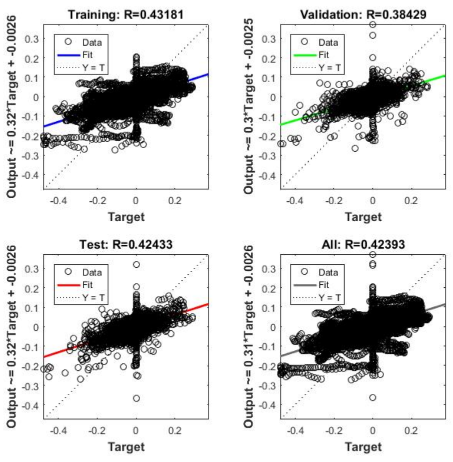

FIGURE 4.20: Regression of LSTM predicted transformed saturation data at permeability condition U2

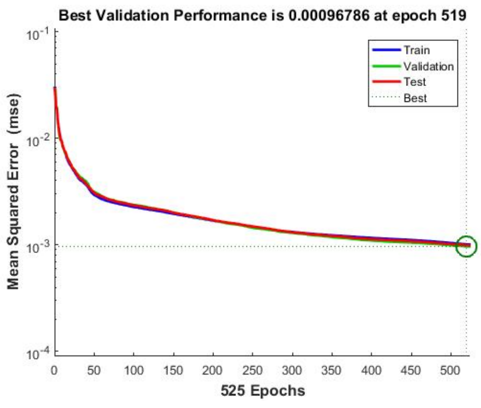

FIGURE 4.21: MSE of LSTM predicted transformed saturation data at permeability condition U2

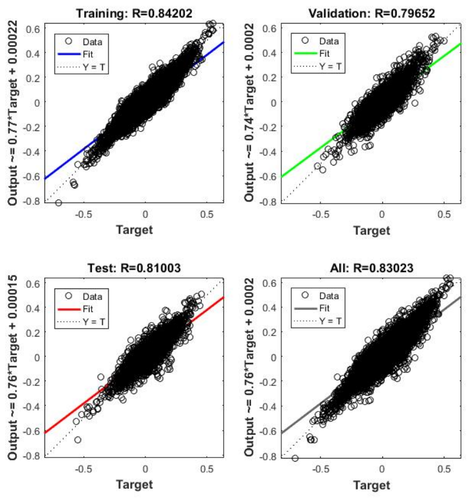

FIGURE 4.22: Regression of MLP predicted transformed pressure data at time step 150

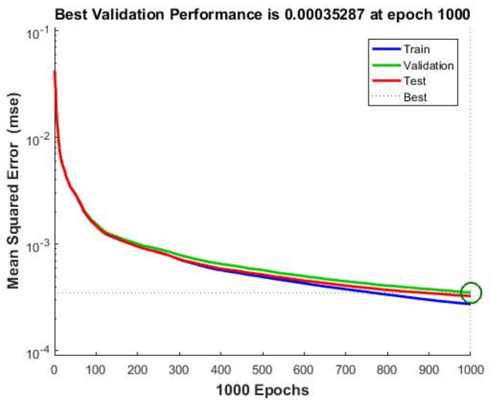

FIGURE 4.23: MSE of MLP predicted transformed pressure data at time step 150

4.4.2 Domain Cropping

According to the results in section 4.2, the MLP outcomes of pressure error has achieved satisfying results, and the remaining issue is to optimize the performance of LSTM technique and the MLP method for saturation error. One obvious method is to crop the domain for early timings of data, and attach the size with the changing speed. This speed is closely related to the permeability condition. Larger permeability means faster water flowing through, and leads to quick changes of the physic model. As a consequence, the error spreads faster in the domain, and resulting in a larger area with bigger values. Therefore, the cropping is operated in line with the data set of the largest permeability. The cropping process operates as follows. According to observation, the saturation error before time step 50 expends quickly, after time step 70 it remains on the same area of the field. So cropping could help for the saturation data before time step 50 in MLP training. Data before this time step is partitioned into five groups with their furthest time steps set to 5, 10, 15, 25 and 45, and each group is cropped into one size. As an example, below shows the process of cropping a saturation data set at time step 25:

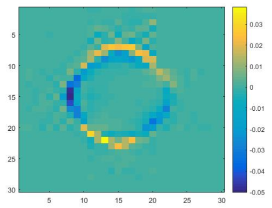

FIGURE 4.24: Saturation error at time step 25

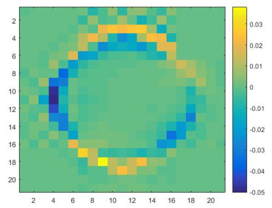

FIGURE 4.25: Cropped saturation error at time step 25

The results after cropping have an obvious improvement, as in Fig.4.26 and Fig.4.27:

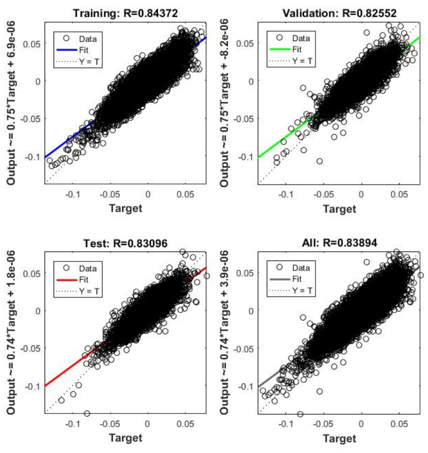

FIGURE 4.26: Regression of cropped MLP predicted saturation data at time step 25

As a comparison, Fig.4.28 and Fig.4.29 present the uncropped results:

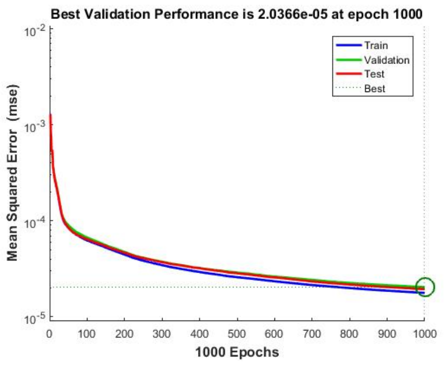

FIGURE 4.27: MSE of cropped MLP predicted saturation data at time step 25

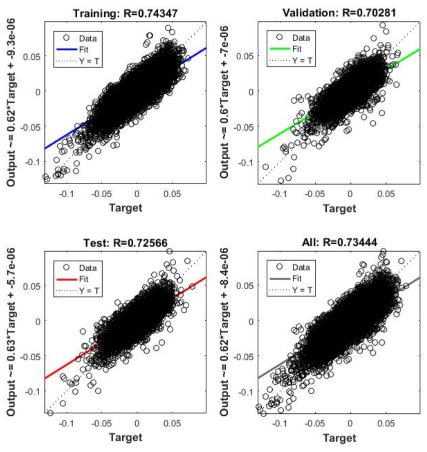

FIGURE 4.28: Regression of MLP predicted saturation data at time step 25

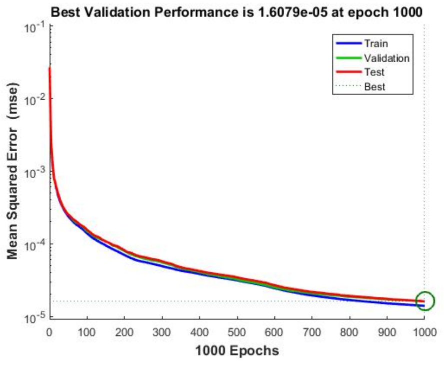

FIGURE 4.29: MSE of MLP predicted saturation data at time step 25

4.4.3 Data clustering

Considering the significant influence of permeability condition of the features of the data, we could apply a data clustering for the early time steps' data, and train each data group independently. This decreases the interaction of unrelated data in the correlation process. Here we operated a k-means clustering based on the different permeability. A major advantage of this method is that, when new permeability conditions are given, they could be assigned into a suitable cluster by calculate the values' distance with the means. The whole data set is divided into three groups. k-means clustering partitions n values into k clusters and each value belongs to the cluster with the closest mean. The process goes as follows, given an initial set of k means $m_1, ..., m_k$, the algorithm proceeds by alternating between two steps:

First step: Allocate each value to the cluster whose mean has the least squared Euclidean distance, this is intuitively the "nearest" mean, see Fig.4.34 and Fig.4.35. This step can be represented by Eq.4.8:

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \,\forall j, 1 \leq j \leq k\} \tag{4.8}$$

where each sample value $x_p$is assigned to exactly one cluster $S^{(t)}$.

Second step: Compute the new means to be the centres of the values in the new clusters with Eq.4.9, see Fig.4.36.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{4.9}$$

The algorithm converges when the previous clusters stop changing, and new samples are then clustered into a cell based on the current parameters, see Fig.4.37.

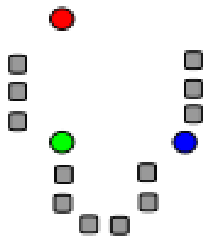

FIGURE 4.30: 1. Initial "means" (in this case $k = 3$) are randomly generated within the data domain (shown in color). Source: Wikipedia

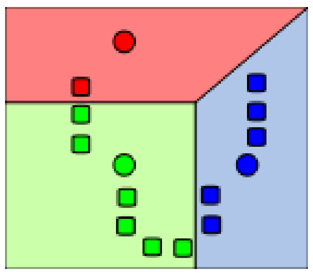

FIGURE 4.31: 2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means. Source: Wikipedia

The data set is then divided into three groups based on the permeability conditions. Different arrangement of permeability conditions may lead to different clustering terms. As shown in Tab.4.8 is an example of it at time step 50.

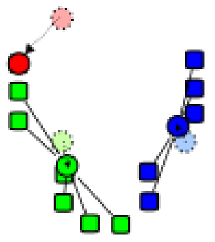
FIGURE 4.32: 3. The centroid of each of the k clusters becomes the new mean. Source: Wikipedia

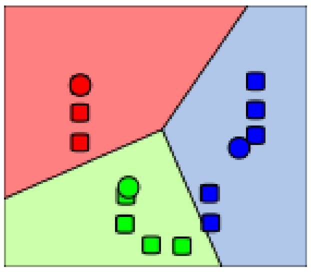
FIGURE 4.33: 4. Steps 2 and 3 are repeated until convergence has been reached. Source: Wikipedia

TABLE 4.8: The groups at time step 50

Groups	number of values
1	508
2	289
3	203

This improves the training results on the training result, compared with a group of unclustered data of the same size. For the current scale of data, there is little probability of data overfitting, thus the larger groups give better results than the small groups, as the statistical methods deeply rely on the amount of its samples. However, the small groups are also able to yield satisfying result given enough training samples. As shown in Fig.4.34,Fig.4.35 and Fig.4.36, Fig.4.37 is a comparison between the training of group one by k-means clustering and of a randomly clustered group of the same size:

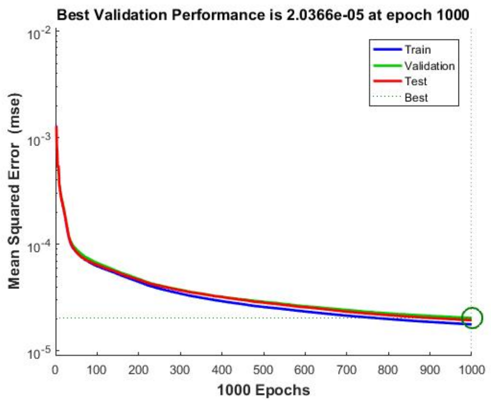

FIGURE 4.34: MSE of MLP predicted saturation data at time step 50 in group 1

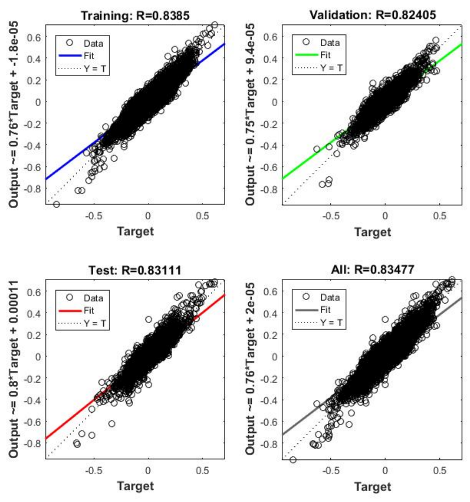

FIGURE 4.35: Regression of MLP predicted saturation data at time step 50 in group 1

