The Pennsylvania State University

The Graduate School

Department of Energy and Mineral Engineering

CHARACTERIZATION OF SEALING AND PARTIALLY COMMUNICATING FAULTS IN DUAL-POROSITY GAS RESERVOIRS USING ARTIFICIAL NEURAL NETWORKS

A Thesis in

Energy and Mineral Engineering

by

Zhazar Toktabolat

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2012

The thesis of Zhazar Toktabolat was reviewed and approved* by the following:

Turgay Ertekin
Professor of Petroleum and Natural Gas Engineering
George E.Timble Chair in Earth and Mineral Sciences
Thesis Advisor

Zuleima T. Karpyn
Associate Professor of Petroleum and Natural Gas Engineering

John Yilin Wang
Assistant Professor of Petroleum and Natural Gas Engineering

R.Larry Grayson
Professor of Energy and Mineral Engineering
Graduate Program Officer of Energy and Mineral Engineering

*Signatures are on file in the Graduate School

ABSTRACT

When unconventional hydrocarbon sources are added to the potential resources, the expansion of petroleum industry becomes a vast opportunity. The transient pressure analysis of gas wells in low permeability naturally fractured (tight gas and shale gas) reservoirs have gained wide interest among reservoir modelers, and various formation evaluation methods have been developed. In this work, the combination of analytical model and numerical reservoir simulator are used to generate and simulate a large number of hypothetical reservoirs which encompass wide range of dual porosity tight systems. The framework of this study is to convert observed measurements of reservoir data into characteristic information about the system that we are interested in. Artificial neural network (ANN) technology is utilized in mapping/interpolating the non-linear complex relationship between observed measurements and reservoir characteristics. The proposed ANN methodology is applied towards analyzing the pressure transient measurements collected from isotropic and anisotropic faulted dual-porosity gas reservoirs, as an inverse solution to formation characteristics, such as the permeability and porosity of the fracture and matrix systems, distance to the fault, orientation of the fault with respect to the principal flow directions, and sealing capacity of the fault are predicted using the reservoir fluid, rock, and bottom-hole pressure as the principal inputs. The main focus of this study is to develop a suitable network to obtain accurate prediction about desired reservoir characteristics of dual-porosity tight gas systems with a fault, and demonstrate the efficient processing power of ANN on this class of reservoir problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Nomenclature

h	formation thickness, ft
φ_m	porosity-matrix
φ_f	porosity-fracture
k_m	permeability-matrix, md, i, j, k
k_f	permeability-fracture, md, i, j, k
k_{fx}	fracture permeability in the *x* direction, md, (anisotropy)
k_{fy}	fracture permeability in the *y* direction, md, (anisotropy)
l	fracture spacing, ft, i, j, k
T	reservoir temperature, ^0F
p_i	initial reservoir pressure, psi
γ_g	specific gravity of gas
c_g	compressibility of gas, 1/psi
c_m	compressibility of matrix pore, 1/psi
c_f	compressibility of fracture pore, 1/psi
q_{sc}	gas production rate, mmscf/day
L_x	distance to the fault, ft
ε	fault communication index
θ	orientation of the fault, degrees
ε	fault communication index
λ	inter-porosity flow parameter
ω	storage capacity

SC	sealing capacity, %
$\frac{dp}{d\ log_{10}(t)}$	slop of pressure against logarithm of time
$\frac{d\psi_D^i}{d\ log_{10}(t)}$	modified dimensionless pseudo-pressure drop against logarithm of time
ψ_D^i	modified dimensionless pseudo-pressure
ψ	pseudo-pressure
$\psi^i{}_{Di}$	modified dimensionless initial pseudo-pressure
p_i	initial pressure
$t_D{}^i$	modified dimensionless time
r_w	well radius
i,j,k	spatial directions in reservoir simulator
Mse	mean square error
msereg	mean square error regularization
O	outputs
x	representation of direction along x axis
y	representation of direction along y axis
Δp	pressure drop
l_F	thickness of the fault (l_f)
h_F	the height of the fault, ft
k_F	fault permeability
L	distance from producing well to the observation well
Sg	gas saturation
Sw	water saturation
Swi	irreducible water saturation

kr	relative permeability
krg	relative permeability of gas
Z	compressibility factor
B_g	formation volume factors
μ_g	gas viscosity at reservoir pressure and temperature
μ_{g1}	gas viscosity at atmospheric pressure and reservoir temperature
p_{wf}	bottom-hole flowing pressure
t	time

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my thesis advisor and my mentor Dr. Turgay Ertekin for his generous guidance and constant support throughout the course of my study. His wide knowledge and logical approach toward problems in new, innovative, and systematic ways have been of great value to me. In spite of the many obstacles, he constantly encouraged me to continue and he helped me navigate through my research with his penetrating insight, and rich experience.

I'd like to take this opportunity to show my sincere appreciation to BG-group (BG) and to Bolashak scholarship for their financial support during my master's education, and extend my warmest gratitude to all the professors, staff and students of the College for their support. Thanks to all those who have helped me with my work.

I owe my loving thanks to my wife Akhmetova Ainur and my son Zhazaruly Aslan for being ultimate joy and purpose of my life. They have sacrificed a lot due to my research abroad. Without their encouragement and understanding it would have been impossible for me to finish this work. My special gratitude goes to my parents, Asemkan and Toktabolat, my siblings, Zhupar, Khaisar, Khaidar, and my extended family, Omirbek, Zhanar, Alisher, and Alibek. To them I dedicate this work.

During this work I have collaborated with many friends, here in Penn State, Kazakhstan, and other parts of the world, for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work in Department of Petroleum and Natural Gas Engineering.

Chapter 1

Introduction

Rising energy demands of the world is compelling us to look for unconventional hydrocarbon resources. Especially, the exploration of unconventional gas reservoirs has gained momentum in last two decades, and naturally fractured tight gas reservoirs represent important class of world's hydrocarbon reserves. Naturally fractured reservoirs are found all over the world in all types of lithologies and throughout the stratigraphic column. They introduce considerable difficulties in the description of rock structure and the fluid flow in matrix-fracture network.

Experts believe that natural gas is the fuel of the future, and its use will spread substantially. Natural gas is actually 70% cleaner than coal while replacing all fossil fuels with renewable energy sources will be a premature proposition. Gas may be the next best thing. This has led to increased research in unconventional reservoirs. Thus, one of the important aspects of this research is well testing.

In reservoir engineering, quantitative description of a hydrocarbon reservoir is vitally important consideration for every aspect of exploration. Identifying the critical parameters and formulating a reservoir characterization would support subsequent development activities, and serve as crucial information for achieving the goal of reducing overall costs. Apart from mud logging, core sampling, and wire line logging, well testing is a powerful reservoir characterization tool and well-performance indicator. Interpreting the key parameters from recorded long term bottom-hole pressure under constant production rate and other direct measurable formation properties is an inverse problem in nature; this study focuses on obtaining accurate prediction about desired reservoir characteristics of dual-porosity gas systems with a fault, and demonstrating the efficient

processing power of ANN on this class of reservoir problems. The goal includes determination of characteristic properties of matrix and fracture systems, namely: permeability, porosity, distance to the fault, orientation with respect to the principal flow direction, and sealing capacity of the fault.

In Chapter 2, a brief history of the basic building blocks of ANN, and its simplified description are presented. Furthermore, mathematical model of the neuron and how these artificial neurons can be interconnected to form variety network architectures are explained and the theoretical and imperial grounds for selecting suitable ANN structure are discussed. As a pressure transient analysis tool, ANN is a simple and economical method for reservoir characterization. This study considers a class of problem where bottom-hole pressure (BHP) profile under long term constant rate, i.e. prolonged drawdown test is studied. For the sake of this study, bottom-hole pressure is recorded through ten year time spans at designed time intervals.

In Chapter 3, a brief description of fractured reservoirs is described; emphasis is placed on quantitative evaluation of formation parameters. A single phase reservoir simulation approach and its mathematical representation is introduced (the forward solution part of the study); basic strategy in acquiring and validating reservoirs data with unique dual-porosity signature is discussed. This discussion goes into details of the constraints and analytical model for generating reservoir data and gas properties. Afterwards, simulation of various production schemes and formation properties are performed, and all relevant reservoir data subject to this study are collected, validated, and incorporated into an acceptable dual-porosity gas field model.

Chapter 4 starts with the building of the ANN structure with a single output. The ANN development for multivariate structures are done in three major stages by increasing the number

of unknowns in the output layer. The predictability of each reservoir characteristics is assessed. The final goal is to predict formation properties as well as fault properties in the system within reasonable accuracy. Case studies for different reservoirs are provided in chapter 5 to demonstrate the implementation of the developed ANN. Chapter 6 is devoted to analyze and discuss the results, and provide summary and conclusions derived from this research.

Chapter 2

Artificial Neural Networks

This chapter provides a brief history of neural networks, and the general rules and protocols followed in constructing neural network architecture. Special emphasis is placed on the back propagation algorithm which is the key to solving the inverse problems.

2.1 Background of Neural Networks

The technological advances in personal and main-frame computer hardware and computational power have enabled us to study more complex systems. Complex problems require collaboration between traditional sciences, such as, biology, computer science, physics, and mathematics, so that the detailed model of this collaboration can mimic human behavior. If we are to describe the function of a human brain in mechanical terms, it is a highly complex, nonlinear, and parallel information processing system, capable of organizing huge amount of data. The adjective "neural" is the direct indication that the inspiration for artificial neural networks comes from neuroscience; the past decade has seen an explosive growth in neural networks and the theory build up around neural networks.

Hermann von Helmholtz, Ernst Mach, and Ivan Pavlov were the pioneers in some of the background work for the field of neural networks in the late 19^{th} and early 20^{th} centuries. The modern view of neural networks began in the 1940s with the work of Warren McCulloch and Walter Pitts [McCulloch et al, 1943]. In 1949, Donald Hebb, who has been described as the father of neuropsychology and neural networks, proposed one of the first neural network learning laws later known as Hebb's Postulate [Hebb, 1949]. This postulate took a bottom-up approach and

steered the field of study toward quantifiable microscopic neurons instead of correlations between stimulus and response. In the late 1950s, a computer scientist, Frank Rosenblatt, built the first perceptron network computer, the MARK 1, with his colleagues that could learn new skills by trial and error, at Cornell University [Rosenblatt, 1958].

Frank Rosenblatt was followed by Bernard Widrow and Ted Hoff [Widrow and Hoff, 1960] who introduced a similar training algorithm, which is widely used in adaptive signal processing. The Widrow-Hoff learning rule is now a well-known supervised learning rule, and its adaptation is known as error back-propagation, where the difference between the input layer and output layer is used as an error to update corresponding weights in each cell. The function of the learning algorithm can be seen as a way to modify synaptic weights of the network in an orderly fashion to obtain desired objective. However, the perceptron was essentially a linear classifier which only solves linearly separable problems, and often failed to map the non-linear input-output relationships of non-separable data.

In late 1960s, Minsky and Papert identified two key issues associated with perceptron type machine learning.First, the single layer (no hidden layer) simple networks appear to be incapable of implementing certain elementary functions. The second significant issue was that the limited processing power of the machines of that time could not handle large networks. The discovery of this inadequacy caused such models to be abandoned and contributed to dampening of continued interest in the neural network, the research in this field halted nearly for a decade. Some important work, however, did continue during 1970s. The work done by Willshaw and von der Malsburg on self-organizing networks using competitive learning was a perfect example of continued commitment to neural networks.

In the 1980s, as computers achieving greater processing power, impediments of hardware were overcome and research in neural networks increased dramatically. In 1982, John Hopfield presented his associative network with clear mathematical analysis. Given his background in physics, chemistry, molecular biology, and mathematics, he was able to identify a close analogy between his neural network and Ernst Ising's mathematical model of ferromagnetism in statistical mechanics [Ising, 1925], and brought other scientists' attention to neural network research. Another important thread of development in the 1980s was the rediscovery of the back-propagation algorithm for training multilayer perceptron networks. The most influential publication of the back-propagation algorithm was by David Rumelhart and James McClelland [McClelland, 1986], although, similar suggestion had been made independently by several other researchers. The new developments reinvigorated the field of neural networks, and they have certainly come a long way from the early days of McCulloch and Pitts. To this date, many advances in theory, design, and application of neural networks have had to do with new concepts. In the last twenty years numerous books have been published, thousands of papers have been written, and there are broad ranges of applications that can be found for neural networks.

A common difficulty of neural networks is that they require a large diversity of training data for real-world problems, since any learning machine needs sufficient representative examples in order to formulate the underlying relationship of a system that allows it to generalize to new cases within established parameters. In practice, neural networks cannot provide the solution on its own at once; rather, the complex problem of interest is decomposed into its basic units or relatively simple tasks where the input- output relationship can be felt and modified accordingly. However, the flexibility, ability to implicitly detect complex relationships, faster computers, and efficient algorithms made it possible to use neural networks to solve complex industrial problems.

2.2 Introduction to Neural Networks

Neural networks are known to be universal function approximators, which is an alternative computational paradigm based on a programmed instruction sequences. Although neural networks are not biologically realistic in detail, they draw their methods a large degree from statistical physics and mathematics, inspiration from neuroscience. The artificial neural networks are remotely related to their biological counterparts. Similar to human the brain, the configuration and function of neural networks depend on their architecture; the processing ability is stored in the units or elements which are suitably updated weights and biases.

The brain is composed of about 80 to 120 billion neurons (nerve cells) of many different types. Figure 2.1 is a schematic drawing of a single neuron. Tree-like networks of nerve fiber called dendrites are connected to the cell body or soma, where the cell nucleus is located. Extending from the cell body is a single long fiber called the axon, which eventually branches into strands and substrands. At the ends of these are the transmitting ends of the synaptic junctions, or synapses, to other neurons which are connected to other neurons. The receiving ends of these junctions on the other cells can be found both on the dendrites and on the cell bodies themselves [John Hertz, 1991]. As depicted in Figure 2.2, the simple bio neuron can be mathematically represented as an artificial neuron. When artificial neurons organized into a network of certain pattern to perform a task, in a way this network imitates human brain.

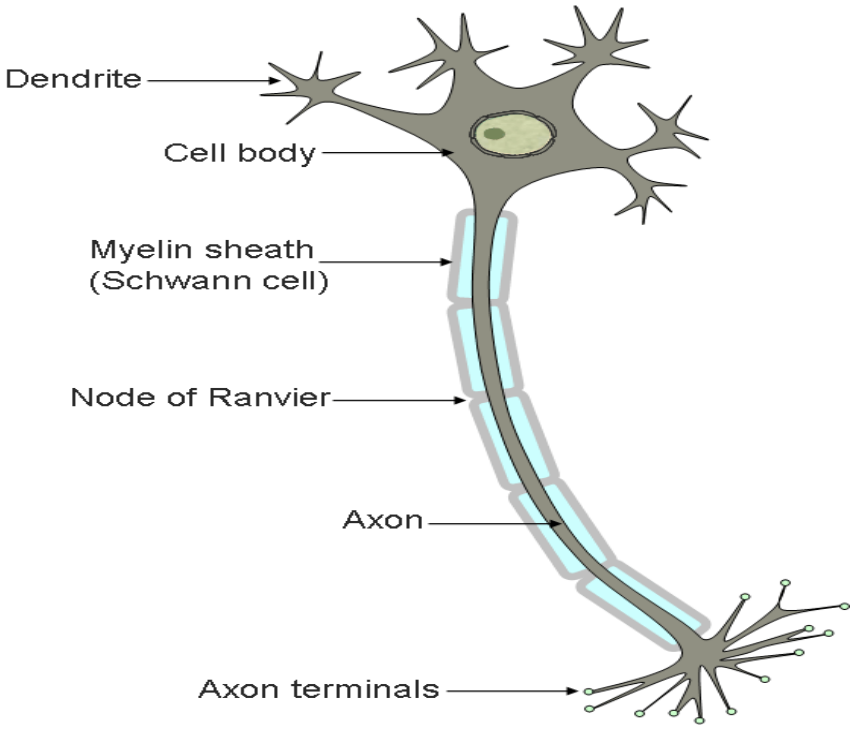


Figure 2.1: Schematic drawing of a typical neuron

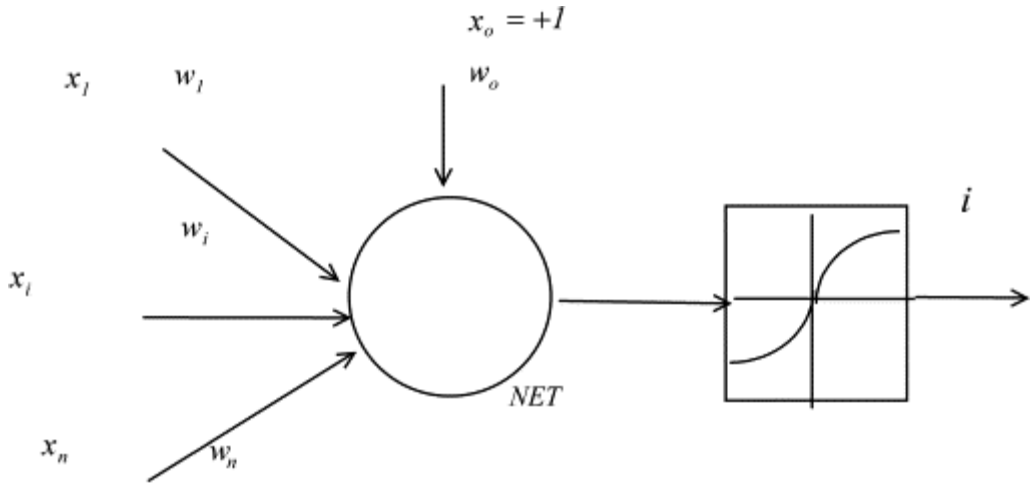


Figure 2.2: Schematic drawing of an artificial neuron

Artificial neural networks do not simulate the complexity of the brain. However, the building blocks of neural networks are similar to that of the brain. It resembles the brain in two respects [Haykin, 1994]:

1. Knowledge is acquired by the network through a learning process
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge

2.3 ANN Selection

There are many types of ANNs. Some are hardware based, some are purely software based, and some are combination of both. Depending on their application and the physics of the problem, the available neural network structures within the developed systems are highlighted the following:

1. Feed forward neural network
2. Radial basis networks
3. Self-organizing networks
4. Learning vector quantization
5. Recurrent networks
6. Modular networks
7. Physical networks

Neural networks employ three basic leaning rules: supervised learning, unsupervised learning, and reinforcement (or graded) learning. In supervised learning, the system specifics and the expected response are presented as inputs applied to the network and the network outputs are compared to the targets. In unsupervised leaning and only the system situations are represented,

the network response is calculated only from inputs, there are no target outputs available. These types of networks are mostly designed to organize or cluster the patterns into a finite number of categories. Reinforcement learning is somewhat similar to supervised learning, where targets outputs are not clearly defined; instead, the network response is graded. Another important factor in defining neural network model is the topology or architecture, and there are wide ranges of possibilities one can choose from. After researching for suitable networks, it is found that the multilayer back-propagation networks are best for solving the inverse problems which are the very nature of this study.

2.4 Multilayer Back-propagation Neural Networks (MBPNN)

The back-propagation algorithm is central to much current work on learning in neural networks, and the algorithm is based on error correction learning rule consists of forward and backward passes. Typically, groups of nodes (i.e. neurons) make up the input layer, hidden layers, and output layer, where the information vectors in the input layer propagate forward through hidden layers using nonlinear differentiable transfer functions to translate the input signals to output signals. This is known as the forward pass in which the synaptic or neural weights are assigned accordingly. During the backward pass, the synaptic weights are all updated in accordance with the error-correction rule. The Figure 2.3 depicts a portion of the multilayer network, where two kinds of signals are identified. In fact, the back-propagation algorithm is bidirectional and the actual response of the network is subtracted from a desired (target) response to produce error signal. This difference is then treated with appropriate error dependent performance methods (mean squared error, mean absolute error, etc.) and back propagated through the network, so that overall network response is in line with desired response. It is mentioning that back-propagation algorithm has its limitations, although, many extensions, modification, and variations of this basic

algorithm have been suggested to make the convergence in multilayer networks computationally efficient. It is used in many different types of applications. This architecture has spawned a large class of network types with many different topologies and training methods. Back-propagation algorithm is considered to be the best algorithm among the multilayer perceptron algorithm. The number of layers, processing node per layer, and how the layers are connected are essential to back-propagation topology. Its greatest strength is in non-linear solutions to ill-posed problems [International Journal of Computer Science and Network, 2009].

2.5 MBPN Architecture

Updating of weights and biases pattern-by-pattern is the preferred method for on-line implementation of the back-propagation algorithm. For this mode of operation, the algorithm cycles through the training data as follows. Figure 2.4 depicts basic features of back-propagation algorithm.