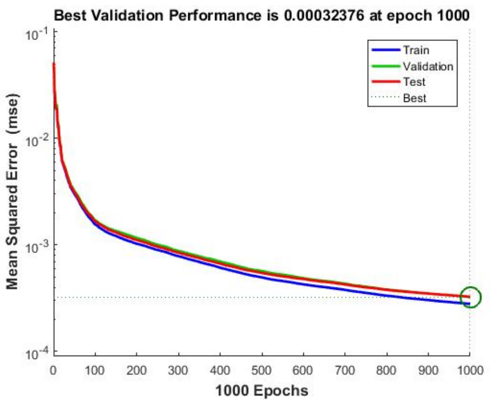

FIGURE 4.36: MSE of MLP predicted saturation data at time step 50 in a random group

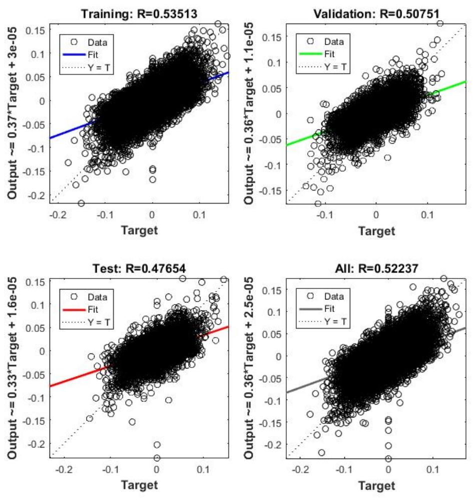

FIGURE 4.37: Regression of MLP predicted saturation data at time step 50 in random group

Chapter 5

Case Study: Applying Error Models to SAIGUP benchmark case

In this chapter, an application of the learned error model to a reservoir problem will be presented. This example uses a reservoir model with 13 wells based on the SAIGUP benchmark case. Then the predicted error will be added to the ROM, and a parameter estimation will be performed to estimate the accuracy of the ROM.

5.1 Application to Benchmark SAIGUP Model

We demonstrate the performance of LP-SD POD-RBF, through a modified version of the SAIGUP benchmark case study containing 13 wells. In what follows, we consider two history matching scenarios that involve: sparse well production data, e.g., fluid rate, watercut and bottom hole pressure (BHP), is assimilated; and both well production data and seismic data, e.g., water saturation in each grid block, are simultaneously assimilated for calibrating the SAIGUP model with a large amount of uncertain parameters. Again, MRST is used to run the full-order model simulations.

5.1.1 Description of model settings

The first layer of SAIGUP model contains 3895 active grid cells and 905 inactive grid cell. The realistic geological properties, such as faults, are preserved. The reservoir model describes a waterflooding system in which six production wells and seven injection wells are placed and labeled from P_1 to P_6, and I_1 to I_7, they are almost uniformly located in this reservoir, as shown in Fig.5.1. The details of this system is described in Tab.5.1.

5.1.2 Description of reduced model procedure

We manually generate 1000 Gaussian-distributed log-permeability. These 1000 log-permeability fields are used to form the global parameter covariance matric and local parameter covariance matrices for each subdomain. One of these realizations is considered to be the truth. In this numerical experiments, 95% global principal components analysis (PCA) patterns are retained for global parameterization which result in N_G=48 global PCA patterns. The log-permeability fields are also locally parameterized with retaining 95% local PCA patterns in each subdomain. The total number of local PCA coefficients $N_L = \sum_{d=1}^{S} l_d$=275. The maximum number of local PCA coefficients among all subdomains is 15. We divide the entire domain into 20

TABLE 5.1: Experiment settings using MRST

Description	Value
Dimension	40×120×1
Number of wells	6 producers, 7 injectors
Fluid density	1014 kg/m^3, 859 kg/m^3
Fluid viscosity	0.4 mP·s, 2 mP·s
Initial pressure	25 MPa
Initial saturation	S_o=0.80, S_w=0.20
Connate water saturation	S_{wc}=0.20
Residual oil saturation	S_{or}=0.20
Corey exponent, oil	4.0
Corey exponent, water	4.0
Injection rate	200m^3/d
BHP	20MPa
History production time	10 year
Prediction time	15 year
Timestep	0.1 year
Measurement timestep	0.2 year

rectangle subdomains (4 subdomains in x direction and 5 subdomains in y direction) which is also considered to be the base-case.

Finally, 2200 snapshots are collected for pressure and saturation separately. For each subdomain, two separate eigenvalue problems for pressure and saturation are solved using POD.

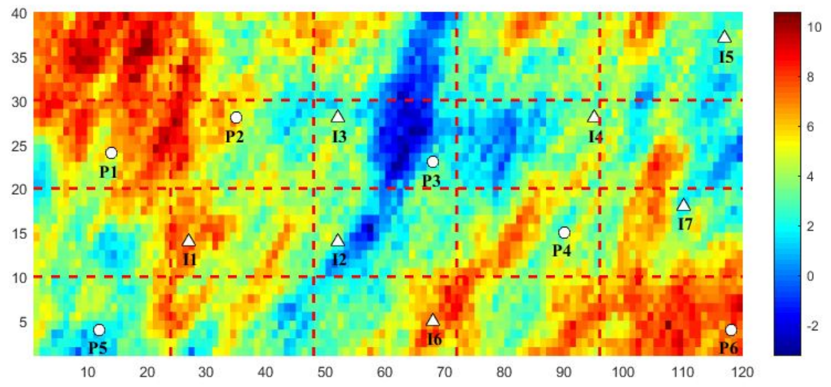

FIGURE 5.1: The illustration of domain decomposition in the 2-D synthetic model for scenario S1

5.2 Error Modelling for the ROM of Benchmark SAIGUP Model

In the error modeling, pressure, saturation and permeability matrices are normalized and combined into one. The normalized pressure

$$Pnorm = \frac{P - P_m in}{P_m ax - P}$$

,
log-permeability

$$Unorm = \frac{U - U_m in}{U_m ax - U}$$

and saturation

$$Snorm = \frac{arctan(S) - arctan(S_m in)}{arctan(S_m ax) - arctan(S)}$$

The first error model is based on LASSO. The number of samples is 4800*100*1000, and the training result looks rather promising:

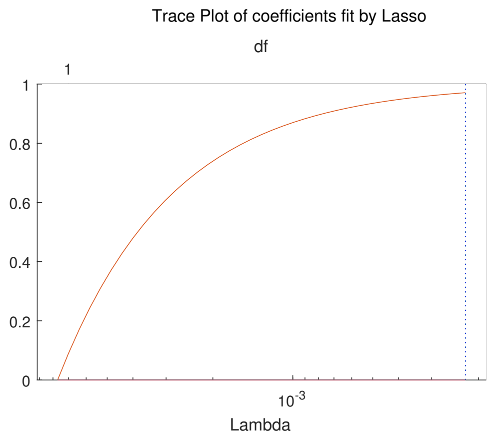

FIGURE 5.2: The fitting process of LASSO model

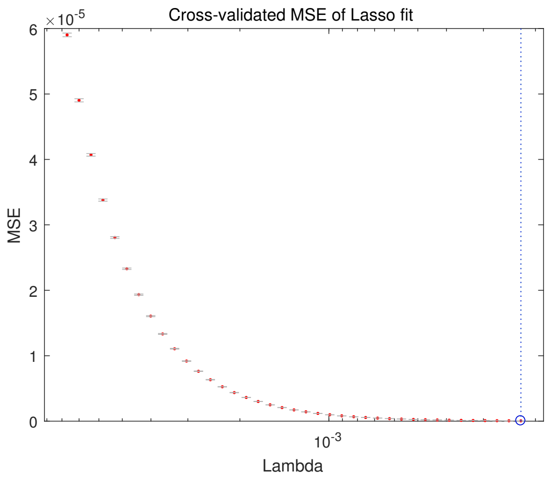

FIGURE 5.3: The MSE of LASSO model

The success of LASSO training indicates the weak nonlinearity of this case . The statistical results shows that the error distribution follows similar pattern as previous case. Therefore, the following error model inherits most hyperparameters as the MLP NN in Chapter 4, and is based on an optimized two layer MLP system.

The hyperparameters are selected as follows in Tab.5.2:

TABLE 5.2: The hyperparameters in the error model

Description	Value
Size of samples	2160 or 2880 or 3600
Number of samples	1000
Training set	70%
Validation set	15%
Test set	15%
Loss function	Mean squared error
Training algorithm	conjugate gradient
Number of layers	2
Neurons per layer	1000
Time step	25 50 90

The size of this field is larger than the previous case, leading to a huge computing time. Therefore, the global domain is divided into 20 subdomains, each of which is of the size 10*24 pixels. The subdomain's size is the same as that in the domain decomposition procedure during the POD-RBF order reduction. Since the ROM training for each subdomain was based on itself and its 4 neighbour subdomains (on the corners it is trained with two neighbours, and on the edges it is trained with 3 neighbours), the data of the central subdomain is closely related with its neighbours, so is the error of the ventral subdomain. Thus the error model will follow the same operation as in the MOR, taking five subdomains into the training for the input data, and one subdomain for the output data, as shown in Fig.5.4.

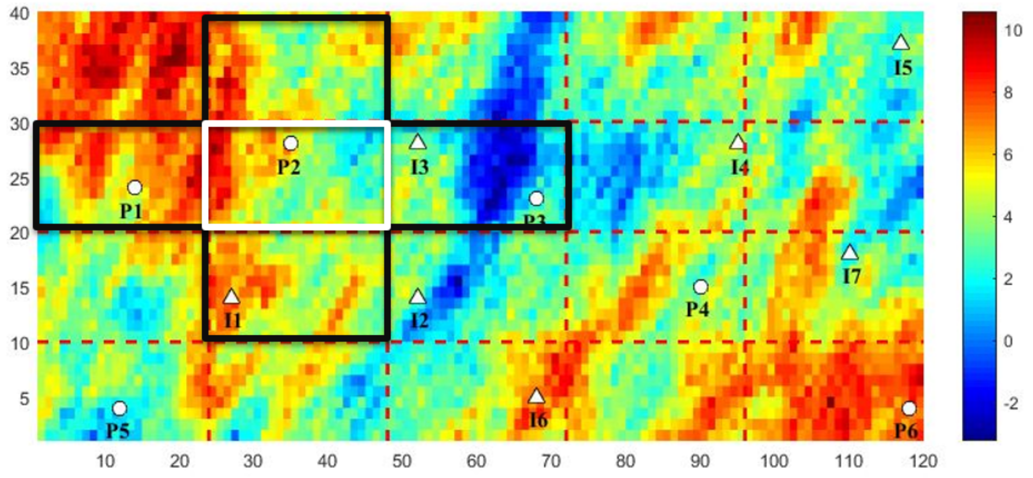

FIGURE 5.4: The domain decomposition in the 2-D synthetic model for scenario S1, the black lines mark the neighbour subdomains and the white lines mark the central subdomain

Without loss of generality, we select the marked subdomain in Fig.5.4 to illustrate the training results of the linear regression and least MSE of the test result. The regression values R measure the correlation between outputs and targets. An R value approaches to 1 means a close relationship, and to 0 means a random relationship. Finally the data has been denormalized to the original scale.

TABLE 5.3: The training result of the MLP error model

Data type	pressure	saturation
Least MSE in T1=25	1878402649	$1.5738*10^{-5}$
Least MSE in T2=50	1599857444	$1.0006*10^{-5}$
Least MSE in T3=90	1962749974	$1.7559*10^{-5}$
Regression of all in T1=25	0.97266	0.92758
Regression of all in T2=50	0.958497	0.97352
Regression of all in T3=90	0.996338	0.97613

The linear correlation between the target and prediction is very high, indicating successful experiment. This is the result from the domain decomposition and separate training of subdomains. For smaller domains, the features are easier to be more completely captured; and for each subdomain, only related neighbour elements are taken in for training, this excludes the negative impact of unrelated data. What's more, the well locations in this case is not as special as the case in Chapter 4, and there is no large area error offset, so the effect of the sparsity is weaker than the previous case.

In order to observe the result, the predicted error matrices are added onto the reduced order matrices as a strengthening, and then compared with the full order model by parameter estimation. Taking time step 90 and permeability condition 995 (the permeability conditions are created randomly, hence this numbering is meaningless) as an example, we present the full order models, the reduced order models and the fortified ROMs of pressure and saturation in Fig.5.5 to Fig.5.10.