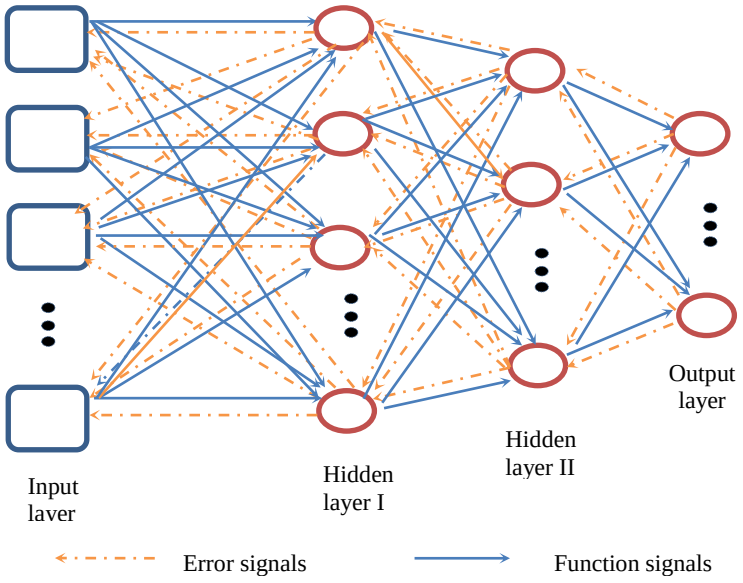


Figure 2.3: The forward propagation of function and back-propagation signals

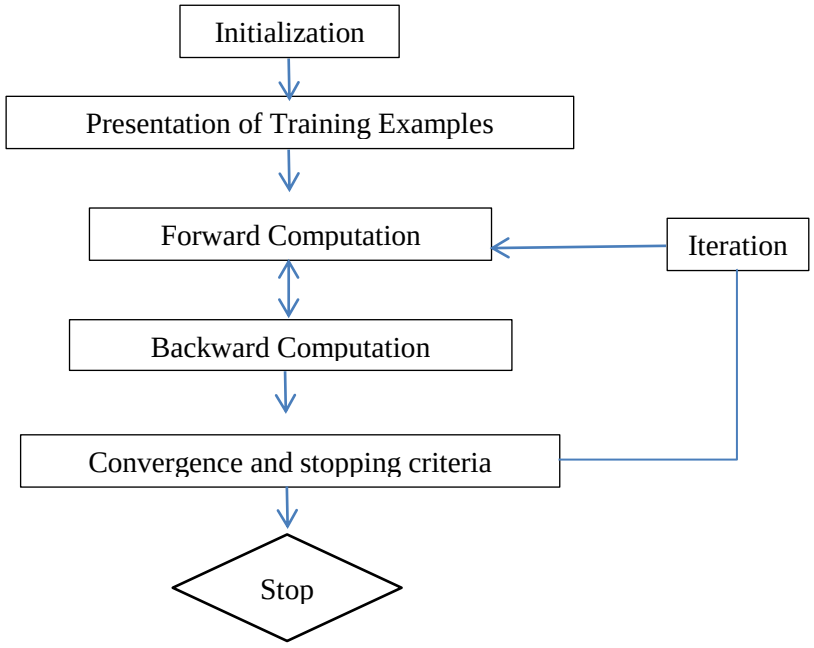


Figure 2.4: Flow chart of back-propagation network

2.6 General model of Back-propagation Algorithm and Its Variations

The mathematical expressions are important in understanding the back-propagation algorithm. In this section, a brief overview of the mathematical expressions of the back-propagation algorithm is described. The exact definitions theorems and proofs can be found in *Neural Network Design* [Hagan, 1996].

To illustrate the structure of a multilayer networks, in figures, mathematical equations, and texts the following shorthand notations are used:

Scalars—small *italic* letters

Input, hidden layer output, output, and target vectors —small **bold** non-italic letters

Weight matrices—Capital **BOLD** letter or **W** with subscripts *i* and *j* representing the layer and neuron numbers

Input vectors— small **bold** *italic* letters, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3 \ldots \boldsymbol{p}_R$ with subscripts

Individual input elements— small *italic* letters, p_1, p_2, $p_3 \ldots p_R$ with subscripts

For a given training set, back-propagation learning may proceed in one of two basic ways.

$$\mathbf{a}^{m+1} = \mathbf{f}^{+1}\left(\boldsymbol{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}\right) \tag{2.1}$$

where m is the number of layers in the network, when m=0, $\mathbf{a}^0=\mathbf{p}$. The outputs in the last layer are denoted to be $\mathbf{a}= \mathbf{a}^M$.

Let **f** be any continues sigmoid type activation function, for example

$$f(s) = \frac{1}{1+\exp(-\lambda s)} \quad \lambda \geq 1 \tag{2.2}$$

Satisfies the nonlinear ordinary differential equation

$$\frac{df}{ds} = \lambda f(1-f) \tag{2.3}$$

For the network with a single output, error signal on the output layer is calculated as:

$$F(x) = E[e^2] = E[(\mathrm{t} - \mathrm{a})^2] \tag{2.4}$$

For the network with multiple outputs, error signal on the output layer is calculated as:

$$F(x) = E[\boldsymbol{e}^T \boldsymbol{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})] \tag{2.5}$$

$$\hat{F}(x) = \big(\mathbf{t}(k) - \boldsymbol{a}(k)\big)^T \big(\mathbf{t}(k) - \boldsymbol{a}(k)\big) = \boldsymbol{e}(k)^T \boldsymbol{e}(k) \tag{2.6}$$

where *k* indicates iteration level. The steepest decent algorithm for the approximate mean square error is

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m} \tag{2.7}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m} \tag{2.8}$$

where α is the learning rate and using chain rule we get

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \quad \text{and} \quad \frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{2.9}$$

The second term in each of above two equations can be expressed as explicit function of the weights and biases of that layer.

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \tag{2.10}$$

Therefore

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{2.11}$$

If one defines

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m} \tag{2.12}$$

The sensitivity of *F* to changes in *i*th element of the net input at layer *m*, then equation can be simplified to

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = s_i^m a_j^{m-1} \tag{2.13}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{2.14}$$

Now one can express the weight and bias update on steepest decent algorithm as

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha s_i^m a_j^{m-1} \tag{2.15}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{2.16}$$

In matrix form

$$\boldsymbol{w}^m(k+1) = \boldsymbol{w}^m(k) - \alpha \boldsymbol{s}^m (\boldsymbol{a}^{m-1})^T \tag{2.17}$$

$$\boldsymbol{b}^m(k+1) = \boldsymbol{b}^m(k) - \alpha \boldsymbol{s}^m \tag{2.18}$$

where

$$\boldsymbol{s}^m = \frac{\partial \hat{\boldsymbol{F}}}{\partial \boldsymbol{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{s^m}^m} \end{bmatrix} \tag{2.19}$$

Now backward pass of solving $\boldsymbol{S^m}$ from $\boldsymbol{S^{m+1}}$ involves formulation of the Jacobian matrix.

$$\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{s^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_{s^m}^m} \end{bmatrix} \tag{2.20}$$

Considering the *i,j* element of the matrix

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{j=1}^{S^m} w_{i,j}^{m+1} a_j^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1}\frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1}\frac{\partial f^m(n_j^m)}{\partial n_j^m}$$
$$= w_{i,j}^{m+1}\dot{f}^m(n_j^m) \tag{2.21}$$

The whole Jacobian matrix can be written as

$$\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m} = \boldsymbol{w}^{m+1}\dot{\boldsymbol{F}}^m(\boldsymbol{n}^m) \tag{2.22}$$

where

$$\dot{\boldsymbol{F}}^m(\boldsymbol{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 \dots & 0 \\ 0 & \dot{f}^m(n_2^m) \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 \cdots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \tag{2.23}$$

Utilizing the chain rule, again, the sensitivity relationship between the two consequent layers $\boldsymbol{S^m}$ and $\boldsymbol{S^{m+1}}$ can be written in matrix forms as

$$\boldsymbol{s}^m = \frac{\partial \widehat{\boldsymbol{F}}}{\partial \boldsymbol{n}^m} = \left(\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m}\right)^T \frac{\partial \widehat{\boldsymbol{F}}}{\partial \boldsymbol{n}^{m+1}} = \dot{\boldsymbol{F}}^m(\boldsymbol{n}^m)(\boldsymbol{w}^{m+1})^T \frac{\partial \widehat{\boldsymbol{F}}}{\partial \boldsymbol{n}^{m+1}}$$
$$= \dot{\boldsymbol{F}}^m(\boldsymbol{n}^m)(\boldsymbol{w}^{m+1})^T \boldsymbol{s^{m+1}} \tag{2.24}$$

The starting point for $\boldsymbol{S^m}$ will be at the output layer $\boldsymbol{S^M}$

$$\boldsymbol{s}^M = -2\dot{\boldsymbol{F}}^M(\boldsymbol{n}^M)(\boldsymbol{t} - \mathbf{a}) \tag{2.25}$$

For more information about the implementation of this general back-propagation model please refer to chapter 11 in *Neural Network Design* [Hagan et al, 1996] and chapter 6 in *Neural Networks* [Haykin, 1994].

2.7 Modifications to the Back-propagation Networks

According to Cybenko's Theorem, one hidden layer feed forward neural network is capable of approximating uniformly any continuous multivariate function to any desired degree of accuracy.

In other words, inadequate input-output mapping of multilayer network is due to the network parameters [Steeb et al, 2011]. The convergence of the basic general algorithm given above tends to be relatively slow; the derivative of the error surface with respect to the weight can be small in magnitude where the error surface is flat, and many iterations are required to navigate through this region. Alternatively, in highly curved error surface regions, the large gradient of the instantaneous error surface in weight space causes the algorithm to overshoot the global minimum. Another possible scenario is that the adjustments applied to weights can cause the algorithm to move in wrong direction. In practice, various methods have been developed to make back-propagation network faster, reliable, and more adaptable in terms of optimization. It is worth mentioning that there are many parameters one can modify within general back-propagation framework. We mainly focus on the update rule, namely, alternative cost functions (error functions), momentum, learning rate and numerical optimization techniques.

2.7.1 Alternative Cost Functions

The first method is to use alternative cost functions in which the error surface is modified to facilitate a safe search space for avoiding local minima. One technique is to smooth out the surface initially by adding parameters to the cost function so as to adjust its steepness or roughness. An example was provided by Makram__Ebeid et al in 1989 shown in Equation 2.26:

$$E = \sum \begin{cases} \gamma(t_i - \mathrm{a}_i)^2 & if\ sgn\ (t_i) = sgn\ (\mathrm{a}_i) \\ \gamma(t_i - \mathrm{a}_i)^2 & if\ sgn\ (t_i) = -sgn\ (\mathrm{a}_i) \end{cases} \tag{2.26}$$

Another technique is to use alternative differentiable functions, one of which has received particular attention [Baum and Wilczek, 1988; Hopfield, 1987; Solla et al, 1988].

$$E = \sum (\frac{1}{2}(1 + a_i) \log\left(\frac{1 + a_i}{1 + t_i}\right) + \frac{1}{2}(1 - a_i) \log\left(\frac{1 - a_i}{1 - t_i}\right)) \quad (2.27)$$

Mean absolute error, sum of squared error, and mean squared error are the most popular cost functions (performance functions) employed in Neural Network Toolbox in Matlab.

2.7.2 Heuristic Variations of Back-propagation

In 1988 Jacobs suggested a heuristic method for accelerating the convergence of back-propagation that has practical value. Modifying the error surface aside, there are a number of sophisticated minimization algorithms apart from gradient decent method, however, the simpler approach is to accelerate or decelerate the weight and bias update by adding momentum term. Recall Equation 2.7 and 2.8.

$$\Delta \boldsymbol{w}^m(k) = -\alpha \boldsymbol{s}^m (\boldsymbol{a}^{m-1})^T \quad (2.28)$$

$$\Delta \boldsymbol{b}^m(k) = -\alpha \boldsymbol{s}^m \quad (2.29)$$

The idea is to give a γ little inertia or kick in the direction of the average downhill to decrease the number of necessary iterations.

$$\Delta \boldsymbol{w}^m(k+1) = \gamma \Delta \boldsymbol{w}^m(k) - (1 - \gamma)\alpha \boldsymbol{s}^m (\boldsymbol{a}^{m-1})^T \quad (2.30)$$

$$\Delta \boldsymbol{b}^m(k+1) = \gamma \Delta \boldsymbol{b}^m(k) - (1 - \gamma)\alpha \boldsymbol{s}^m \quad (2.31)$$

where γ is the momentum parameter and α is the learning rate [Hagan, Demuth and Beale, 1996]. In this work, since we deal with multiple outputs and multilayer networks and the size of the training set is large, our best option is to update the weight and bias in batch mod (in batch mode the weights and biases of the network are updated only after the entire training set has been presented to the network. The gradients calculated at each training example are added together to determine the change in weights and biases [Hagan et al, 1996]). Adjusting the learning rate

(scale parameter or step size) during the search iterations is another strategy that further improves the convergence. There are many other heuristic approaches developed for learning rate adaptation. Nevertheless, the basic guideline is to check the effect of the weight update. If the cost function change decreases within the established margin of error after the weight update, one can try to increase the learning rate. If the opposite of which was true, then the process could overshoot, the learning rate should be decreased proportionately. It is advisable to increase α by step wise, but decrease it geometrically [Cater, 1987; Franzini 1987; Vogl et al., 1988; Jacobs, 1988].

2.7.3 Numerical Optimization Techniques

In this section we consider a couple of gradient based numerical optimization (minimization) methods which are pertinent to this work, namely, conjugate gradient (and its variations) and Levenberg-Marquardt. Numerical optimization techniques are studied in depth over the years, in which two steps are iteratively performed:

- Finding a direction that will improve the objective
- Searching in this direction until no more improvement is available

Conjugate gradient techniques are well suited for solving large scale nonlinear problems. The advantages of these algorithms are that they require no matrix storage and are faster than general back-propagation algorithm convenient for parallel computing. The Levenberg-Marquardt, on the other hand, is a variation of Newton's method, where it involves the inversion of Hessian matrix. One should consider the memory storage capacity of the computer since large matrix inversion requires big memory allocation, especially, when the training set size is large.

There have been comparative studies to determine the overall performances of above mentioned techniques. All these modifications to general back-propagation algorithm—Steepest decent, Momentum, variable learning rate, conjugate gradient, and Levenberg-Marquardt—replace only the gradient decent rule, not the gradient calculation.

Chapter 3

Well Performance and Well Test Analysis in Fractured Reservoirs with a Fault

This section focuses on the pressure transient behavior of a well in an naturally fractured reservoirs. The theoretical basis of drawdown test analysis and the model chosen to explain the dual-porosity system where sealing and partially communicating faults are present is demonstrated.

3.1 Introduction

Faults and fractures are commonplaces in a reservoir study. The basic objective of this study is to determine the naturally fractured reservoir characteristics through well tests, in particular, the drawdown tests. In the reservoirs with fractures, the transient flow behavior is identified by the plot of pressure against the logarithm of time consisting of two straight lines with similar slopes joined by an *S* shaped curve both by the theoretical modeling and the field observation. The linear boundary (a fault) is identified by the occurrence of slope change. For faults close to the well, the slope behavior in both the early and the middle times may be influenced and result in additional pattern on top of double-porosity pattern. Figure 3.1 shows the drawdown test of a homogenous, double-porosity gas reservoir with a sealing fault located 200 feet away from the wellbore. The linear boundary effect is seen first in the early time period, the transition period is distorted, and the late time response is also affected. The features expected for separation of two straight lines typical of a reservoir with fractures is also depicted in the figure. It is important to note that all simulated pressure transients are limited to data free of wellbore storage and skin.

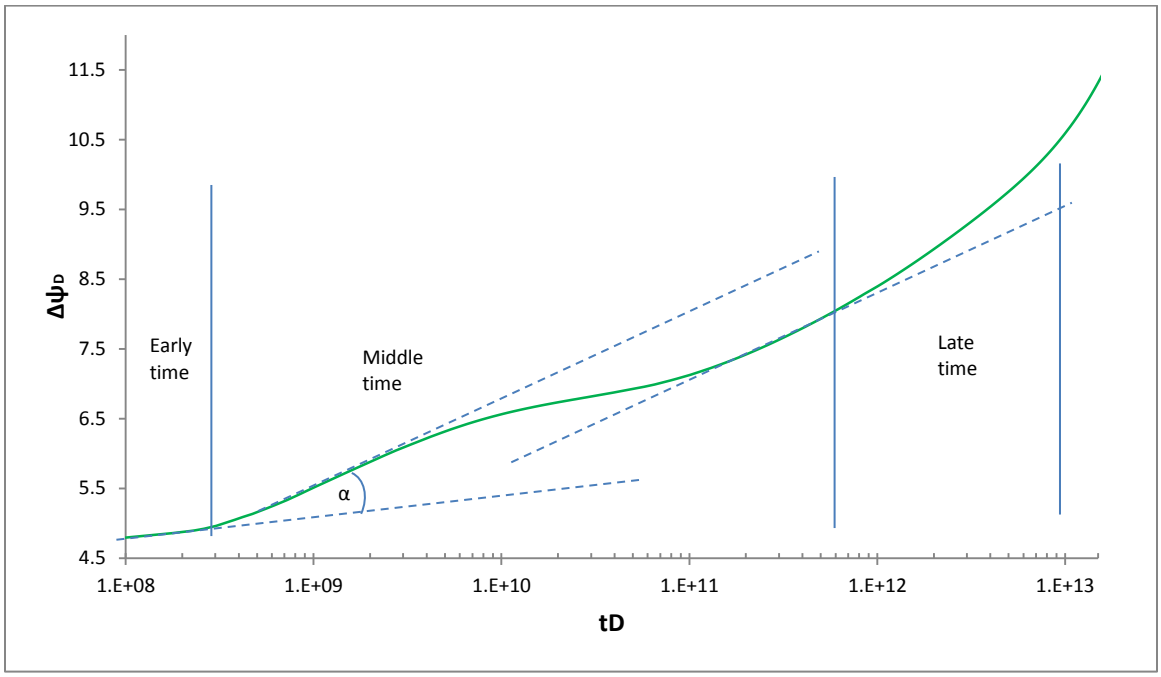


Figure 3.1: Pressure drawdown test of the double-porosity model with sealing fault on semi-log plot

The drawdown tests in this study designed in the domain of naturally fractured gas reservoirs with a sealing fault or partially communicating fault. Characteristic patterns of the tests are investigated to extract the main variables L_x, λ, ω, k_f, k_m, φ_f, φ_m, θ, SC, and ε.

3.3 Mathematical Modeling of Given System

The mathematical representation of fluid-flow in naturally fractured media is based on following assumptions:

- The reservoir is infinite-acting homogenous naturally fractured gas reservoir
- Single well under constant production rate
- Continuous uniform fracture system
- Fluid channeled to the well only via fracture network

- Pseudo steady state inter-porosity flow condition between matrix and fracture systems
- The fault line is infinitely long with almost no storage capacity

The compressible-fluid-flow in Equation 3.7 is the combination of Darcy's law with continuity equation; and the fracture-matrix relationship is modeled by Warren and Root formulation. The gravitational effects of gas phase (the only phase in this study) are neglected. It is assumed in Warren and Root formulation that no pressure transient takes place in matrix blocks. Therefore

$$-Q * q_{SC/D} = \frac{V_b}{5.615}\Phi_m \frac{\partial}{\partial t}\left(\frac{p_m}{z_m}\right) \tag{3.7}$$

and

$$\frac{\partial}{\partial x}\left(\frac{A_x k_x}{\mu_g}\frac{p_f}{z_f}\frac{\partial p_f}{\partial x}\right)\Delta x + \frac{\partial}{\partial y}\left(\frac{A_y k_y}{\mu_g}\frac{p_f}{z_f}\frac{\partial p_f}{\partial y}\right)\Delta y + Q * q_{\frac{SCF}{D}} = \frac{V_b}{5.615}\Phi_f \frac{\partial}{\partial t}\left(\frac{p_f}{z_f}\right) + \frac{V_b}{5.615}\Phi_m \frac{\partial}{\partial t}\left(\frac{p_m}{z_m}\right) \tag{3.8}$$

Equation 3.8 is solved numerically for each and every block.

3.2 Characterization of Naturally Fractured Reservoir with a Fault

There are extensive studies about the physical properties of rocks and fluids. Since the objective of this chapter is to understand formation evaluation of naturally fractured reservoirs by well testing it is important to provide the definitions of the properties of naturally fractured reservoirs which are different from conventional reservoirs. Naturally fractured reservoirs can be studied through the use of dual porosity models. The essence of this model is to discretize the naturally fractured reservoir into two collocated continua (two sets of grid blocks located in the same space), matrix and fracture. The system consists of matrix blocks which are separated spatially by fractures. Thus the rock properties, such as permeability, porosity, compressibility, fracture

spacing (intensity) will be discussed as properties of the fractures or of the fracture-matrix system.