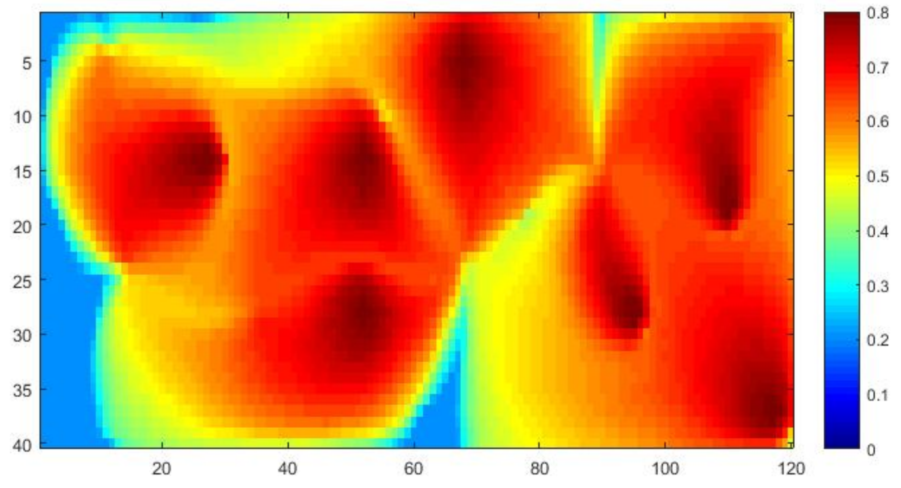

FIGURE 5.5: The saturation HFM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

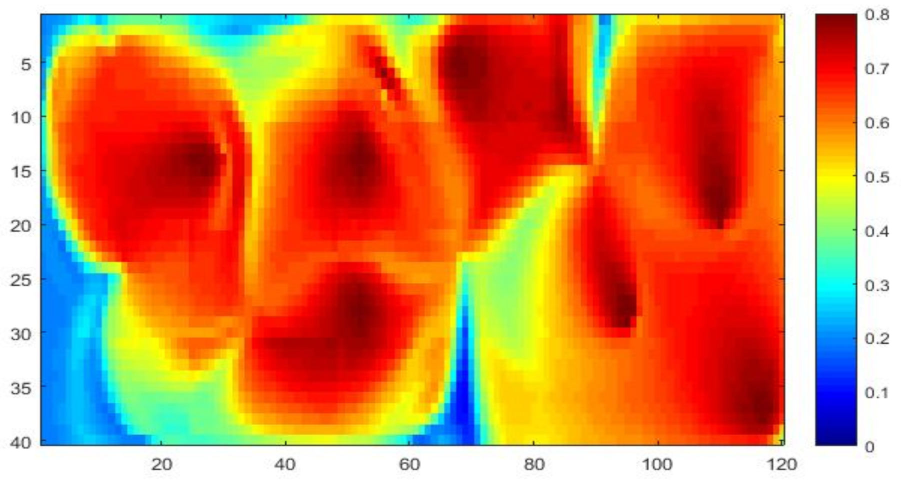

FIGURE 5.6: The saturation ROM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

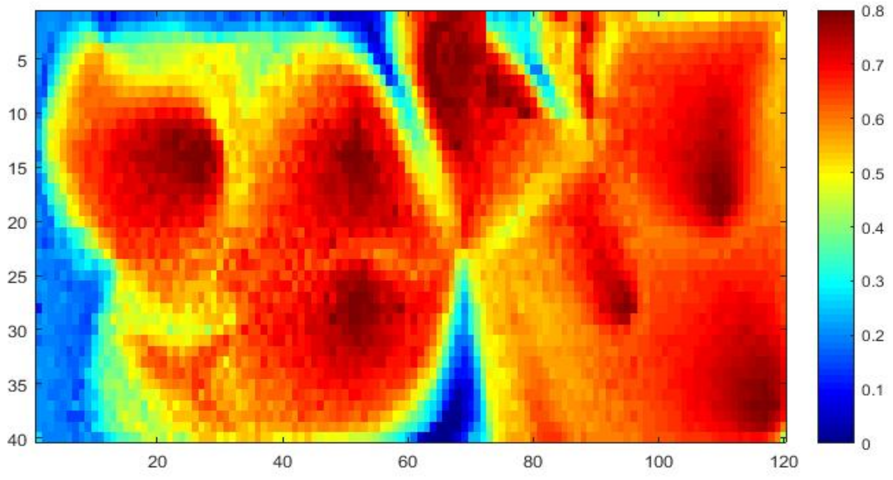

FIGURE 5.7: The fortified saturation ROM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

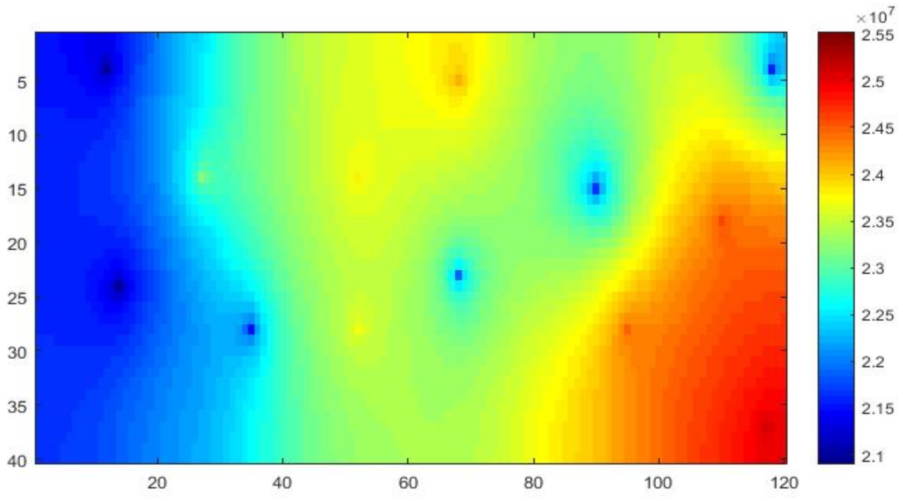

FIGURE 5.8: The pressure HFM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