Fractured reservoirs have two porosity systems: intergranular spaces and void spaces of fractures. The total porosity is simply the addition of the two systems (Equation 3.1), and the conventional definitions are primary and secondary porosities.

$$\begin{cases} \Phi_t = \Phi_m + \Phi_f \\ \Phi_m = matrix\ pore\ volume/total\ bulk\ volume \\ \Phi_f = fracture\ pore\ volume/total\ bulk\ volume \end{cases} \quad (3.1)$$

Large number of laboratory measurements on various types of rocks suggests that the matrix porosity is larger than fracture porosity. However, it is difficult to evaluate such systems' especially when the value of permeability is small. Under certain conditions, Warren-Root model (otherwise known as sugar cube model) is chosen to idealize the fracture-matrix system such that the relationship between φ_m, φ_f, and φ_t, is simplified. In the Warren-Root model, it is assumed that the matrix blocks feed the fractures and fluid flows to wellbore only through the fracture network. The presence of fracture contributes to the increase in overall permeability of the system.

In well testing, the compressibility is an essential parameter. Since the object of this study is gas reservoirs, the gas compressibility (c_g) represents the large portion of the total system's compressibility. There are four components to the total compressibility in this context; reservoir fluid compressibility (c_f), fracture pore compressibility (c_{pf}), matrix pore compressibility (c_{pm}), and rock compressibility (c_r). Considering the wide variety of double porosity gas reservoirs under study, The quantitative measurement of fracture and matrix pore compressibilities are respectively taken within the ranges of $10^{-5}>c_f>7\times10^{-5}$ and $1.5\times10^{-5}>c_f>2\times10^{-6}$. In the double porosity system, the transient pressure behavior is directly related to the storage capacity

parameter which, is in turn, expressed by compressibility. The parameter ω represents dimensionless storage capacity.

$$\omega = \Phi_f C_{pf}/(\Phi_m C_{pm} + \Phi_f C_{pf}) \quad (3.2)$$

Another important parameter which controls pressure behavior is interporosity flowing capacity λ

$$\lambda = n(n+2)\left(\frac{r_w}{\alpha}\right)^2 \frac{K_m}{K_m(K_f + K_m)} \frac{(\Phi_m C_{pm} + \Phi_f C_{pf})}{\Phi_m C_{pm}} \quad (3.3)$$

where

n = number of fracture planes (in this study *n* is 3)

α = typical dimension of element matrix.

The values of λ and ω, based on various evaluations, would have to be of the following order of magnitude: $10^{-3} > \lambda > 7\times10^{-8}$ and $10^{-2} > \omega > 10^{-4}$ [Golf-Racht, 1982].

In the double porosity model, the geometry of the fractures is determined by fracture spacing in the principal directions. This parameter indicates the special density of the fractures, essentially, the distance between two adjacent matrix blocks' centers. Large fracture spacing indicates reduced matrix-fracture fluid transfer. Conversely, small values of fracture spacing indicate high fracture density per block, consequently, large volume of matrix-fracture fluid transfer. According to Warren and Root formulation, the dimension of element matrix (α) is given as

$$\alpha = 4n(n+2)/l^2 \quad 3.4$$

Since the sugar cube model is assumed, *l* represents the fracture spacing. In other words, fracture intensity in x, y and z directions is constant.

3.3 Fault Characterization

Characterizing the fault parameters from pressure transient response in naturally fractured reservoirs is a challenging task, especially; permeability anisotropy of the fracture system complicates the well test analysis. However, the detectable fault response has its fingerprint on semi-log Horner plot where three straight lines reflect early, middle and late times respectively, separated by two transition periods. The first straight line represents the early time infinite-acting response in the fracture system. Once the fault starts to come into the picture, the transitions and the slopes of the remaining two straight lines are controlled not only by the formation parameters, but are also directly influenced by the distance to the fault (L_x), the height of the fault (h_F), and the effective width of the fault (l_F). A simple relationship was proposed by L.M. Yaxley in 1987 to describe the fault communication index (ε) as defined in following equation:

$$\varepsilon = \frac{k_F\, h_F L}{k_f\, h_f\, l_F} \tag{3.5}$$

Where (h_f) and (h_F) in Equation 3.4 are for extending the model for the case of unequal formation thickness on opposite side of the fault. Since the formation thickness is constant in this study, the equation reduces to the specific transmissibility of the fault to specific transmissibility of the fracture system, as shown in Equation 3.5.

$$\varepsilon = \frac{k_F L}{k_f\, l_F}, \qquad where\ L = 2 \times L_x \tag{3.6}$$

Figure 3.2 depicts the pertinent fault parameters discussed above [Al-Ghamdi, 1999].

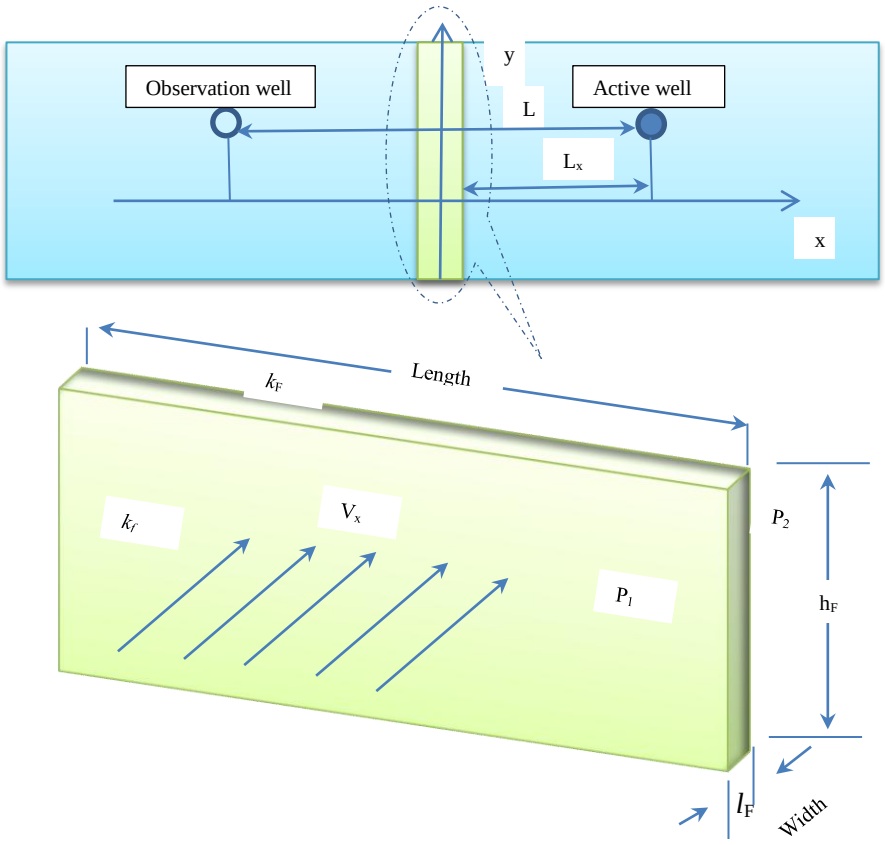


Figure 3.2: Representation of a semipermeable fault zone

3.3 Reservoir Proxies of Naturally Fractured Formation with a Fault

The forward solution part of the problem stated in chapter 2 achieved using the reservoir simulator, Adaptive Implicit-Explicit Black-oil Simulator (IMEX) developed by Computer Modeling Group (CMG). In order to investigate a wide range of naturally fractured gas reservoirs with various configurations and semi-sealing faults within limits of correspondent reservoir characteristics, a domain of combinations of formations are generated. Special attention has been given to those sets of formation properties and conditions to have the distinguishable features of a double porosity system where Horner plots of each case takes on three shapes between the limits of a reservoir with a totally sealing fault and a reservoir with no fault. While defining the ranges of different reservoir characteristics, huge emphasis is placed on physical accurability and solvability of each case.

All of the hypothetical cases under investigation involve a single production well, located in a single layer, dry gas, dual-porosity reservoir. The grid system is a 50×50 Cartesian grid system with the size of 10000×10000 ft^2 or 2295 acres, including refined grids blocks on the fault lines. Variations in the rock properties are created by random combinations which have uniform distribution, while their respective interrelationships and inter-correlations are kept.

In characterizing the rock fluid options in the reservoir simulator, the rock type is thought to be water wet, and a typical relative permeability curve is used in all cases. The curve is generated using Stone's Second Model (Stone, 1973) with an irreducible water saturation of S_{wi}=0.15. However, as mentioned in all of the hypothetical cases, formations are producing dry gas; therefore, water-gas contact is placed well below the bottom of the reservoir so that the water

never enters the well-bore. The relative permeability curves used in this study are shown in Figure 3.3.

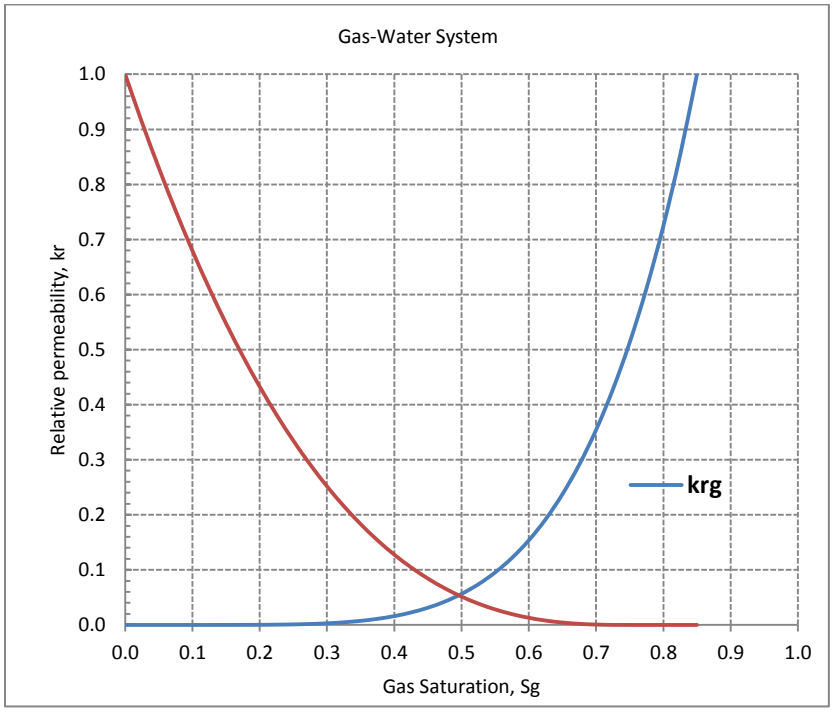


Figure 3.3: Two-phase water-gas relative permeability curves used in this study

3.4 Fluid Properties

Mathematical modeling of a given system requires the dependence of fluid properties on primary simulation unknowns (pressure and saturation). In black oil simulation, the tabulated PVT tables are needed to carry out the solution of mathematical material balance equations. The fluid properties of interest in simulating the gas reservoirs in CMG's black oil simulator include gas-compressibility factors (Z factors), formation volume factors (B_g), and gas viscosities (μ_g).

The physical properties of natural gases described above are important components for the evaluation of gas well performances. If temperature and pressure are known, other physical properties can be calculated based on physical laws. Thomas, Hankinson, and Phillips (1970), proposed analytical correlation to determine pseudo-critical properties where it is assumed that specific gravity of the gas is always available. In order to contract a PVT table for each case, a computer subroutine is developed to calculate the include gas-compressibility factors and viscosities at designated pressures and temperatures. In this program, the viscosity calculation is based on Carr, Kobayashi, and Burrows' viscosity ratio charts. Basically, combining two charts, viscosity ratio versus the pseudo-reduced temperature and viscosity ratio versus the pseudo-reduced pressure and turning those charts into a viscosity ratio surface in a space of viscosity ratio, pseudo-reduced pressure and temperature can be tabulated, and then gas viscosity can be calculated at atmospheric pressure and reservoir temperature using gas specific gravity. On the other hand, gas-compressibility factors are calculated using numerical approximations by Dranchuk and Abu-Kasem (1975).

The required initial computational investment is done to provide training data for ANNs. In typical reservoir modeling studies, detailed attention is paid to setting the simulation parameters

and the analysis of results on a case-by-case basis. While in generating training data, both the creation of simulation parameters and the collecting of results are automated. Given the ranges of simulation parameters and rules of parameter combinations to simulate, a data file generator for reservoir simulator is created to tailor executable input files for each run and launch the simulation automatically in batches, and collect the information.

The reservoirs are thought to be infinite acting within the duration of entire production period. The well is designed to be vertical, and located at the center of the grid, and the minimum bottom-hole pressure is set to be 14.7 psia. For the purpose of batch processing massive amount of data that are consistent with naturally fractured reservoirs' signature and theoretical basis, computer programs were written to build the simulator model and extract results. As intricate parts of fluid properties, methods for obtaining gas viscosity and Z factor are given in Appendices B and C, and computer programs for reservoir data acquisition were done in nine subsequent subroutines:

1. Viscosity Interpolation Subroutine
2. Gas Compressibility Calculation Subroutine (Z factor)
3. Reservoir Proxy Generator
4. Compressibility of Gas
5. Production Rate Specification using ψ approach
6. PVT table for CMG inputs
7. Tabulation of generated reservoir data
8. Reservoir data into reservoir simulator
9. Physical accurability checker

The development procedures of these subroutines are given in the Appendix D.

Chapter 4

ANN Development Stages

As has been idicated in first chapter, Artificial Neural Networks (ANNs) are a powerful predictive tool in the field of petroleum engineering. Presented here is the step by step development of network architecture using large number of well pressure data, gas production rate, properties of rock media and gas. The basic guideline is to start with small network architecture, and slowly increase the complexity from low-dimensional output space to high-dimensional output space.

Subject of this study started out with homogenous isotropic reservoirs. Later, the experiences gained from isotropic systems applied to anisotropic reservoirs. Development of ANNs consists of three main stages, and each stage is broken down to more manageable small steps. The complexity of the network parameters slightly rose in each step to increase the robustness, and accuracy of the network.

The philosophy that we adopted here is to include all input parameters that have possible influence on the model outputs regardless of their contribution factors, and to predict as many outputs as possible while keeping the errors within acceptable tolerance range. The rest of the work is left to the ANN to determine which inputs are significant. Since the bottom-hole pressure under constant flow rate can be monitored constantly for any given time, it is the most influential input parameter among others. However, initial formation pressure, compressibility (fracture, matrix, and gas), formation thickness, initial viscosity, and gas gravity, are the parameters that are available to us, and they distinguish particularity of a reservoir from a domain of naturally fractured reservoirs. The influential parameters that are varied to construct proxies of reservoirs

are presented in Table 4.1. The Table 4.2 shows the main steps in the development of ANN models.

Table 4.1: Ranges of reservoir characteristics		
Parameters	Minimum	Maximum
Thickness, ft	20	200
Porosity-matrix	0.01	0.15
Porosity-fracture	0.0001	0.005
Permeability-matrix, md, I, j, k	0.000001	0.0001
Permeability-fracture, md, I, j, k	0.001	0.1
kfi/kfj, (anisotropy)	0.05	1
Permeability-fracture, md, k, (anisotropy)	0.00000001	0.00000001
Permeability-matrix, md, k, (anisotropy)	0.000000001	0.000000001
Fracture spacing, ft, I, j, k	20	150
Reservoir temperature, ^{0}F	100	200
Initial reservoir pressure, psi	1500	5000
Earth pressure gradient, psi/ft	0.55	1
Specific gravity of gas	0.56	0.9
Compressibility of gas, 1/psi	0.000063	0.0007
Compressibility of matrix pore, 1/psi	2.00E-06	1.50E-05
Compressibility of fracture pore, 1/psi	7.20E-05	9.60E-05
Gas production rate, mmscf/day	0.01	9.5
Distance to the fault, ft	300	2000
Sealing capacity, %	70	100
Orientation of the fault, dgrees	30	120
Fault communication index	0	10
Inter-porosity flow parameter, λ	10^{-3}	10^{-9}
Storage capacity, ω	10^{-2}	10^{-4}
Fault communication index, ε	0	30

Table 4.2: Development stages of ANN		
Inputs:	Stages	Outputs
$p_i, \frac{dp}{d\ log_{10}(t)}\ or\ \psi_i, \frac{d\psi}{log_{10}(t)}$ c_m c_f c_g $c_f + c_m + c_g$ γ_g μ_i h q_{sc}	Stage I Homogenous Isotropic reservoirs	1. Matrix porosity, φ_m 2. Fracture porosity, φ_f 3. Matrix permeability, k_m 4. Fracture permeability, k_f 5. Inter-porosity flow parameter, λ 6. Storage capacity, ω 7. Fracture spacing, l 8. Fault communication index, ε 9. Sealing capacity of the fault, SC 10. Distance to the fault, Lx
	Stage II Homogenous Isotropic reservoirs	1) k_f, k_m, φ_f, φ_m, ω, λ, l 2) L_x, SC 3) k_f, k_m, φ_f, φ_m, ω, λ, l, L_x, SC, ε
	Stage III Homogenous anisotropic reservoirs	k_{fx}/k_{fy}, k_{fx}, k_{fy}, k_m, φ_m, φ_f, l, L_x, ε, SC, θ, ω, λ

4.1 Stage I

During the course of this research, the Matlab neural network toolbox extension was used because a wide variety of networks are built into the software, and network parameters can be adjusted at will. The main advantage of the neural network toolbox is that it has been improved over the years to encompass the latest high performance algorithms.

This stage is the most important part of the network building. We started with concluding the size of the data set necessary for accomplishing an accurate training. Apparently, the more data we have the better the performance of the network. However, considering the time constraints for generating large number of cases and simulating them with reservoir simulator, it is advisable to resort to successful ANNs of other researchers. There is an approximate relationship between the number of decision variables and the number of examples required for ANN training and testing which is derived from the ANN rule of thumb that between 5 to 10 learning examples be given for each ANN network weight [M. Anderson et al, 2000]. Initially, it is decided to generate 200 cases, later, it was proven to be insufficient and 1000 cases in total were generated to widen the reservoir proxies. As we established the development protocol for Stage I shown in Table 4.2, a single output neuron for each output is tested individually. In order to keep consistent data points for collecting bottom-hole pressure generated by the simulator, 280 time points are selected and extracted in all cases within 10 years under constant production rate. The reason for choosing so many data points is that the production period is very long, and the effect of dual porosity, and sealing or partially communicating fault can be felt from the bottom-hole pressure could come into picture at different times for each case, given the different values for permeability, porosity, distance to the fault, and sealing capacity, and communication index of the fault in the system. In order to make a fault line in the reservoir simulator for an isotropic reservoir, grid blocks crossed

by the fault line were refined into smaller blocks. The complete or semi-sealing feature of the fault was modeled by adjusting permeability of those grid blocks accordingly as demonstrated in Figure 4.1.

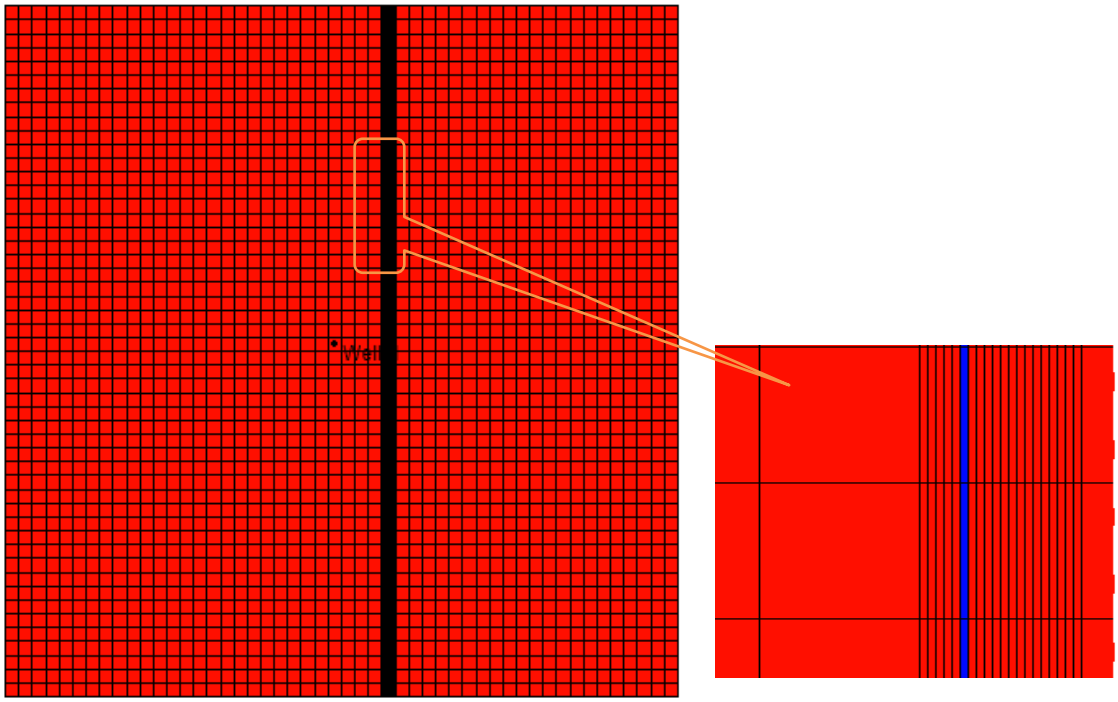


Figure 4.1: Fault line in isotropic reservoir modeled in CMG with grid refinement

Table 4.3 marks the times at which the bottom-hole pressures are collected, and the examples of bottom-hole pressure collected along those points are shown in Figure 4.2. Another reason for choosing the data points so densely is to reserve a possibility of shrinking the data by selecting the points in long intervals.