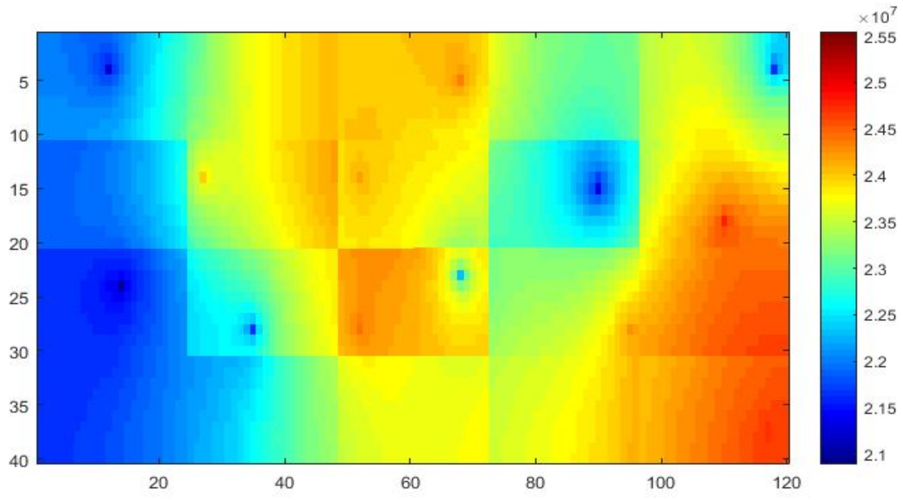

FIGURE 5.9: The pressure ROM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

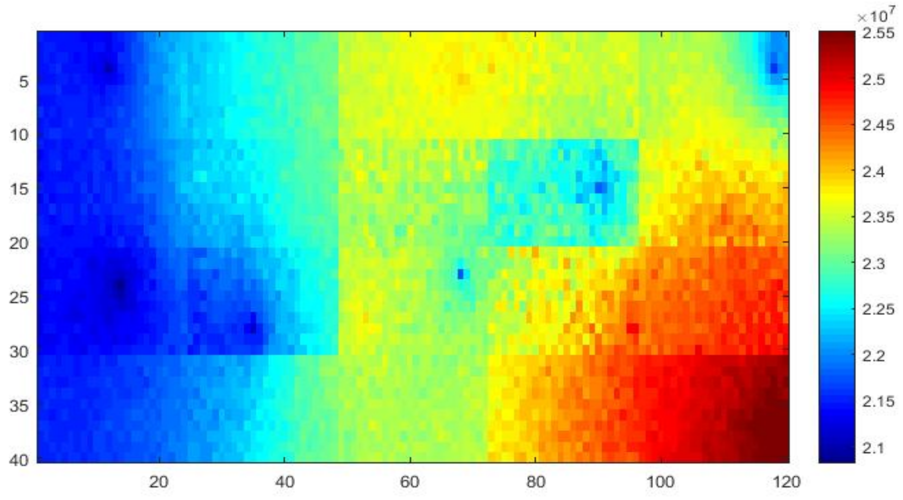

FIGURE 5.10: The fortified pressure ROM in the 2-D synthetic model for scenario S1 at time step 90 in the permeability case 995

Next, a parameter estimation through least square method is adapted to estimate the accuracy of the error model. In the function of least square method

$$min\{J = 1/2\sum_{j=1}^{n}(x_i - y_i)^2\} \tag{5.1}$$

x_i is the observation at step i, it is obtained by adding a normally distributed noise to the HFM model. y_i=v_i, v_i symbols the ROM model. Or y_i is the fortified ROM at step i, y_i=$v_i + e$, e stands for the error model at this step. As shown in Tab.5.4 is an example of the parameters of the same model as the previous example:

TABLE 5.4: Polynomial parameters

Description	**Value**	
Model	ROM	Fortified ROM
Case pressure x_i^0	$1.9 * 10^6$	$1.1 * 10^6$
Case pressure x_i^1	0.88	0.92
Case pressure regression	0.86286	0.9096
Case saturation x_i^0	0.02	0.013
Case saturation x_i^1	0.96	0.97
Case saturation regression	0.86956	0.92068

The parameter estimation has shown that, the fortified ROM is more closely related to the observation data, compared with the original ROM, which has proven the usability of the error model in this case. More comparisons are made in the next section.

5.3 Comparison and Discussion

There are several conclusions that can be conjectured. One is that subdomain MLP training would be more precise than the rest methods in our experiments. It is stronger on collecting features, and less effected by unrelated data. Compared with RNN, its weakness is on processing sequences such as time series. However, although the input data contains the phenomenon of time evolution, the permeability-dependent changings we attempt to predict is not a time varying problem, thus MLP is appropriate and sufficient here. For the two layer MLP, more neurons could directly lead to better result, from this perspective we would select as many neurons as possible. However, in deeper NNs, this may lead to overfitting, thus a careful selection of hyperparameter would be required, but this is not the main focus of our work. The application case study with the bechmark SAIGUP model in this is still highly simplified and weak nonlinear, so both the LASSO error model and MLP error model could achieve good result, and the hyperparameter can be inherit from the case in Chapter 4.

However, MLP is a little out of date now, and people have put more focus on Convolutional neural network (CNN) for image training problems. This works as classification, it takes an input image, process it and classify it under certain categories. It can be combined with RNN so there will be both outstanding ability to recognize the features and to learn the connection in a series to form stronger prediction. Previous work has also presented a random forest classification combined with LASSO for regression[6]and achieved significant progress. An interesting direction to continue the work in this thesis would be combining MLP with RNN.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has developed and compared the error models for surrogate model in oil reservoir simulation based on LASSO, LSTM, traditional RNN and fully connected NN.

The full order model is built by MRST toolbox, and the reduced order model is built by a subdomain POD-RBF algorithm. The quantities of interest are saturation and pressure, while the controls are time and permeability. The objective is to predict the saturation and pressure of the next time step under a new permeability condition, so both controls have to be considered. Permeability values are distributed on every grid of the field, and keep as constants through out the time. The prediction bases on the observation from previous time step, therefore the input of the training is the saturation or the pressure from time step n, and the target is the saturation or the pressure of time step $n + 1$.

The very first procedure is to preprocess the data. Having collected the HFM and ROM data, we calculate the error by making differences of the HFM and ROM values. Next, the pressure values and the log-permeability values are normallized, and saturation values are transformed and normallized for better training performance. Here, the saturation values are processed by an arctan function additionally, because statistics have shown that saturation values are distributed in a dense Gaussian distribution area, which means that 95% of the values are within two times standard deviation, while most of them are approximate 0. But on the few well locations are their surroundings, the error can be extremely large in this scale. In the process of training, it is probable that these rare larger values are penalized and the small values are overlooked in the current scale. Therefore, we added a trigonometric function to balance the values.