Table 4.3: Times at which the bottom-hole pressures are collected (Day)									
0.0006944	1	29	165	305	530	810	1110	1950	2790
0.0076389	2	30	170	310	540	820	1140	1980	2820
0.0145833	3	35	175	315	550	830	1170	2010	2850
0.0215278	4	40	180	320	560	840	1200	2040	2880
0.0284722	5	45	185	325	570	850	1230	2070	2910
0.0354167	6	50	190	330	580	860	1260	2100	2940
0.0833333	7	55	195	335	590	870	1290	2130	2970
0.125	8	60	200	340	600	880	1320	2160	3000
0.166667	9	65	205	345	610	890	1350	2190	3030
0.208333	10	70	210	350	620	900	1380	2220	3060
0.25	11	75	215	355	630	910	1410	2250	3090
0.291667	12	80	220	360	640	920	1440	2280	3120
0.333333	13	85	225	370	650	930	1470	2310	3150
0.375	14	90	230	380	660	940	1500	2340	3180
0.416667	15	95	235	390	670	950	1530	2370	3210
0.458333	16	100	240	400	680	960	1560	2400	3240
0.5	17	105	245	410	690	970	1590	2430	3270
0.541667	18	110	250	420	700	980	1620	2460	3300
0.583333	19	115	255	430	710	990	1650	2490	3330
0.625	20	120	260	440	720	1000	1680	2520	3360
0.666667	21	125	265	450	730	1010	1710	2550	3390
0.708333	22	130	270	460	740	1020	1740	2580	3420
0.75	23	135	275	470	750	1030	1770	2610	3450
0.791667	24	140	280	480	760	1040	1800	2640	3480
0.833333	25	145	285	490	770	1050	1830	2670	3510
0.875	26	150	290	500	780	1060	1860	2700	3540
0.916667	27	155	295	510	790	1070	1890	2730	3570
0.958333	28	160	300	520	800	1080	1920	2760	3600

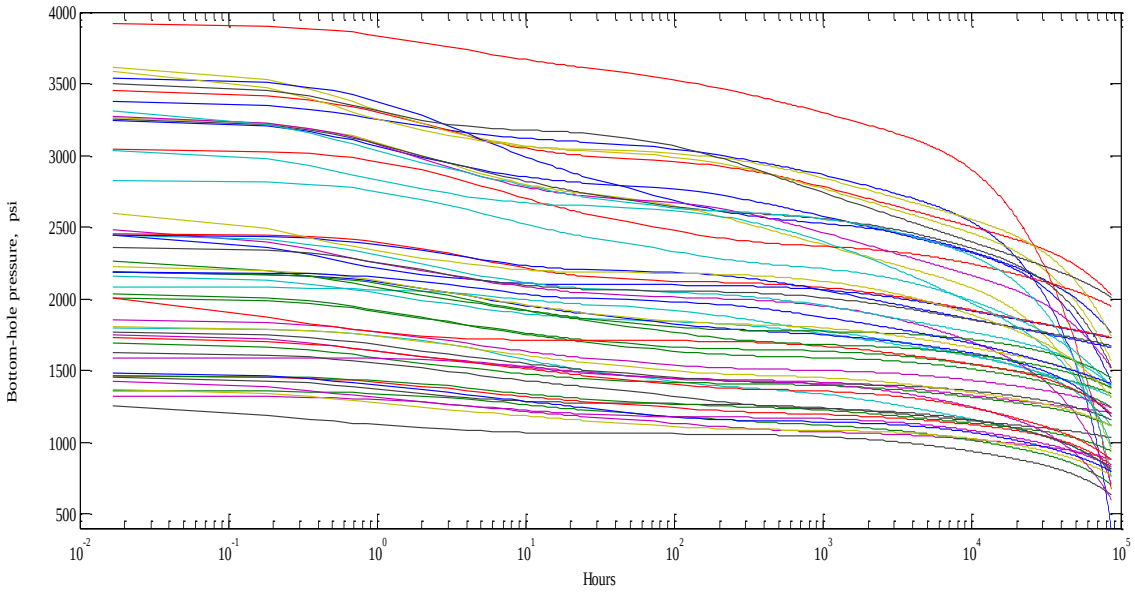


Figure 4.2: Horner plot of 50 example cases of isotropic reservoirs

At the beginning of the Stage I, two methods of presenting pressure transient data have been tried. As a primary method of pressure value representation in the input vector, pressure values were changed into slopes of pressure against logarithm of time using function 4.1.

$$\frac{dp}{d\ log_{10}(t)}=\frac{p_{n+1}-p_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)} \tag{4.1}$$

where n=1, 2, 3… 279.

Later, pseudo-pressure (ψ) approach was tried where ψ was calculated by numerical integration of function 4.2.

$$\psi = \int_0^p \frac{p}{\mu Z} dp \tag{4.2}$$

The slopes were calculated as:

$$\frac{d\psi}{d\ \log_{10}(t)}=\frac{\psi_{n+1}-\psi_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)} \tag{4.3}$$

where *n* is the same as in Equation 4.1.

The idea was to incorporate the viscosity and the gas compressibility factor at different pressure values into input data. The simple-pressure and pseudo-pressure derivatives against logarithm of time are compared in Stage II.

An automatic script generator was used, first, to construct a basic architecture in order to take advantage of customization with minimal effort. Other features of the neural network toolbox software are exploited using command-line operations. This basic architecture was trained without normalized data, and soon, it was evident that the pre-processing of data is vital to the network to obtain the desired level of success. The data pre-processing includes normalization, variable combination, and the order of variables. There are several neural network toolboxes' built-in processing functions, among which, *mapminmax* is the standard input output normalization function. It processes matrices by mapping row minimum and maximum values to [-1 1]. Furthermore, it was found that scaling of input and output variables greatly improved the

estimation error and calculation time needed in training process. The underlying theory is to compress all the weight and bias searching space to a unitary hypercube, to reduce the distance to be covered by the selected training algorithm. Conversely, if scales are dissimilar in the order of magnitude, the error reduction algorithm (training algorithm) will focus on the high-valued variables. Consequently, those variables will have a higher contribution to the output error. Other than scaling, there are other data pre-treatment methods which are not part of neural network toolbox, namely, centering and data transformation, which also have been tried in pre-processing of the data. For example, removing the average value from a specific variable in order to focus on difference rather than similarities in the data, log transformation, power transformation etc. The overview of the data pre-treatment methods mentioned above is provided in Appendix A.

In terms of transfer functions, it was observed that *tansig* ($f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$), *logsig* ($f(x) = \frac{1}{e^x + e^{-x}}$), and *purelin* were better choices for the layers; the combination of *tansig, logsig* activation functions were tried in hidden layers, by default; *purelin* is used in the output layer. After experimenting with all the available training algorithms in neural network toolbox, it was identified that *Levenberg-Marquardt back-propagation (trainlm), scaled conjugate gradient back-propagation (trainscg), and resilient back-propagation (trainrp), gradient descent with momentum and adaptive learning rule back-propagation (traingdx)* are the potential candidates for training algorithms. Further analysis of the network architecture revealed that *trainlm and trainscg* outperform the other training algorithms, yet, we reserve the use of other algorithms in stages II and III for the purpose of compairing the performances in different situations.

In this section, the goal is to structure ANNs to predict the following nine outputs' performances separately for any given combination of the input variables presented in Figure 4.3. The results of

this stage are important for future analysis aimed at growing complexities in stages II and III, such as predicting more outputs with single architecture, focusing on variables that are hard to predict.

The proposed network is composed of 3 hidden layers with 100, 50 and 5 neurons respectively. The hyperbolic tangent sigmoid (*tansig*) and liner (*purelin*) functions were used as transfer functions for each layer as shown in Figure 4.3. The scaled conjugate gradient training algorithm (*trainscg*) and Levenberg-Marquardt training algorithm *(trainlm)* were implemented as back-propagation techniques.

It is important to point out those variations of Network A that have been tried. The modifications are made to the hidden layers and transfer functions. For example, the three hidden layer configuration of 60 (*tansig*) ×30 (*tansig*) ×5 (*tansig*) and 80 (*tansig*) ×40 (*tansig*) ×5 (*tansig*) were tested, and the *tansig* and *logsig* combinations were also used. It was observed that the 100 (*tansig*) ×50 (*tansig*) ×5 (*tansig*) hidden layer combination produces the most desirable results. It is also important to mention that, in the process of predicting output variables one by one, the same network architecture was used for every single output in order to avoid confusion, and the purpose was to identify the predictability of each output variable. All prediction results shown in Figure 4.4 are summarized in Table 4.4. Once the parameters are isolated in terms of their predictability, in Stage II, we will focus on the parameters that are very difficult to predict.

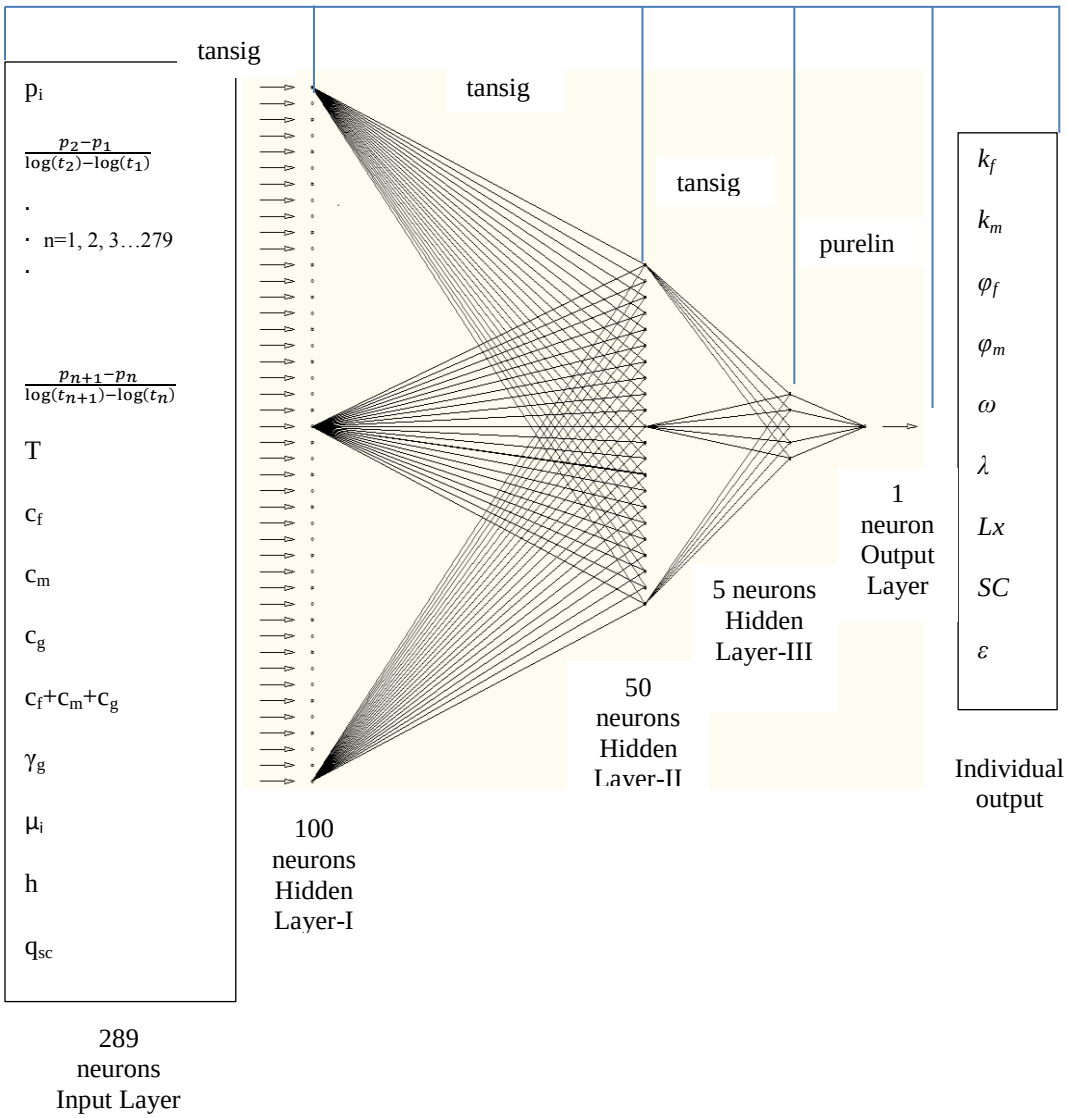
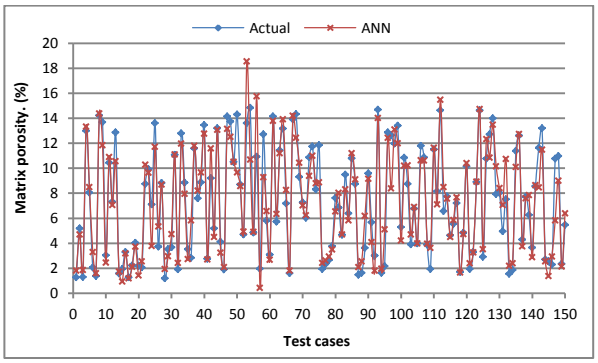
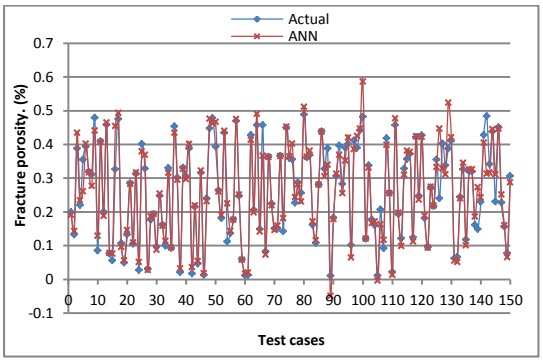
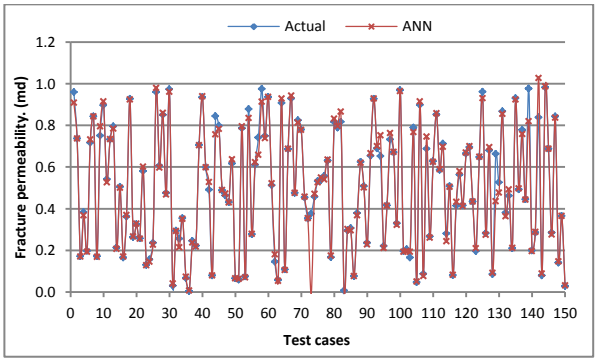
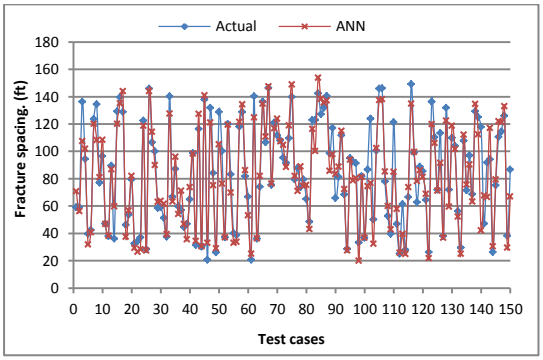


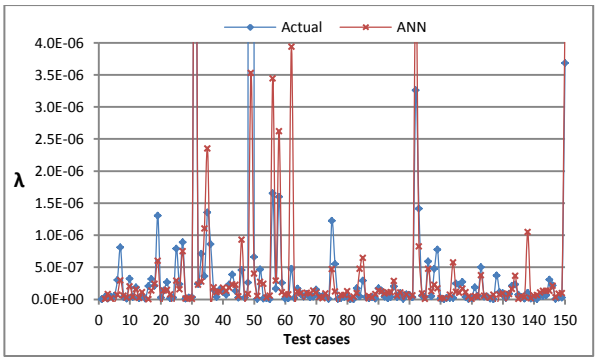
Figure 4.3: Network architecture with single output neuron (network A, Stage I)

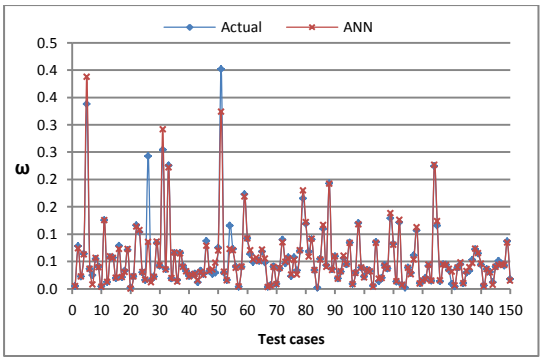

Actual
ANN
Matrix porosity. (%)
Test cases

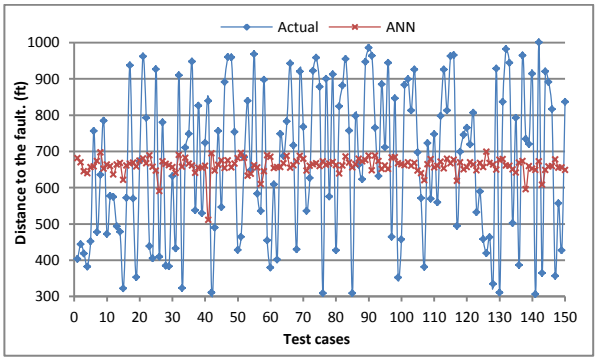

Actual
ANN
Fracture porosity. (%)
Test cases

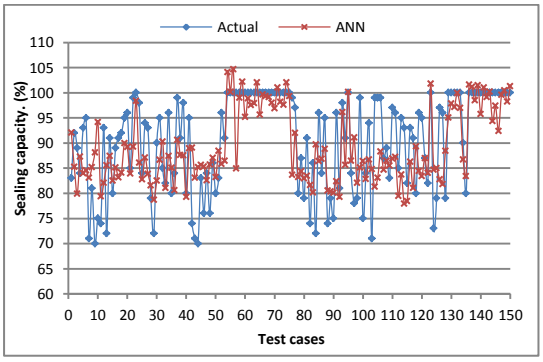

Actual
ANN
Fracture permeability. (md)
Test cases


Actual
ANN
Fracture spacing. (ft)
Test cases


Actual
ANN
λ
Test cases


Actual
ANN
ω
Test cases


Actual
ANN
Distance to the fault. (ft)
Test cases


Actual
ANN
Sealing capacity. (%)
Test cases

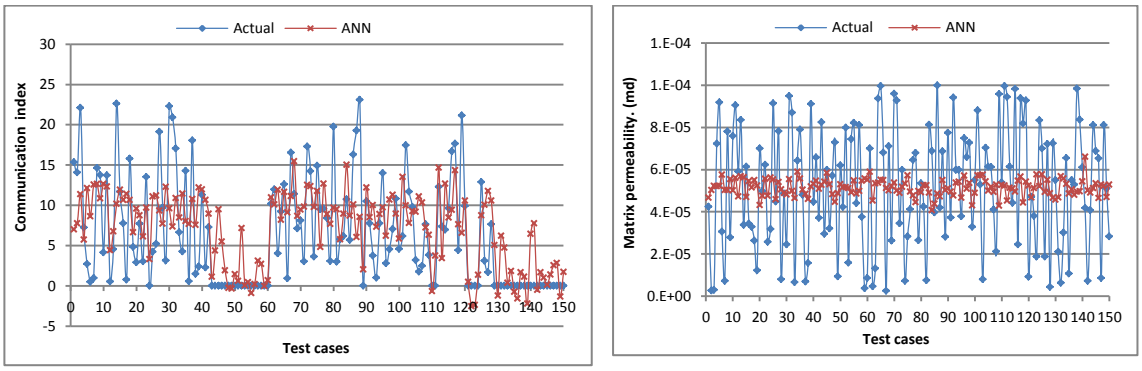


Figure 4.4: All output predictions generated by network A architecture (Stage I)

Table 4.4: Predictability of parameters	
Easy	Difficult
Matrix porosity, φ_m	Distance to the fault, Lx
Fracture porosity, φ_f	Sealing capacity of the fault, SC
Fracture permeability, k_f	Fault communication index, ε
Fracture spacing, *l*	Matrix permeability, k_m
Storage capacity, ω	Inter-porosity flow parameter, λ

Although the trained nets' error goal can be set to any desired small number, the large magnitude of difference in different types of output variables decrease the prediction capability of the networks. Moreover, higher deviations were observed near lower boundaries of small variables. This implies that the prediction capabilities of the networks are sensitive to the smallest values of small variables. If the scales are very dissimilar for different values, the bigger ones will have a higher contribution to the output error, and the error reduction algorithm (training algorithm) will be focused on the variables of higher values. Luckily, there are built-in data normalization functions in the neural network toolbox. However, it is better to normalize the data by bringing all variables into the same order of magnitude before presenting them to the toolbox. In other words, the inputs and outputs are normalized twice; I'd like to call the first normalization outside the neural network toolbox is called "magnitude equalization", and the second normalization inside the neural network toolbox is referred to as "default normalization". In terms of variables' magnitude equalization, the small variables, such as, inter-porosity flow parameter, matrix permeability, matrix permeability, compressibility of matrix pore, and compressibility of fracture pore are multiplied by large numbers to increase the their contribution. The summary of magnitude equalization is given in Table 4.5.