For LASSO and LSTM systems, the data is ready for training after above procedure. But for RNN and MLP training, there are more options to further process the data. As RNN and MLP training takes in the values of all permeability conditions at one time step for each training, and the data show obvious patterns with the ongoing time steps, we could adjust the data according to its current time step. It is not hard to find that the errors always start from the well location, and spread generally to fully fill the whole field. This is because in the MOR procedure, the next step data comes from the previous iterations, and the error accumulates as the number of the iteration increases, until it reaches the largest at the last time step.

However, the errors are rather small in the early time steps, leaving large areas in blank. One obvious action would be cutting off these blanks. The explicit area and shape depends on the exact time step, and can be universalized in several different time sections. Permeability also have unneglectable influence on the spreading speed of saturation error. Physically, permeability means the ability of a material (such as rocks) to transmit fluids, the larger it is, the faster fluids travel. Since saturation relates closely to the oil and water travel speeds, it is highly affected by local permeability. Therefore, a natural idea is to 'cut off' the unrelated permeability conditions from the training set. This can be operated by clustering the data into multiple groups based on the permeability values. The permeability values on one field is randomly generated according to Gaussian distribution. For this sake, a k-means clustering is adapted on the permeability values. A suggested number of groups for 1000 samples is 3. Although precise division could better describe the permeability's feature, the size of the training set should be large enough for better training, thus the number of clusters should not be to large. It is often the case in statistical methods that the sample number and the precision are linearly related when it is not overfitted.

The second procedure is to build error models, while the first one is based on LASSO. For both experiment and application cases, LASSO has achieved surprisingly nice results by reducing the mean square error of the test set to 10^{-5}. As a linear regression, the explanation for such result would be that the error values are showing weak non-linearity, and since LASSO training takes the whole data set into training at once, its number of parameter is the largest. The error is defined as the difference between HFM and ROM data, and ROM data is linear. Through a simple inference, we would discover that the HFM values show weak non-linearity. And the origin for this phenomenon is likely to be the highly simplification of the governing equation.

The second training method is LSTM, a type of RNN that keeps short term memory for long time. During the training, samples of continuous time steps under one permeability condition are taken into one batch. The results are quite mediocre, partly because the sample number is as small as less than 200, also because in our case RNN feed forward is doing a one to one mapping for vectors, and when there are dropouts, previous feature information is lost. What's more, as the previous results presented, when the time interval equals to 1, it gives better prediction than when it is 2. When the time interval is 2, there are 500 iterations for one epoch, only half times of when the time interval is 1, and the time dependence of data is not so strong as to ignore the influence of under-fitting. The best linear regression of test data in LSTM has reached 0.82974 for pressure, and 0.61893 for saturation, this is far less than that in MLP. The lost of features and under-fitting could be the main cause.

In our case, feature loss happens to all RNN training with information loss between layers. Therefore, a fully connected MLP is taken as replacement. For this case, we take samples at one time step from different permeability conditions into a training. Through this, the data behaviour of a new permeability at current time step could be predicted. It has achieved the best result among all the other models, because the total inheritance of the previous states is actually necessary for such prediction. In the experiments in Chapter 4, the linear regression between target and prediction data has reached 0.97133 for pressure, and 0.93126 for saturation. In Chapter 5, a domain decomposition is operated on the data set. The whole domain was divided into twenty subdomains, as same as that in the domain decomposition procedure during the POD-RBF order reduction. The error model took five subdomains into the training as the input data, and one central subdomain as the output data. It was first adapted to fasten the computation, but since the domain decomposition eliminated the effect of unrelated data, and the changes in subdomains are easier to capture than the global domain, this helped to achieve high accuracy. During the training of MLP error model, the linear regression between target and prediction has reached up to 0.996338 for saturation, and 0.97613 for pressure. After the fortification for the POD-RBF model, the linear regression of pressure data has been improved from 0.86956 to 0.9096, while for saturation data the improvement was 0.86286 to 0.92068.

Although the error model is proved to be feasible on predicting errors under different controls (this is permeability in our case, but could also be other values such as bottom hole pressure), when it comes to predict the error of a new control under a new time step, the solution in this thesis has to be improved.

6.2 Possible improvement in Future Works

One possible improvement would be combining MLP with LSTM, or convolutional neural network (CNN) with LSTM. CNN has been developed for long time, and is popular in image processing field. Considering its advantage on image classification (that it has sparse connectivity and shared weight to reduce the parameter number and save computation), We suggest a CNN classification combining with an LSTM regression for similar problems.

Bibliography

[1] Håvard Devold. *Oil and gas production handbook: An introduction to oil and gas production, transport, refining and petrochemical industry*. Edition 3.0, ABB Oil and Gas 2013.

[2] Yongqiang Chena, Quan Xiea, Ahmad Saria, Patrick V. Bradyc and Ali Saeedi. *Oil/water/rock wettability: Influencing factors and implications for low salinity water flooding in carbonate reservoirs*. Fuel 215 (2018) 171177.

[3] Dean Olive, Albert Reynolds and Ning Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press.

[4] T. Heijn, SPE, R. Markovinovi, Delft University of Technology (DUT), and J.D. Jansen, SPE, DUT and Shell International Exploration and Production (SIEP). *Generation of Low-Order Reservoir Models Using System-Theoretical Concepts*. SPE Reservoir Simulation Symposium, 35 February 2003.

[5] Cong Xiao, Olwijn Leeuwenburgh, Hai Xiang Lin, Arnold Heemink. *An Efficient Non-intrusive Subdomain POD-TPWL Algorithm for Reservoir History Matching*. Computational Geosciences, 26 November 2018.

[6] Sumeet Trehan. *Surrogate Modeling For Subsurface Flow: A New Reduced-order Model And Error Estimation Procedures*. Ph.D. thesis, Stanford University 2016.

[7] Sumeet Trehan, Louis J. Durlofsky. *Trajectory piecewise quadratic reduced-order model for subsurface flow, with application to PDE-constrained optimization*. Journal of Computational Physics 326 (2016) 446473.