As part of the default normalization, *mapstd* and *processpca* are also tried. After several training procedures, it was obvious that those two normalization functions cannot give desired error reduction; instead, they confuse the network.

At this stage, the size of each input vector consists of 289 elements where it includes 279 slopes of pressure against logarithm of time plus ten other reservoir, rock, and gas characteristics. Considering the large number of training examples and the size of the input vectors, using Levenberg-Marquardt training algorithm *(trainlm)* will need more memory allocation and training

time to converge. Instead, the scaled conjugate gradient training algorithm (*trainscg*) is the logical choice for obtaining all the results presented in Figure 4.4.

After constructing the acceptable network with a single neuron on the output layer, the prediction accuracy of the network for each output variable was identified. Provided that some of the output variables are difficult to predict accurately as a single target, large error values can be observed when predicting the matrix permeability, sealing capacity of the fault, distance to the fault, and fault communication index. Undesirable outcome of those variables is sensitive to transfer functions and hidden layers' size (numbers of neurons in the hidden layer). Even more striking is the fact that even with more hidden layers, increased number of hidden neurons, and robust normalization method the situation does not improve. This brings us to the Stage II where we investigate inter connections among output variables.

Table 4.5: Summary of the magnitude normalization

Outputs	Units	Scaling factor
Fracture porosity, φ_f	%	
Fracture permeability, k_f	md	*100
Matrix permeability, k_m	md	*1e6
Distance to the fault, Lx	ft	/10
Inter-porosity flow parameter, λ		*1e7
Storage capacity, ω		*100
Fault communication index, ε		
Sealing capacity of the fault, SC	%	
Fracture spacing, *l*	ft	
Matrix porosity, φ_m	%	
Inputs		
Specific gravity of gas, γ		*100
Compressibility of gas, c_g	1/psi	*1e7
Compressibility of matrix pore, c_m	1/psi	*1e7
Compressibility of fracture pore, c_f	1/psi	*1e7
$c_g + c_m + c_f$	1/psi	*1e7
Initial gas viscosity, μ_i	cp	*1000

4.2 Stage II

After analyzing prediction precisions of each and every variable, the “troublesome” variables were isolated. The problem is mainly because of fault interference in the system. Given the reservoir is under constant production rate, theoretically, there are almost infinite number of sealing capacity and distance to the fault combinations that can produce an exact same bottom-hole pressure profile, which, in turn, makes it hard for the network to predict the fault characteristics, namely, sealing capacity and distance from the well. Another important feature is that the double-porosity gas reservoirs in this study have tight matrix porosity and matrix permeability is in nano-darcy range. Basically, predicting small values with high precision is difficult.

In this stage, the degree of nonlinearity was increased along with the increased number of outputs and the level of complexity. In order to accommodate more outputs, first, we resorted to the underlying physics of the problem to provide the network with more information by including analytical expressions of some variables. Second, the interconnectivity of the target variables were exploited, also more connection between input and output variables were provided. The third option was to incorporate modified outputs as functional links. The final goal remains finding a generalized network to predict the multiple outputs as accurately as possible with single network architecture.

4.2.1 Input and Output Modifications

Following the argument presented above, it is decided to modify the variables in order to make it possible for the network to distinguish ambiguous variables. As an intricate part of the input

vector, bottom-hole pressure profile sustains the main body of input variables. The slopes of pressure against logarithm of time in Equation 4.1 were transformed into slopes of pseudo-pressure against logarithm of time shown in Equation 4.3 based on pseudo-pressure (ψ) approach. Each ψ value in Equation 4.2 at a time point with corresponding pressure value was calculated by numerical integration. μ and Z vales for each pressure and temperature (it was assumed that reservoir temperature is constant) values were calculated using a subroutine developed in the process of reservoir data generation. The idea was to incorporate the viscosity and the gas compressibility factor at different pressure values into input data. Details of this subroutine are included in Appendix D. The pseudo-pressure further transformed into modified dimensionless pseudo-pressure is

$$\psi_D^i = \frac{h\,\psi_{wf}}{1.422 \times 10^6\, T\, q_{sc}} \tag{4.4}$$

The slopes of modified dimensionless pseudo-pressure drop against logarithm of time is calculated as

$$\frac{d\psi_D^i}{d\,log_{10}(t)} = \frac{h(\psi_n - \psi_{n+1})}{1.422 \times 10^6\, T\, q_{sc}(log_{10}(t_{n+1}) - log_{10}(t_n))} \tag{4.5}$$

where n=1, 2, 3 … 279.

This modification bounds $\frac{d\psi_D^i}{dt}$ values within a range of [0, 800], while incorporating viscosity, gas compressibility factor, formation thickness, reservoir temperature, and production rate. When being normalized into the interval [-1, 1] this modification helped to increase the prediction accuracy.

Another important group of parameters indicating the uniqueness of a reservoir is also presented several times in different forms in order to increase their impact on the network. It was observed that the rock and gas compressibility (c_m, c_f, c_g), initial viscosity (μ_i), and specific gas gravity (γ) can be arbitrarily raised to powers to correct for heteroscedasticity of the small values. Initial

reservoir pressure (p_i), modified dimensionless initial pseudo-pressure (ψ^i_{Di}), formation thickness (h), and production rate (q_{sc}) were presented in inverted forms coupling with their original values to emphasize their importance.

As for the output variables, magnitude equalization introduced in Table 4.5 is retained throughout Stage II. Additionally, logarithmic transformation is tried on the product of sealing capacity and distance to the fault, $\log_{10}(L_x \times SC)$, where the log transformation also corrects for heteroscedasticity and makes the multiplicative modes additive.

After many attempts with above mentioned transformation methods, it was advised to include eigenvalues of closely related output variables. We have decided to lump four most influential output variables together to decrease prediction errors even further in the presence of other output variables.

$$Eigenvalues1 = eig\begin{bmatrix} k_m & \varphi_m \\ k_f & \varphi_f \end{bmatrix} \tag{4.6}$$

Eigenvalues of the hard-to-predict variables, L_x and SC coupled with two other easy-to-predict variables, φ_m and k_f, were introduced to increase their predictability.

$$Eigenvalues2 = eig\begin{bmatrix} L_x & SC \\ \varphi_m & k_f \end{bmatrix} \tag{4.7}$$

In terms of outputs' analytical expression, the following modified dimensionless time function was devised to cope with outputs when using ψ on the input side.

$$t_D^i = \frac{0.02637\ k_f}{\left(\varphi_m c_m + \varphi_f c_f\right) \mu_i\ r_w^2\ L_x\ SC} \tag{4.8}$$

In fact, all the discussions on input and output modifications are part of data transformation methods. In this study, those applied modifications are referred to them as functional links. Data transformations tried in different steps of networks with multiple outputs were summarized in

Table 4.6, and the theoretical bases for transforming the data are summarized in Appendix A. It is important to note that power transformations and magnitude equalizations on both input and output variables were determined heuristically.

Table 4.6: Summary of functional links used in Stage II	
Inputs	$\gamma_g^{0.5}$, $\gamma_g^{0.05}$ $c_f^{0.01}$, $c_f^{0.1}$ $c_m^{0.01}$, $c_m^{0.1}$ $c_g^{0.01}$, $c_g^{0.1}$ $(c_f+c_m+c_g)^{0.01}$, $(c_f+c_m+c_g)^{0.1}$ $\mu_i^{0.75}$, $\mu_i^{1.5}$, μ_i^{2} $\frac{d\psi_D^i}{dt}$ ψ_{Di}^{i} q_{sc}^{-1}, $q_{sc}^{0.5}$
Outputs	$k_f \times L_x/10$ $\varphi_m \times L_x/100$ $l \times L_x/1000$ $k_f \times SC$ $\varphi_m \times SC/10$ $l \times SC/100$ $k_f \times k_m \times 10^6$ $\varphi_m \times k_m \times 10^5$ $l \times k_m \times 10^4$ $\log_{10}(L_x \times SC)$ t_D^i $eig\begin{bmatrix} k_m & \varphi_m \\ k_f & \varphi_f \end{bmatrix}$ $eig\begin{bmatrix} L_x & SC \\ \varphi_m & k_f \end{bmatrix}$

4.2.2 Network Architectures with Multiple Outputs

At this point, the output sets used in the previous stage were merged. The first step in merging different outputs is to predict k_f, k_m, φ_f, φ_m, ω, λ, *and* l, while keeping equalized magnitude for outputs suggested in Stage I, Table 4.5, namely, $\varphi_f * 10^2$, $k_f * 10^2$, $km * 10^5$, $\omega * 10^2$, $\lambda * 10^7$. The results of those output variables are important for predicting other variables aimed at incorporating new complexities into present network architecture. The network development was started with three hidden layered network structure–100 (*tansig*) ×50 (*tansig*) ×7 (*purelin*). After several training attempts, it was evident that multiple outputs require more hidden neurons. Subsequently, three hidden layer configurations with more hidden neurons were tried. When tested with 200 (*tansig*) ×100 (*tansig*) ×7 (*purelin*) structure shown in Figure 4.5, the results were promising. The performance of the selected network is greatly improved except for the matrix permeability.

Figure 4.6 presents network predictions for data sets that the network has not seen. In this step and following two subsequent steps, data sets generated during Stage I of this study are utilized in training and testing of networks with multiple outputs. After the training process was completed, the outputs are converted back to their original magnitude.

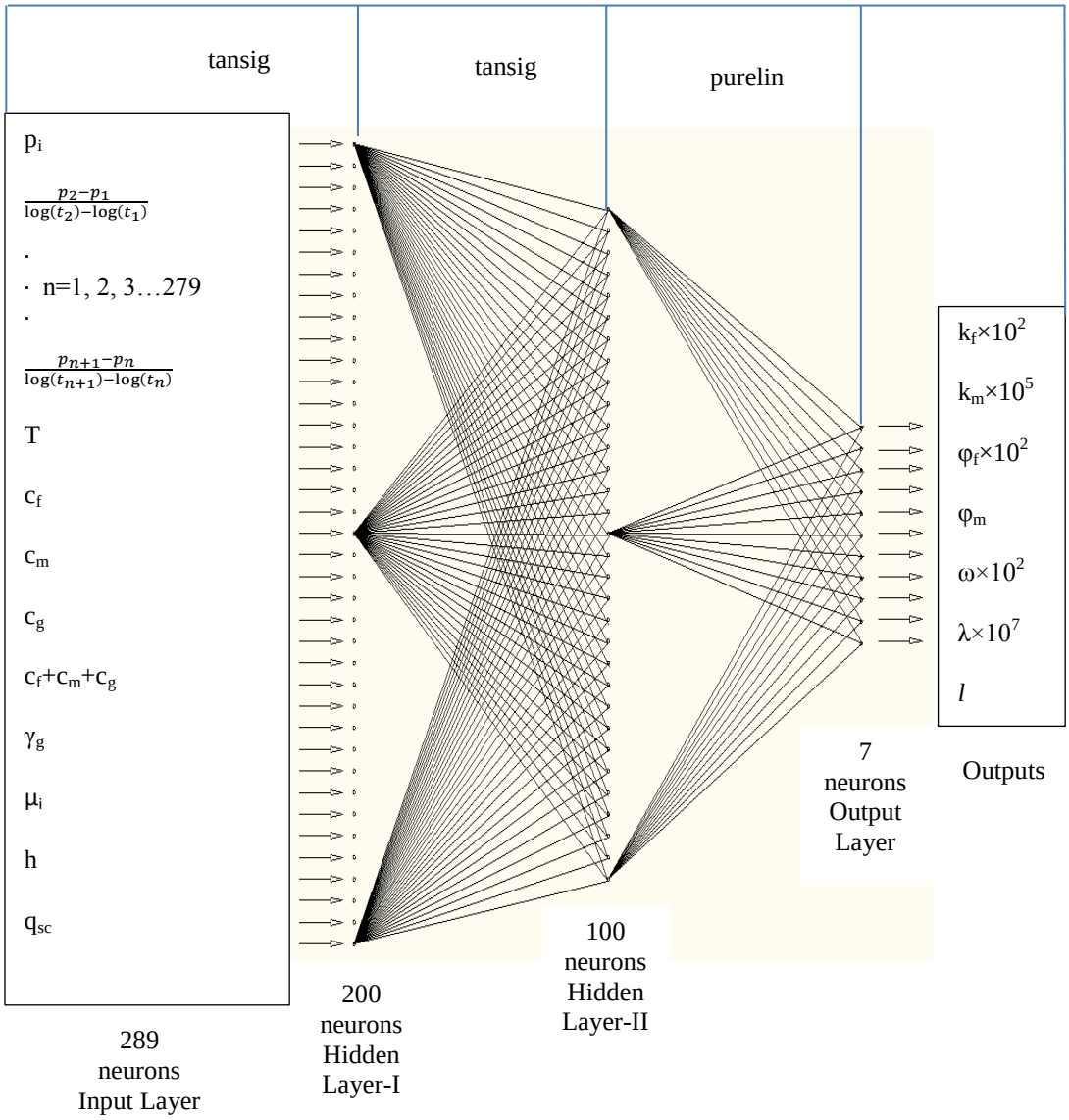


Figure 4.5: Network architecture with 7 output neurons (network B, Stage II)

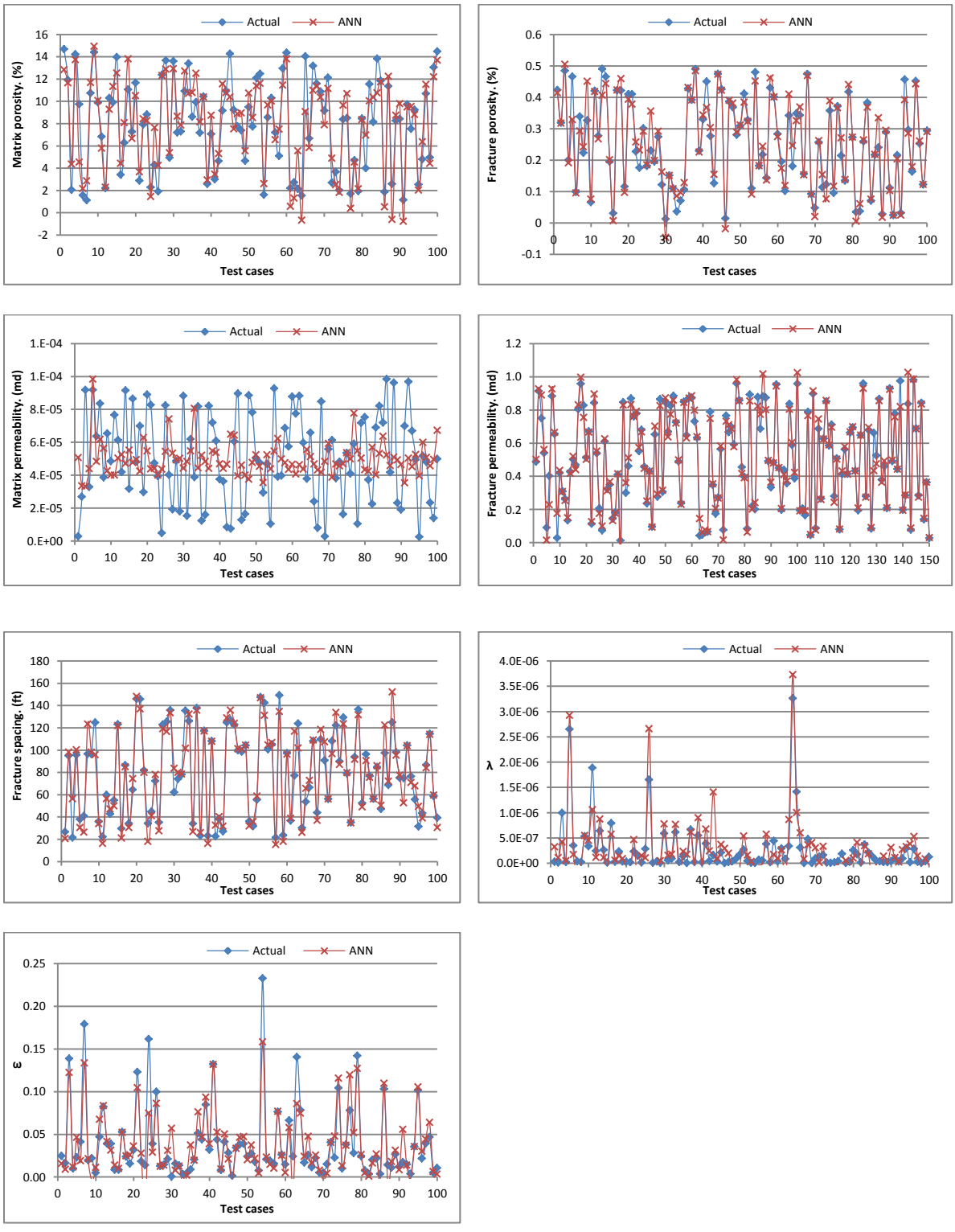


Figure 4.6: Training results using network B architecture (Stage II)

The second step of Stage II is to evaluate distance to the fault, sealing capacity, and communication index as a group. As it is designed for predictions made in the previous step by network B (Figure 4.5), network C (Figure 4.6) was designed to find out most effective functional links for distance to the fault and sealing capacity. In order to decrease ambiguity of those variables, reliable output predictions, namely, matrix porosity, fracture permeability and fracture spacing were selected from network B and multiplied with fault characteristics accordingly. Scaled functional links, $k_f \times L_x/10$, $\varphi_m \times L_x/100$, $l \times L_x/1000$, $k_f \times SC$, $\varphi_m \times SC/10$, $l \times SC/100$, $k_f \times k_m \times 10^6$, $\varphi_m \times k_m \times 10^5$, $l \times k_m \times 10^4$, and $\log_{10}(L_x \times SC)$ were tested together with k_f, φ_m, and l as a group of outputs. The network constructed to distinguish the reliable functional links is illustrated in Figure 4.8. After training is done, when L_x, SC, and k_m were transformed from corresponding functional links back to their original forms, the prediction errors ($Error_{Lx} = | L_{x\_actual} - L_{x\_ann}|/L_{x\_actual} *100$, $Error_{SC} = | SC_{actual} - SC_{ann}|/SC_{actual} *100$, $Error_{km} = | km_{actual} - km_{ann}|/km_{actual} *100$) are plotted on top of each other (see Figure 4.8). By comparing test-error plots of each functional link, it was determined that the most effective ones among them are the scaled product of sealing capacity and distance to the fault with fracture spacing, matrix permeability with fracture permeability. It was evident that the functional links, $l \times L_x/1000$, $l \times SC/100$, and $k_f \times k_m \times 10^6$ can help to predict those ambiguous variables.

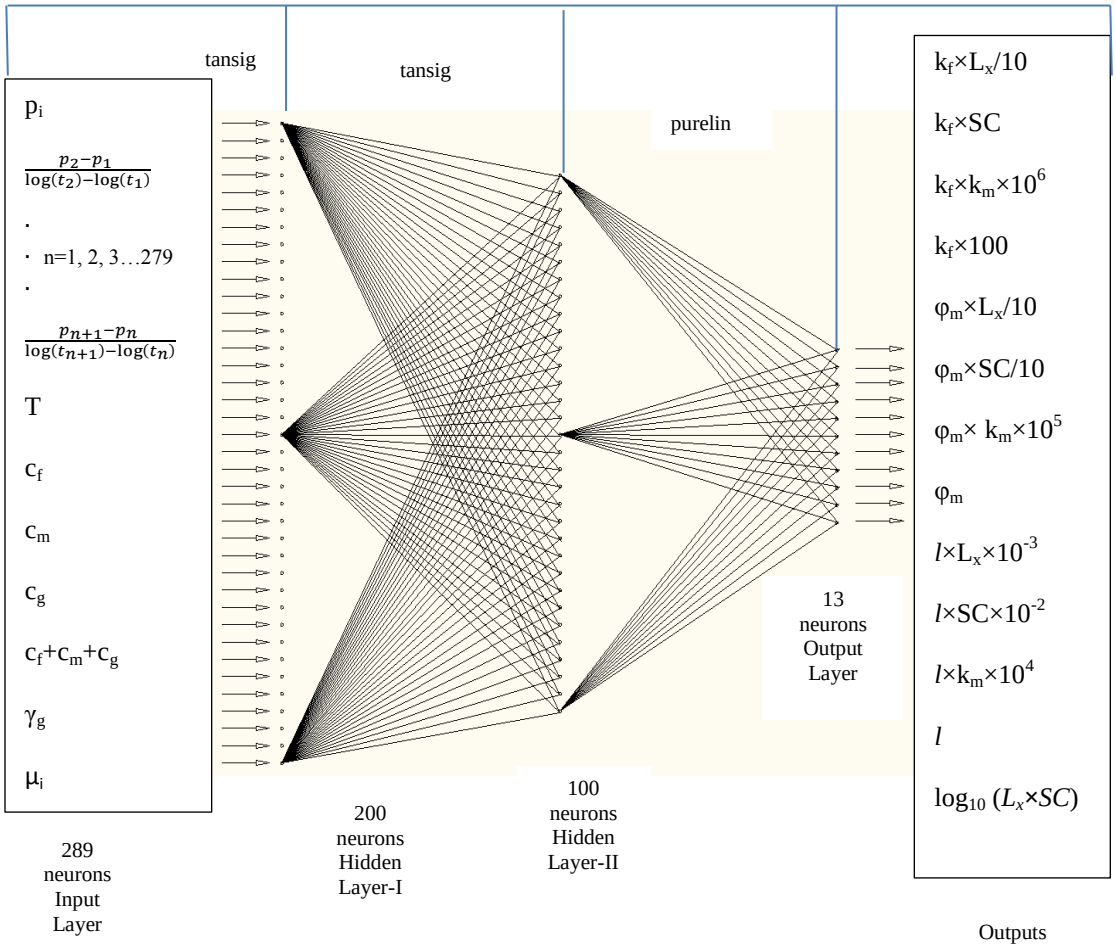


Figure 4.7: Network architecture with 10 output neurons (network C, Stage II)

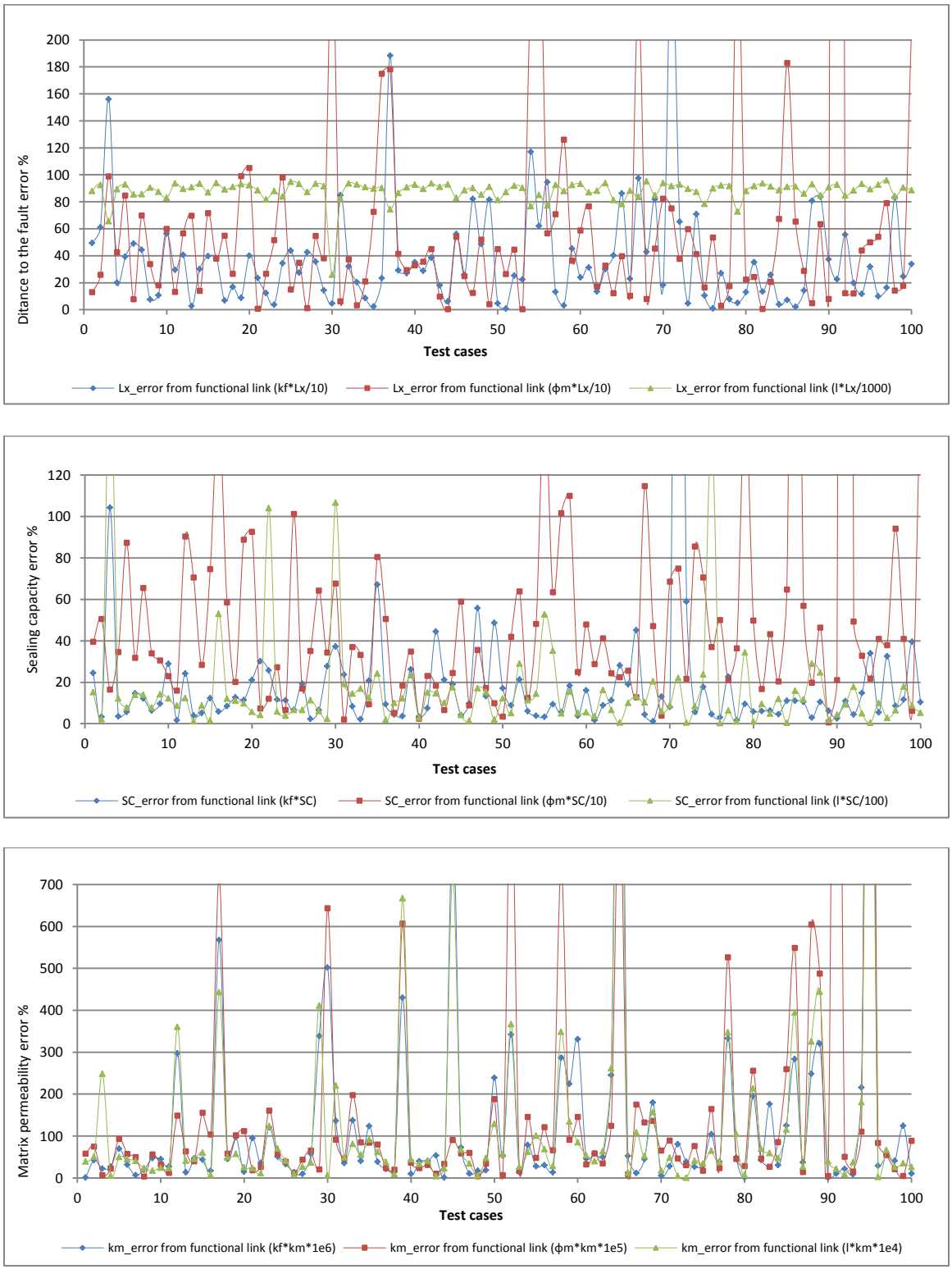


Figure 4.8: Predictability of Lx and SC with functional links

Until this point, slopes of pressure against logarithm of time were used as major components of input vectors to utilize in networks A, B and C. The most modifications done on output variables are scaling the magnitude and selecting reliable functional links. Taking advantage of the knowledge gained from previous sections, in the final step of Stage II, more complex networks were constructed to predict all ten variables at once. Complex network structures aside, all possible input- output modifications, transformations, and functional links discussed in previous section were utilized. The goal is to evaluate the competence of the network in accommodating complicated scenarios when nonlinearity is increased along with the increased number of outputs.

On the basis of *p* approach and ψ approach, modified rock and fluid characteristics into the inputs side were combined with all successful scaling factors, functional links, and additional modifications to increase the prediction accuracy; networks D, E, F, and G were constructed to validate augmented input-output pairing, and to explore more possibilities. The *p* approach was used in networks D and E to obtain slopes of pressure against logarithm of time, $\frac{p_{n+1}-p_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)}$, ψ approach was used in network F and G to obtain slopes of modified pseudo-pressure against logarithm of time, $\frac{\psi^i_{Dn}-\psi^i_{Dn+1}}{\log_{10}(t_n)-\log_{10}(t_{n+1})}$.

In networks D and F, the matrix rock compressibility, fracture rock compressibility, gas compressibility, total compressibility, gas gravity, gas viscosity, and initial reservoir pressure were raised to different magnitude – c_m*10^7, c_f*10^7, c_g*10^7, $(c_f+c_m+c_g)*10^7$, γ_g*10^2, μ_i*10^3, p_i*10^{-3} – to replace their original values on the input side. Additionally, the initial gas compressibility factor, $Zi*10^2$, was also included as an input. In networks E and G, functional links, $c_m^{0.01}$, $c_m^{0.1}$,$c_f^{0.01}$, $c_f^{0.1}$, $c_g^{0.01}$, $c_g^{0.1}$, $(c_f+c_m+c_g)^{0.01}$, $(c_f+c_m+c_g)^{0.1}$, $\gamma_g^{0.5}$, $\gamma_g^{0.05}$, $\mu_i^{0.75}$, $\mu_i^{1.5}$, μ_i^{2}, $Z_i^{0.5}$, $Z_i^{0.05}$, q_{sc}^{-1},

$q_{sc}^{0.5}$ and modified initial pseudo-pressure, ψ^{i}_{Di}, were added as input variables in terms of power transformation.

On the output side, the functional links determined from network C, which are $k_f*k_m*10^6$, $k_f*L_x*10^{-1}$, $l*SC*10^{-2}$, $l*L_x*10^{-3}$, $\varphi_m*L_x*10^{-2}$, were included. Furthermore, the variables φ_m, φ_f, k_m, k_f, λ, ω were replaced by φ_m*10, φ_f*10^2, k_m*10^6, k_f*10^2, $\lambda*10^7$, $\omega*10^2$, and the remaining variables *l, SC,* ε were utilized in their original values.

These four networks each trained three times to make sure networks are well generalized to compare their average error. Figure 4.9 represents the architecture used in training networks D, E, F, and G. The main difference of this structure from other network structures discussed so far is the curve lines representing semi-cascade connection from input layer to all the hidden layers. It was observed that this type of connections decreases the networks' performance error faster, and help the generalization process. The full architecture for networks D, E, F, and G is summarized in Table 4.7. The input-output pairing for those four networks are given in Table 4.8 through Table 4.11. It should be noted that the size of input vectors in each network is different due to the number of functional links used in each case. However, the outputs remained the same except for networks F and G where an additional functional link, modified dimensionless time, t_D^i, in Equation 4.8 was added.

Table 4.7: Summary of network structures used in networks D, E, F, and G						
Network	Neurons and transfer function					Original outputs
	Input Layer	Hidden Layer I	Hidden Layer II	Hidden Layer III	Output layer	
D	290 -tansig	100 -tansig	60 -tansig	30 -tansig	15 -tansig	10
E	307 -tansig				15 -tansig	
F	290 -tansig				16 -tansig	
G	307 -tansig				16 -tansig	

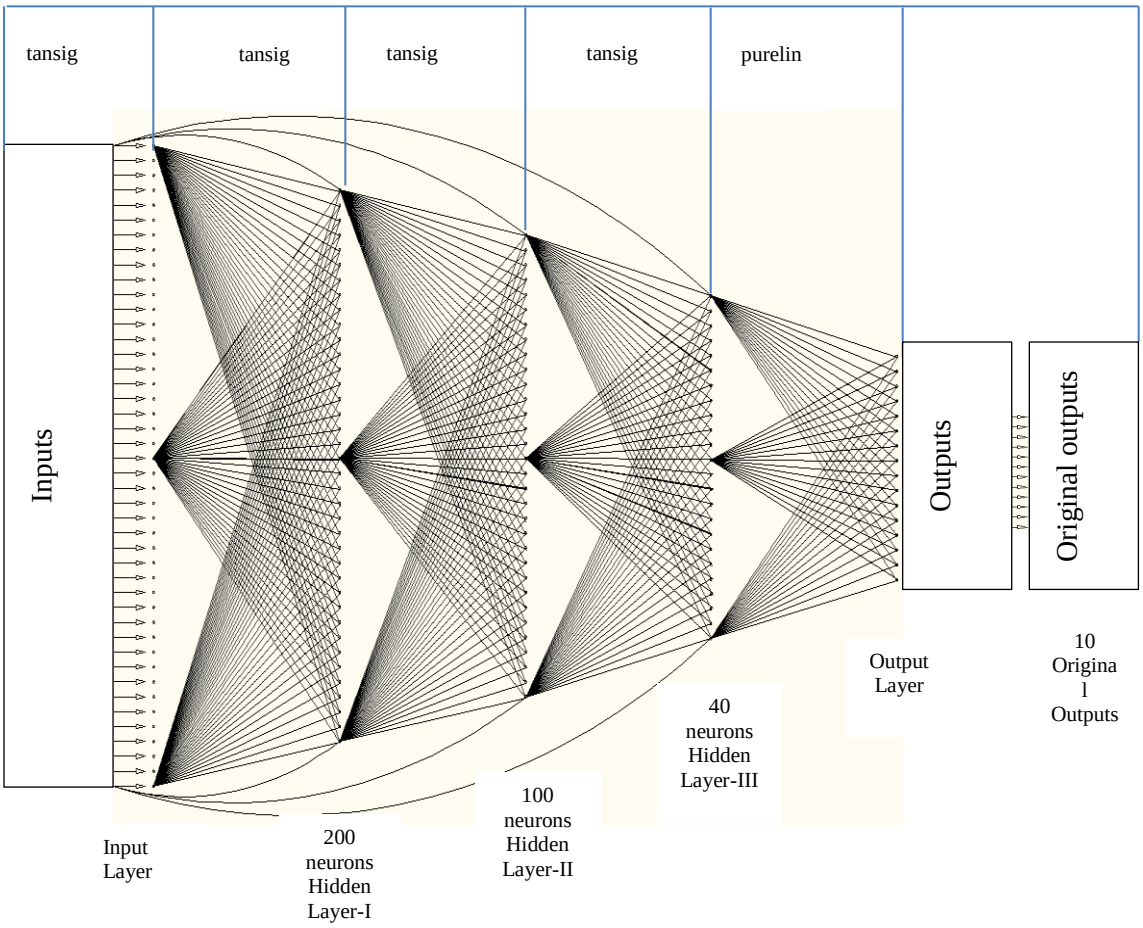


Figure 4.9: Semi-cascade architecture for networks D, E, F, and G

Table 4.8: Input-output pairing for network D

Inputs	Modified outputs	Original outputs
$\frac{p_2-p_1}{\log_{10}(t_2)-\log_{10}(t_1)}$	$O_1=k_f*k_m*10^6$	
.	$O_2=k_f*10^2$	
· n=1, 2, 3…279	$O_3=k_m*10^6$	$k_f= O_2 *10^{-2}$
.	$O_4=k_f *L_x*10^{-1}$	$k_m= O_3 *10^{-6}$
$\frac{p_{n+1}-p_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)}$	$O_5=l*SC*10^{-2}$	$SC=\begin{cases}O_6\\O_5/O_7*10^2\end{cases}$
c_m*10^7	$O_6=SC$	$l = O_7$
c_f*10^7	$O_7=l$	$L_x=\begin{cases}O_9*10\\O_8/O_7*10^3\end{cases}$
c_g*10^7	$O_8=l*L_x*10^{-3}$	$\varphi_m= O_{11}*10^{-1}$
$(c_f+c_m+c_g)*10^7$	$O_9=L_x*10^{-1}$	$\varphi_f= O_{12}*10^{-2}$
γ_g*10^2	$O_{10}=\varphi_m *L_x*10^{-2}$	$\varepsilon= O_{13}$
μ_i*10^3	$O_{11}=\varphi_m *10$	$\lambda= O_{14}*10^{-7}$
q_{sc}	$O_{12}=\varphi_f*10^2$	$\omega= O_{15}*10^{-2}$
T	$O_{13}=\varepsilon$	
h	$O_{14}=\lambda*10^7$	
p_i*10^{-3}	$O_{15}=\omega*10^2$	

Table 4.9: Input-output pairing for network E

Inputs	Modified outputs	Original outputs
$\frac{p_2-p_1}{\log_{10}(t_2)-\log_{10}(t_1)}$	$O_1=k_f*k_m*10^6$	
. . . n=1, 2, 3…279 . . .	$O_2=k_f*10^2$	
$\frac{p_{n+1}-p_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)}$	$O_3=k_m*10^6$	$k_f= O_2 *10^{-2}$
c_m, $c_m^{0.01}$, $c_m^{0.1}$	$O_4=k_f *L_x*10^{-1}$	$k_m= O_3 *10^{-6}$
c_f, $c_f^{0.01}$, $c_f^{0.1}$	$O_5=l*SC*10^{-2}$	$SC = \begin{cases} O_6 \\ O_5/O_7 * 10^2 \end{cases}$
c_g, $c_g^{0.01}$, $c_g^{0.1}$	$O_6=SC$	$l = O_7$
$c_f+c_m+c_g$,	$O_7=l$	$L_x = \begin{cases} O_9 * 10 \\ O_8/O_7 * 10^3 \end{cases}$
$(c_f+c_m+c_g)^{0.01}$,	$O_8=l* L_x*10^{-3}$	$\varphi_m= O_{11}*10^{-1}$
$(c_f+c_m+c_g)^{0.1}$,	$O_9=L_x*10^{-1}$	$\varphi_f= O_{12}*10^{-2}$
γ_g, $\gamma_g^{0.5}$, $\gamma_g^{0.05}$	$O_{10}=\varphi_m *L_x*10^{-2}$	$\varepsilon= O_{13}$
μ_i, $\mu_i^{0.75}$, $\mu_i^{1.5}$, μ_i^2	$O_{11}=\varphi_m *10$	$\lambda= O_{14}*10^{-7}$
Z_i, $Z_i^{0.5}$, $Z_i^{0.05}$	$O_{12}=\varphi_f*10^2$	$\omega= O_{15}*10^{-2}$
q_{sc}, q_{sc}^{-1}, $q_{sc}^{0.5}$	$O_{13}=\varepsilon$	
T	$O_{14}=\lambda*10^7$	
h	$O_{15}=\omega*10^2$	
p_i		

Table 4.10: Input-output pairing for network F		
Inputs	Modified outputs	Original outputs
	$O_1=k_f*k_m*10^6$	
$\frac{\psi^i_{D1}-\psi^i_{D2}}{log_{10}(t_2)-log_{10}(t_1)}$	$O_2=k_f*10^2$	
.	$O_3=k_m*10^6$	
. n=1, 2, 3…279	$O_4=k_f*L_x*10^{-1}$	$k_f= O_2*10^{-2}$
.		$k_m= O_3*10^{-6}$
$\frac{\psi^i_{D1}-\psi^i_{D2}}{\log_{10}(t_{n+1})-\log_{10}(t_n)}$	$O_5=l*SC*10^{-2}$	
c_m*10^7	$O_6=SC$	$SC=\begin{cases}O_6\\O_5/O_7*10^2\end{cases}$
c_f*10^7	$O_7=l$	$l = O_7$
c_g*10^7	$O_8=l*L_x*10^{-3}$	$L_x=\begin{cases}O_9*10\\O_8/O_7*10^3\end{cases}$
$(c_f+c_m+c_g)*10^7$	$O_9=L_x*10^{-1}$	
γ_g*10^2	$O_{10}=\varphi_m*L_x*10^{-2}$	$\varphi_m= O_{11}*10^{-1}$
μ_i*10^3	$O_{11}=\varphi_m*10$	$\varphi_f= O_{12}*10^{-2}$
q_{sc}	$O_{12}=\varphi_f*10^2$	$\varepsilon= O_{13}$
T	$O_{13}=\varepsilon$	$\lambda= O_{14}*10^{-7}$
h	$O_{14}=\lambda*10^7$	$\omega= O_{15}*10^{-2}$
ψ^i_{Di}	$O_{15}=\omega*10^2$	
	$O_{16}=t^i_D$	

Table 4.11: Input-output pairing for network G

Inputs	Modified outputs	Original outputs
$\frac{\psi_{D_1}^i - \psi_{D_2}^i}{log_{10}(t_2) - log_{10}(t_1)}$	$O_1 = k_f * k_m * 10^6$	
.	$O_2 = k_f * 10^2$	
. n=1, 2, 3…279	$O_3 = k_m * 10^6$	
.		$k_f = O_2 * 10^{-2}$
$\frac{\psi_{D_1}^i - \psi_{D_2}^i}{\log_{10}(t_{n+1}) - \log_{10}(t_n)}$	$O_4 = k_f * L_x * 10^{-1}$	$k_m = O_3 * 10^{-6}$
c_m, $c_m^{0.01}$, $c_m^{0.1}$	$O_5 = l * SC * 10^{-2}$	
c_f, $c_f^{0.01}$, $c_f^{0.1}$	$O_6 = SC$	$SC = \begin{cases} O_6 \\ O_5/O_7 * 10^2 \end{cases}$
c_g, $c_g^{0.01}$, $c_g^{0.1}$	$O_7 = l$	$l = O_7$
$c_f + c_m + c_g$	$O_8 = l * L_x * 10^{-3}$	$L_x = \begin{cases} O_9 * 10 \\ O_8/O_7 * 10^3 \end{cases}$
$(c_f + c_m + c_g)^{0.01}$	$O_9 = L_x * 10^{-1}$	$\varphi_m = O_{11} * 10^{-1}$
$(c_f + c_m + c_g)^{0.1}$	$O_{10} = \varphi_m * L_x * 10^{-2}$	$\varphi_f = O_{12} * 10^{-2}$
γ_g, $\gamma_g^{0.5}$, $\gamma_g^{0.05}$	$O_{11} = \varphi_m * 10$	$\varepsilon = O_{13}$
μ_i, $\mu_i^{0.75}$, $\mu_i^{1.5}$, μ_i^2	$O_{12} = \varphi_f * 10^2$	$\lambda = O_{14} * 10^{-7}$
q_{sc}, q_{sc}^{-1}, $q_{sc}^{0.5}$	$O_{13} = \varepsilon$	$\omega = O_{15} * 10^{-2}$
T	$O_{14} = \lambda * 10^7$	
h	$O_{15} = \omega * 10^2$	
ψ_{Di}^i	$O_{16} = t_D^i$	

After training networks D, E, F, and G, with the same hidden layer and hidden neuron numbers, it was observed that input-output pairing in network E was able to outperform better than others by decreasing validation error dramatically; therefore, it was decided to further improve the prediction capability and robustness of the network E by changing hidden layer sizes while keeping input-output pairing intact. The best architecture in Stage II was obtained through three hidden layers size of 157 (*tansig*)×77 (*tansig*) ×30 (*tansig*) depicted in network H shown in Figure 4.10.

In terms of evaluating the prediction performances, it was advisable to treat output variables with small values differently from ones with large values because of large difference in output variable ranges. Instead of focusing on real values of matrix permeability (k_m), inter-porosity flow parameter (λ), it was decided to compare the actual values to their ANN predictions. The testing evaluation obtained from final architecture of network H (shown in Figure 4.11) is much better compared to the results of previous networks.

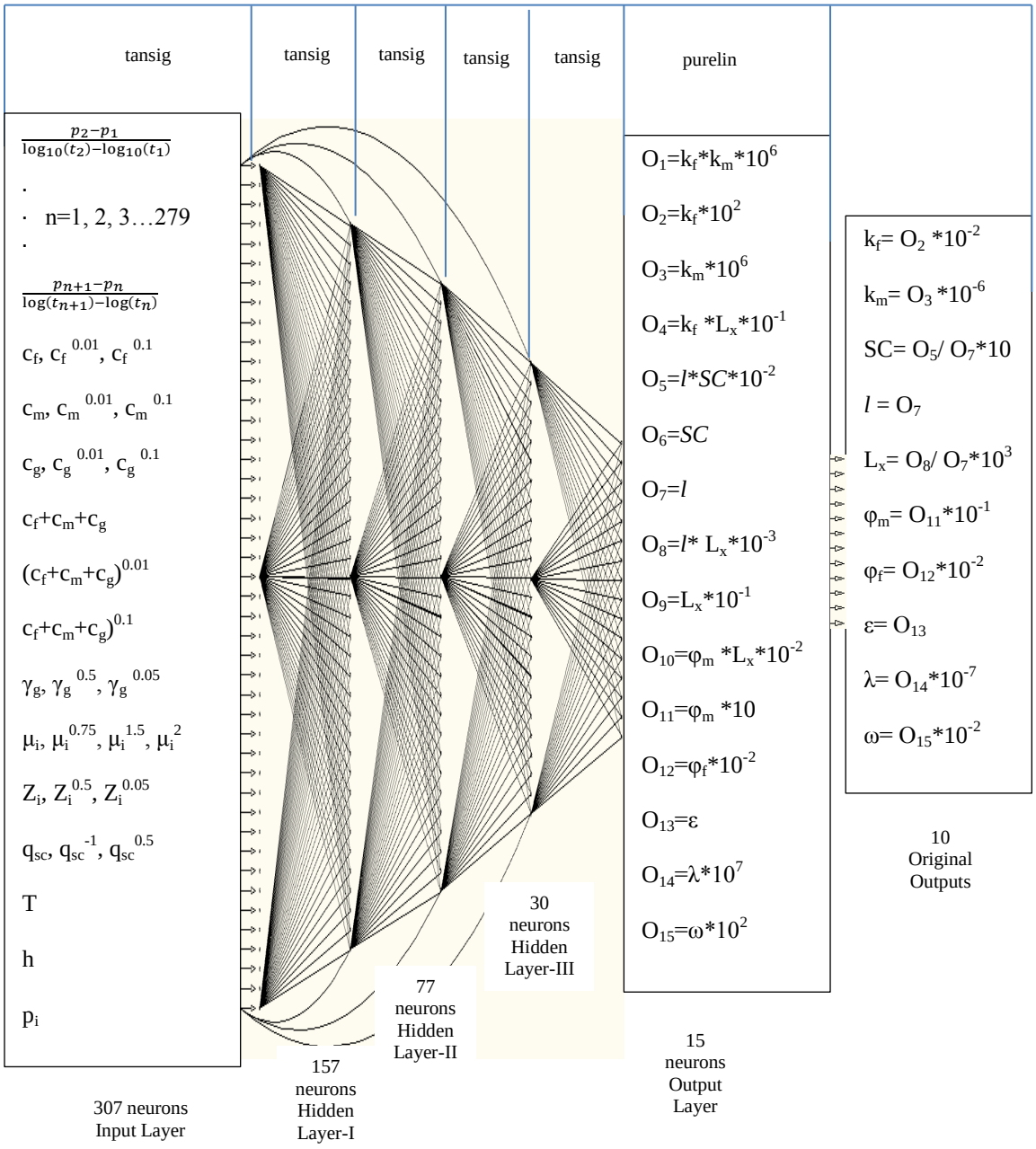
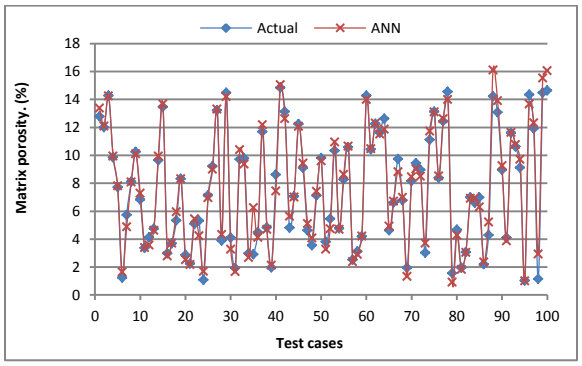
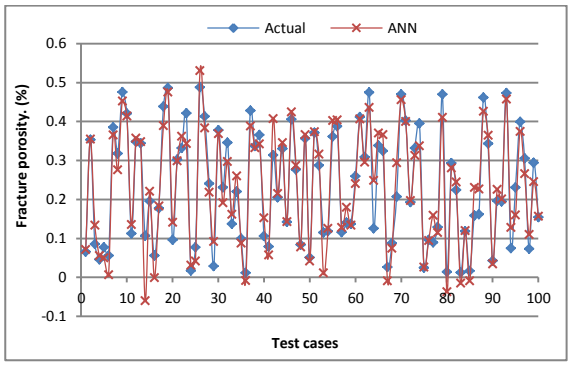
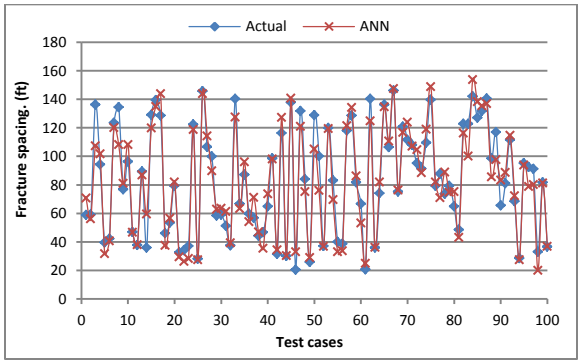
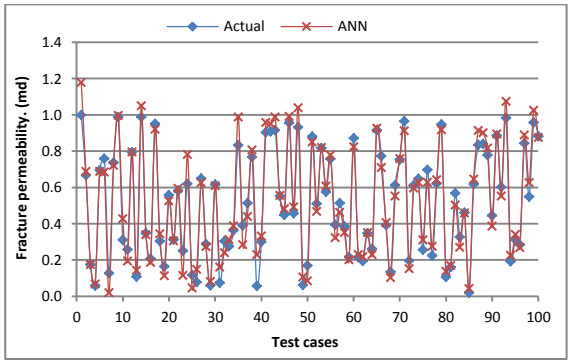


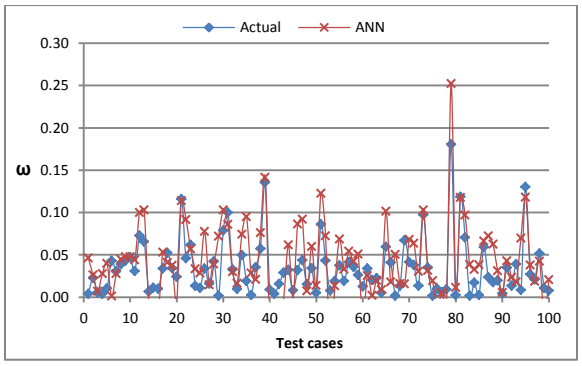
Figure 4.10: Network H, the final architecture in Stage II

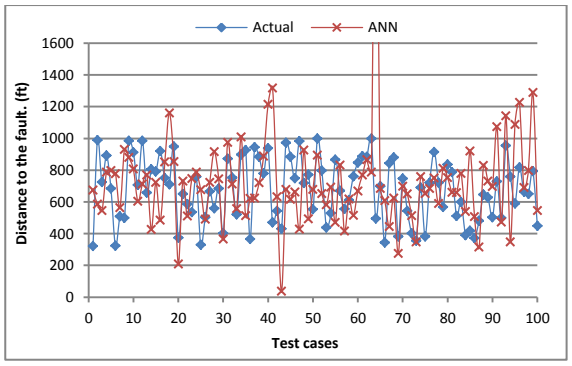

Actual
ANN
Matrix porosity. (%)
Test cases

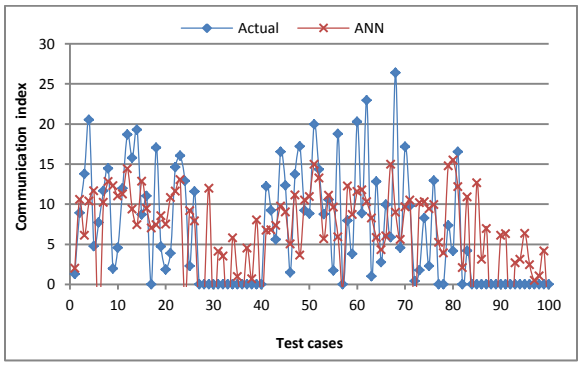

Actual
ANN
Fracture porosity. (%)
Test cases

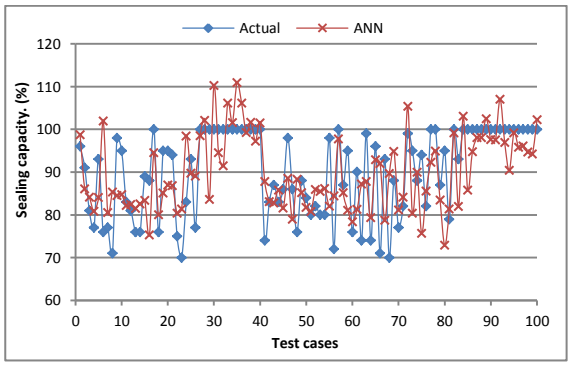

Actual
ANN
Fracture spacing. (ft)
Test cases


Actual
ANN
Fracture permeability. (md)
Test cases


Actual
ANN
ω
Test cases


Actual
ANN
Distance to the fault. (ft)
Test cases


Actual
ANN
Communication index
Test cases


Actual
ANN
Sealing capacity. (%)
Test cases

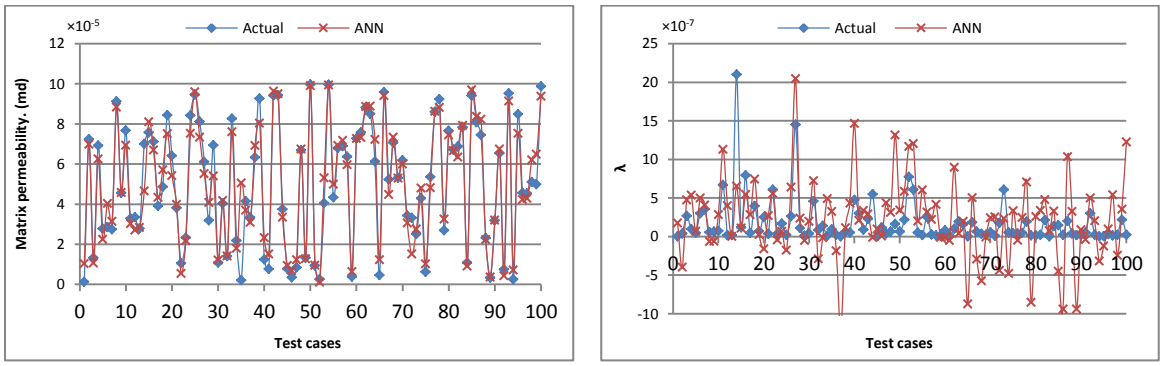


Figure 4.11: Training results using network H final architecture (Stage II)

4.3 Stage III

The reservoir becomes more complex when the fracture exhibits anisotropy while having a fault that is sealing or partially communicating. The evaluation of formation characteristics in anisotropic naturally fractured gas reservoirs becomes more challenging where directional permeability values and orientation of the fault with respect to the principal flow directions have to be incorporated into the output variables. In order to have reasonable predictions for anisotropic permeability and fault orientation on top of other formation characteristics, the following assumptions are made:

- Fracture permeability is anisotropic
- The matrix permeability is isotropic
- The fault line is located across the reservoir with almost no storage capacity

In the reservoir simulation model, different values for fracture permeability in the *x* and *y* flow directions were randomly generated within the established ranges to formulate the anisotropy, and the fault line across the reservoir was placed in such a way that the angle, θ, between fault line and the principal flow direction *x* was randomly chosen from the range [0^o, 180^o]. Moreover, the grid refinement was done to all grid blocks where the fault line resides in a diagonal orientation (Figure 4.12). In this stage, apart from the distance to the fault (L_x) sealing capacity (SC) and communication index, ε fault orientation (θ) is an important parameter for identification of a fault as shown in Figure 4.13. 500 sets of new data were generated and simulated in reservoir simulator. When trained with network H, the prediction errors observed to be increased as compared to the results in Figure 4.11 because of additional unknowns in the output layer.

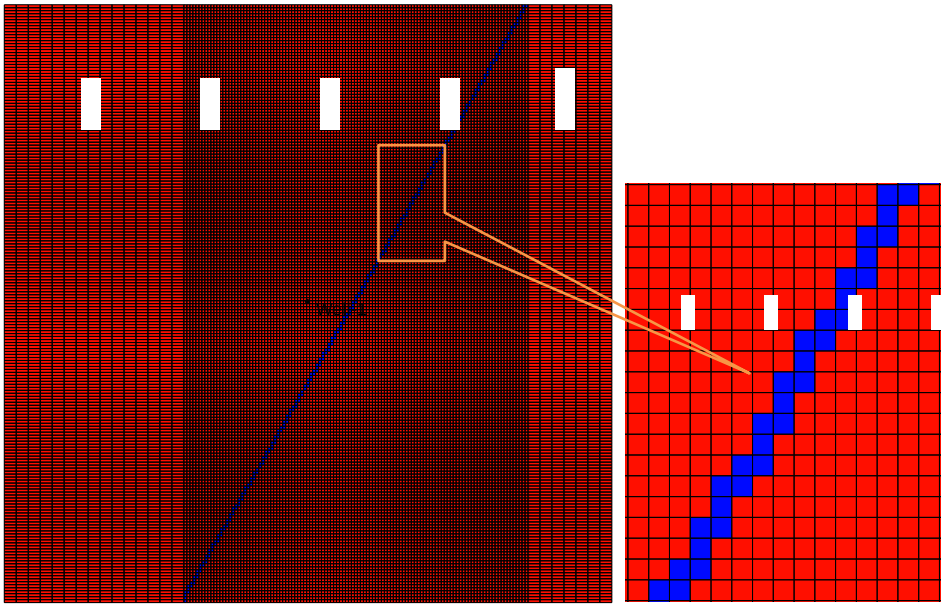


Figure 4.12: Fault with an angle modeled in CMG with grid refinement

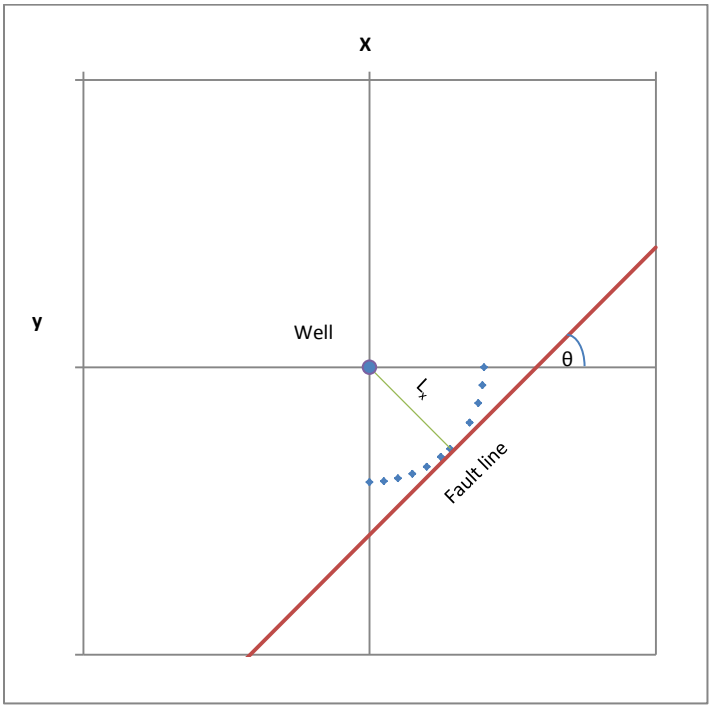


Figure 4.13: Fault position in anisotropic reservoir with respect to the well

After many attempts, it was observed that the network performance is sensitive to change in bottom-hole pressure per unit interval. In order to increase the significance of pressure decline, the production rate was manually increased in each case to space out the pressure decline in a unit time interval while keeping bottom-hole pressure under atmospheric pressure at the end of ten years production period. In other words, with increased production rate, the slopes of pressure change against logarithm of time increase significantly such that there is a sharp decline in Horner plot as shown in Figure 4.14. The general purpose of doing so is to emphasize the influence of main component of input vector– slopes of pressure against logarithm of time.

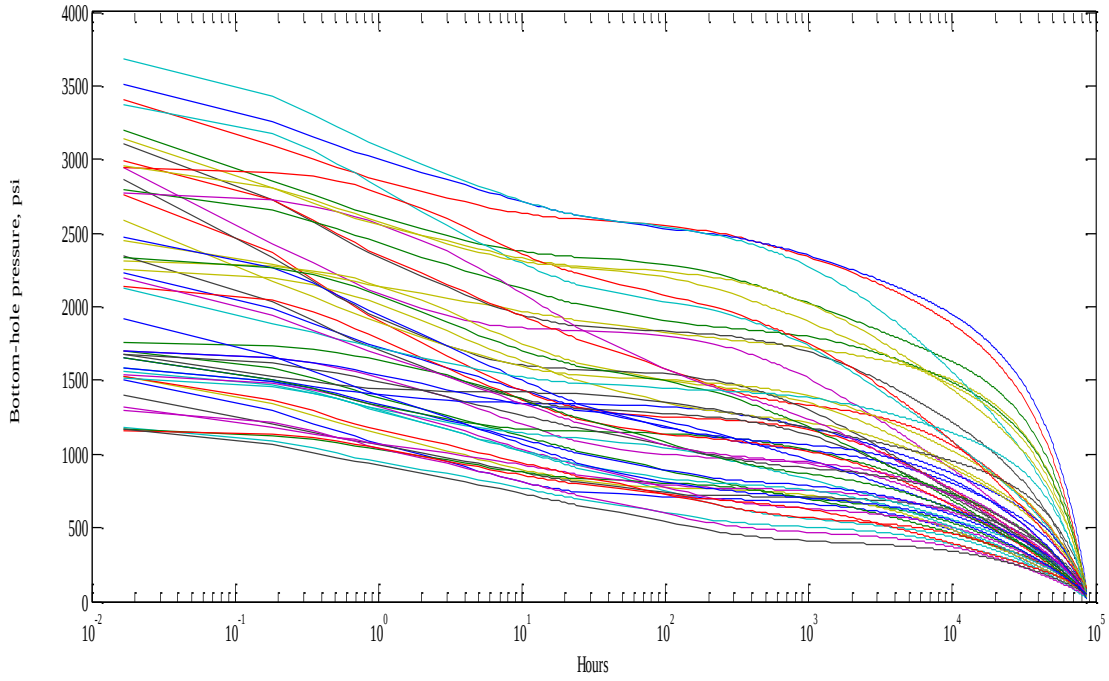


Figure 4.14: Horner plot of 50 cases over a ten year period

From ANN development in the previous two stages, it is known that predicting fault characteristics were difficult, especially, near the boundaries of the variables. Low permeability values in *x* and *y* directions increase the nonlinearity of the problem even further. To provide

functional links to coupe with such difficulties, it was decided to include the production rate on both the input and output side. More hard-to-predict variables were bundled with production rate to increase their predictability. Another major change was to increase the number of hidden layers and to adjust the hidden neuron numbers accordingly. The final network architecture of Stage III is illustrated in Figure 4.15; complete input-output pairing for the network used in this stage is presented in Table 4.12. The testing results achieved in training the network I are given in Figure 4.16. In terms of data processing and Designing ANNS, the following computer programs were developed.

- Input and output data organization for the ANN
- Example of Matlab Neural Network Toolbox

The source codes are included in Appendix D.

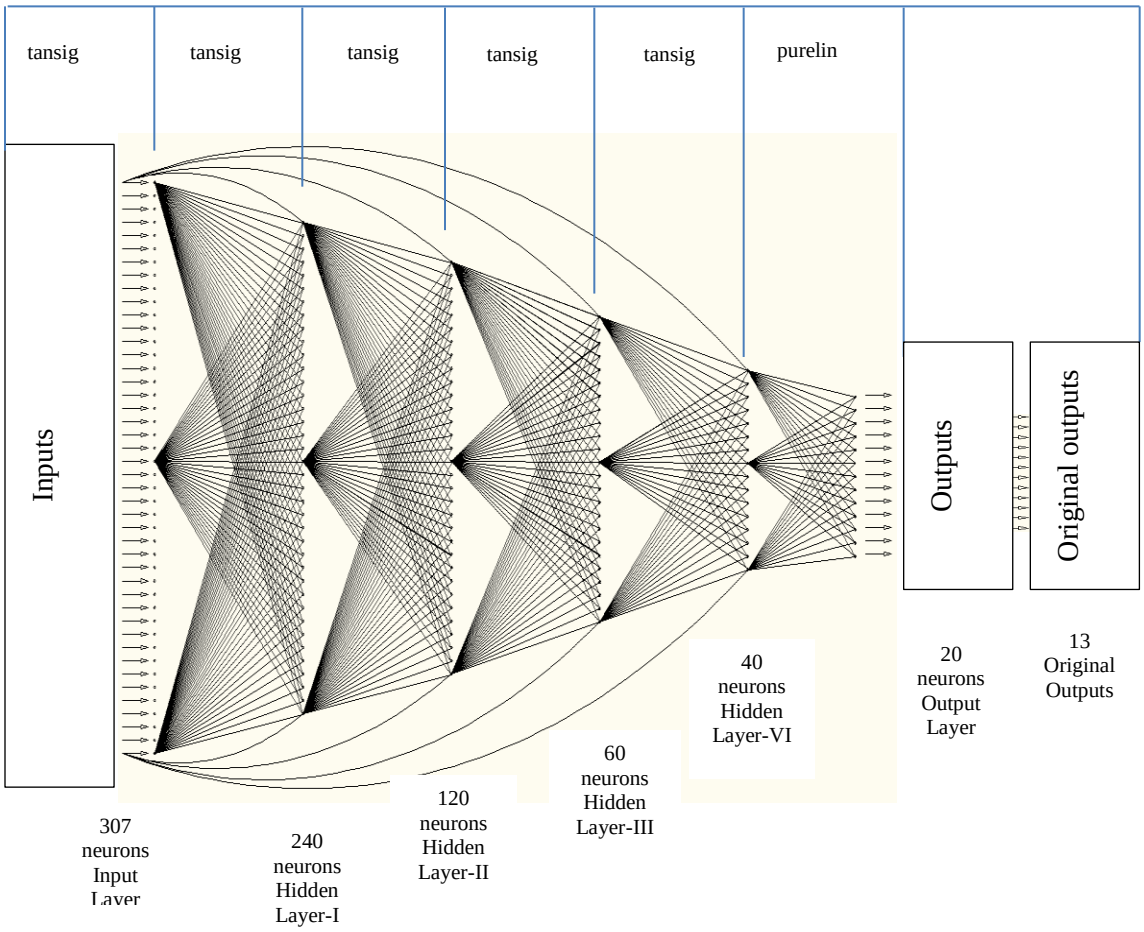
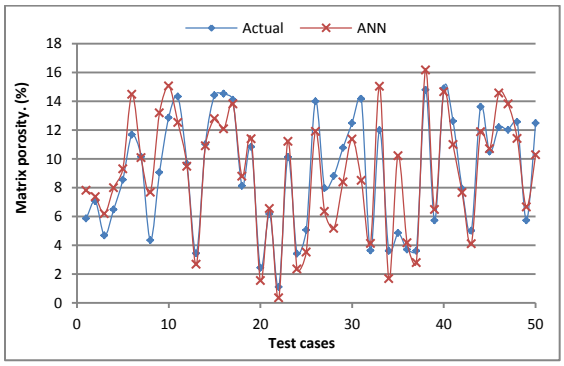
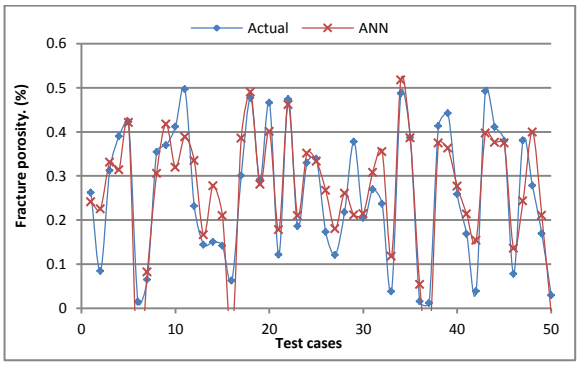


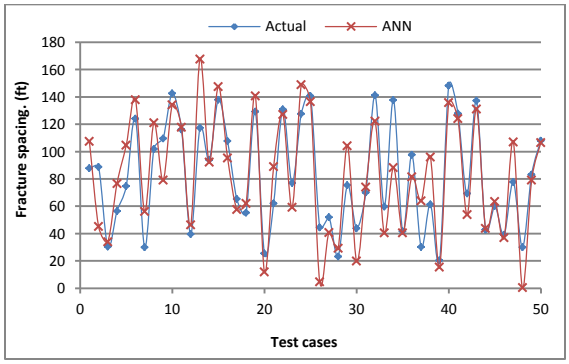
Figure 4.15: Semi-cascade architecture network I

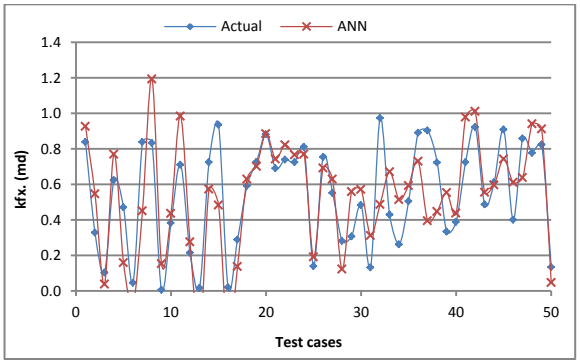
Table 4.12: Input-output pairing for network I

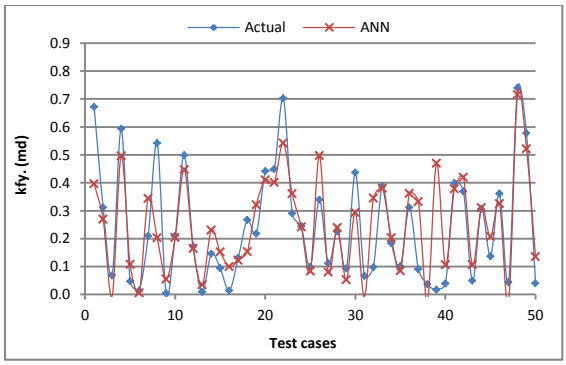
Inputs	Modified outputs	Original outputs
	$O_1=k_{fy}/k_{fx}*10^2$	
	$O_2=k_{fx}*10^2$	
$\frac{p_2-p_1}{\log_{10}(t_2)-\log_{10}(t_1)}$	$O_3=k_{fy}*10^2$	
.	$O_4=(k_{fx}*k_{fy})^2*10^2$	
· n=1, 2, 3…279		$k_{fy}/k_{fx}=O_1*10^{-2}$
.	$O_5=q_{sc}*k_m*10^5$	
$\frac{p_{n+1}-p_n}{\log_{10}(t_{n+1})-\log_{10}(t_n)}$		$k_{fx}=O_2*10^{-2}$
c_m, $c_m^{0.01}$, $c_m^{0.1}$	$O_6=k_m*10^5$	
	$O_7=q_{sc}*SC*10^{-1}$	$k_{fy}=O_3*10^{-2}$
c_f, $c_f^{0.01}$, $c_f^{0.1}$		$k_m=O_6*10^{-5}$
	$O_8=SC*10^{-1}$	
c_g, $c_g^{0.01}$, $c_g^{0.1}$		$SC=O_8*10$
	$O_9=q_{sc}*L_x*10^{-2}$	
$c_f+c_m+c_g$,		$L_x=O_{10}*10^2$
	$O_{10}=L_x*10^{-2}$	
$(c_f+c_m+c_g)^{0.01}$,		$\theta=O_{12}*10$
	$O_{11}=q_{sc}*\theta*10^{-1}$	
$(c_f+c_m+c_g)^{0.1}$		$\varepsilon=O_{14}*10$
	$O_{12}=\theta*10^{-1}$	
γ_g, $\gamma_g^{0.5}$, $\gamma_g^{0.05}$		$\varphi_m=O_{16}*10^{-1}$
	$O_{13}=q_{sc}*\varepsilon*10^{-1}$	
μ_i, $\mu_i^{0.75}$, $\mu_i^{1.5}$, μ_i^2		$\varphi_f=O_{17}*10^{-2}$
	$O_{14}=\varepsilon*10^{-1}$	
Z_i, $Z_i^{0.5}$, $Z_i^{0.05}$		$l=O_{18}*10$
	$O_{15}=q_{sc}$	
q_{sc}, q_{sc}^{-1}, $q_{sc}^{0.5}$		$\lambda=O_{19}*10^{-7}$
	$O_{16}=\varphi_m*10$	
T		$\omega=O_{20}*10^{-2}$
	$O_{17}=\varphi_f*10^2$	
h		
	$O_{18}=l*10^{-1}$	
p_i		
	$O_{19}=\lambda*10^7$	
	$O_{20}=\omega*10^2$	

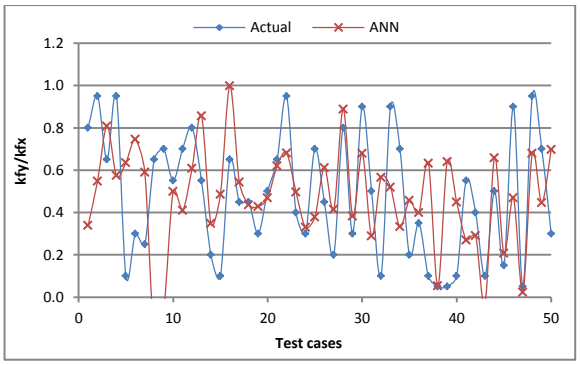

Actual
ANN
Matrix porosity. (%)
Test cases

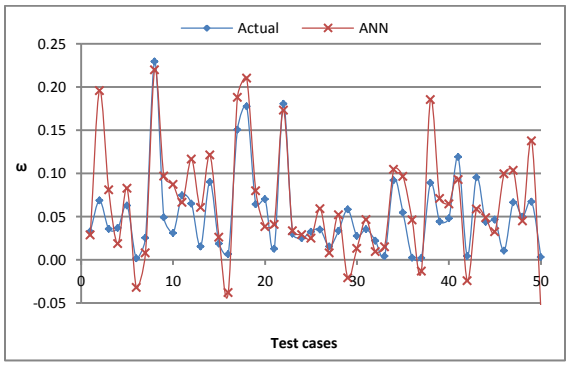

Actual
ANN
Fracture porosity. (%)
Test cases


Actual
ANN
Fracture spacing. (ft)
Test cases


Actual
ANN
kfx. (md)
Test cases


Actual
ANN
kfy. (md)
Test cases


Actual
ANN
kfy/kfx
Test cases


Actual
ANN
ε
Test cases

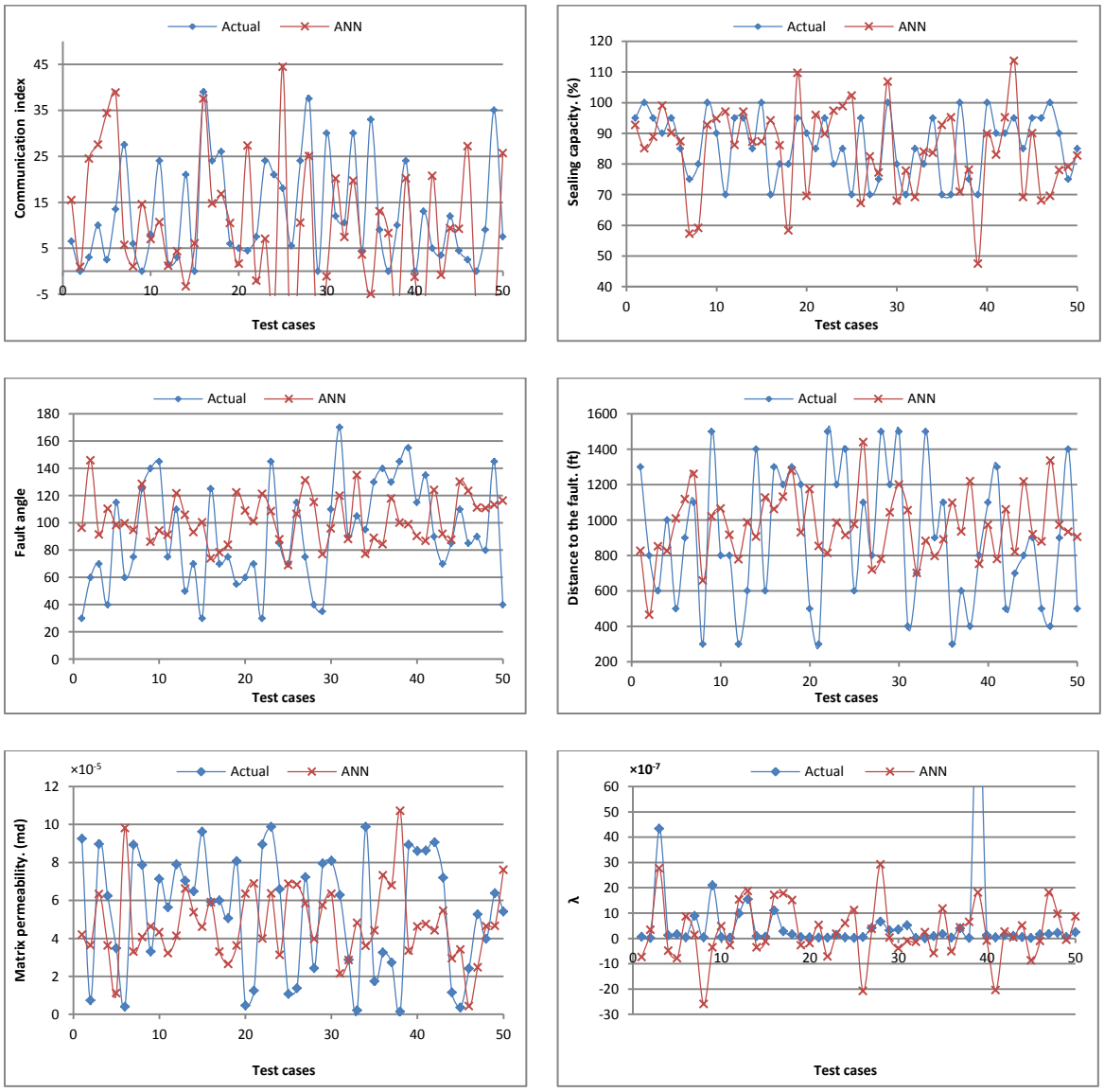


Figure 4.16: Training results using network H final architecture (Stage II)

From observation of the testing results done on fifty cases in Figure 4.16, the network model did perform well on the data constructed for anisotropic gas reservoirs, it was able to predict the principle formation characteristics, namely, matrix porosity, fracture porosity, fracture spacing, fracture permeability in both *x* and *y* directions with acceptable accuracy. The double porosity features, inter-porosity flow parameter and storativity ratio were also captured. Characteristic parameters of the fault line were configured in more than one way to paint a reasonable picture of a semipermeable or complete sealing barrier in a reservoir.

4.4 Case Study

In this section, three cases of anisotropic reservoirs with random faults were generated to evaluate the capability of the semi-cascade multivariate network architecture presented in Figure 4.15, while maintaining the input-output pairing listed in Table 4.12. The full characteristics of those three reservoirs were given in Table 4.13. The simulated drawdown test results are presented in the form of the Horner plot shown in Figure 4.17.

Three case studies in which the network I is used to compare the prediction quality are presented in Table 4.14. Results are also plotted in spider diagram to help visualize the reservoir characteristics versus the ANN predictions shown in Figure 4.18. The focus is on applying the pre-trained network (network I) in standalone mode to predict the desired outputs and to validate the prediction results. As it was pointed out in the previous section, the prediction performances of outputs with small values still remain low; however, considering the magnitude differences between targets and ANN predictions, the network still manages to give reasonable results within the same order of magnitude. Results show considerable improvement in prediction accuracy of the fault characteristics.

Table 4.13: Reservoir parameters for evaluation of network I										
Inputs										
Cases	p_i	T	h	γ_g	q_{sc}	c_m	c_f	c_g	Z_i	μ_i
1	4361	173	64	0.589	3.2867	8.10E-06	8.68E-05	1.68E-04	0.962	0.0220
2	4856	181	40	0.665	0.7484	4.23E-06	7.50E-05	1.30E-04	0.978	0.0248
3	1521	157	64	0.764	0.8154	1.36E-05	9.07E-05	7.54E-04	0.815	0.0149

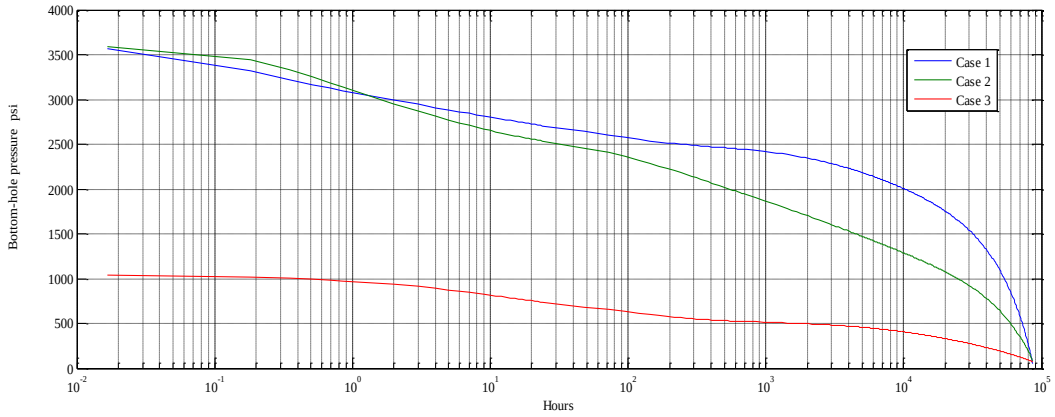


Figure 4.17: Horner plot of three cases over ten year period

Table 4.14: Results comparison in the case studies						
Characteristics	Case1		Case2		Case3	
	Actual	ANN	Actual	ANN	Actual	ANN
k_{fy}/k_{fx}	0.75	0.23	0.45	0.27	0.25	0.49
k_{fx}	0.667	1.042	0.192	0.284	0.834	0.729
k_{fy}	0.500	0.392	0.086	-0.088	0.208	0.427
l	112	130	36	92	110	112
φ_m	6.5	6.7	2.5	1.5	9.7	11.8
φ_f	0.27	0.16	0.46	0.53	0.37	0.37
ω	0.056	0.115	0.220	0.218	0.041	0.048
ε	0	2	24	29	20	17
θ	170	144	45	71	170	114
L_x	800	1070	400	309	1000	636
SC	100	65	70	29	90	85
$k_m \times 10^5$	4.9	0.9	0.8	7.8	4.6	4.7
$\lambda \times 10^7$	0.3	19.4	1.8	7.2	0.3	-9.3