[8] Jincong He. *Enhanced Linearized Reduced-order Models For Subsurface Flow Simulation*. Master thesis, Stanford University 2010.

[9] M.A. Cardoso and L.J. Durlofsky. *Linearized reduced-order models for subsurface flow simulation*. Journal of Computational Physics 229 (2010) 681700.

[10] Jan Dirk Jansen and Louis J. Durlofsky. *Use of reduced-order models in well control optimization*. Optim Eng (2017) 18:105132.

[11] Sumeet Trehan, Kevin Carlberg and Louis J. Durlofsky. *Error modeling for surrogates of dynamical systems using machine learning*. Int.J. Numer. Meth. Engng 2016; 00:1-32.

[12] Alexander I.J Forrester, András Sóbester and Andy J Keane. *Multi-fidelity optimization via surrogate modeling*. Proc. R. Soc. A (2007) 463, 32513269.

[13] Micha Jerzy Rewienski. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems*. Ph.D. thesis, Technical University of Gdansk, Poland, 1998.

[14] M. Herrmann. *A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure*. Journal of Computational Physics 229 (2010) 745759.

[15] Christoph J. Mack, Peter J. Schmid. *A preconditioned Krylov technique for global hydrodynamic stability analysis of large-scale compressible flows*. Journal of Computational Physics 229 (2010) 541560.

[16] David J Lucia, Paul I King, and Philip S Beran. *Reduced order modeling of a two-dimensional flow with moving shocks.*. Computers and Fluids, 32(7):917938, 2003.

[17] Qifeng Liao And Karen Willcox. *A Domain Decomposition Approach For Uncertainty Analysis*. SIAM J. SCI. COMPUT.Vol. 37, No. 1, pp. A103A133.

[18] Jeremie Bruyelle and Dominique Guerillot. *Neural networks and their derivatives for history matching and reservoir optimization problems*. Computational Geosciences, 18(3-4):549, 2014.

[19] Phani P Chinchapatnam, K Djidjeli, and Prasanth B Nair. *Domain decomposition for time-dependent problems using radial based meshless methods*. Numerical Methods for Partial Differential Equations, 23(1):3859, 2007.

[20] G. Lin, A.M. Tartakovsky, D.M. Tartakovsky . *Uncertainty quantification via random domain decomposition and probabilistic collocation on sparse grids*. Journal of Computational Physics 229 (2010) 69957012.

[21] Janusz S Przemieniecki. *Matrix structural analysis of substructures*. AIAA Journal, 1(1):138147, 1963.

[22] Xin Bian, Zhen Li, and George Em Karniadakis. *Multi-resolution flow simulations by smoothed particle hydrodynamics via domain decomposition*. Journal of Computational Physics, 297:132155,2015.

[23] Q. Deng, V. Ginting, B. McCaskill, P. Torsu. *A locally conservative stabilized continuous Galerkin finite element method for two-phase flow in poroelastic subsurfaces*. Journal of Computational Physics 347 (2017) 7898.

[24] Jaehun Leea and Maenghyo Cho. *An interpolation-based parametric reduced order model combined with component mode synthesis*. Comput. Methods Appl. Mech. Engrg. 319 (2017) 258286.

[25] Mickaël Duval, Jean-Charles Passieux, Michel Salaün and Stéphane Guinard. *Non-intrusive Coupling: Recent Advances and Scalable Nonlinear Domain Decomposition*. Arch Computat Methods Eng (2016) 23:1738.

[26] Knut-Andreas Lie. *An Introduction to Reservoir Simulation Using MATLAB*. SINTEF ICT, Departement of Applied Mathematics, Oslo, Norway.

[27] Oddvar Lia, Henning Omre, Hakon Tjelmeland, Lars Holden and Thore Egeland. *Uncertainties in Reservoir Production Forecasts*. AAPG Bulletin,V.81, No.5(May 1997),P.775-802.

[28] Zhong Y. Wan, Pantelis R. Vlachas, Petros Koumoutsakos, Themistoklis P. Sapsis. *Data-assisted reduced-order modeling of extreme events in complex dynamical systems*. 1 Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA 2 Chair of Computational Science, ETH Zurich, Zurich, Switzerland.

[29] Paul D. Arendt and Daniel W. Apley and Wei Chen. *Improving Identifiability in Model Calibration Using Multiple Responses*. Wei Chen, MD-11-1367, 1.

[30] Sepp Hochreiter. *Long Short-term Memory*. Article in Neural Computationů December 1997.

[31] Tracey B, Duraisamy K, Alonso J. *Application of supervised learning to quantify uncertainties in turbulence and combustion modeling*. 51st AIAA Aerospace Sciences Meeting, vol. 259, 2013.

[32] Weatheritt J, Sandberg R. *A novel evolutionary algorithm applied to algebraic modifications of the RANS stress-strain relationship*. Journal of Computational Physics 2016; 325:22-37.

[33] Duraisamy K, Zhang ZJ, Singh AP. *New approaches in turbulence and transition modeling using data driven techniques*. 53rd AIAA Aerospace Sciences Meeting 2015; doi:10.2514/6.2015-1284.

[34] Arvind T. Mohan* and Datta V. Gaitonde. *A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks*. Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH.

[35] Thomas Fischer and Christopher Krauss. *Deep learning with long short-term memory networks for financial market predictions*. European Journal of Operational Research, Volume 270, Issue 2, 16 October 2018, Pages 654-669.

[36] Saimadhu Polamuri. *How Random Forest Algorithm Works in Machine Learning*. `https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/`

[37] Botros N Hanna, Nam T. Dinh, Robert W. Youngblood and Igor A. Bolotnov. *Coarse-Grid Computational Fluid Dynamic (CG-CFD) Error Prediction using Machine Learning*. Journal of Fluids Engineering 2017.

[38] Botros N Hanna, Nam T. Dinh, Robert W. Youngblood and Igor A. Bolotnov. *LASSO: A feature selection technique in predictive modeling for machine learning*. Advances in Computer Applications (ICACA), IEEE International Conference, 24-24 Oct.2016.

[39] Martin Drohmann And Kevin Carlberg. *The Romes Method For Statistical Modeling Of Reduced-order-model Error*. JUQ. December 2014.

[40] Trevor Hastie and Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning*. Second Edition, Springer.

[41] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer.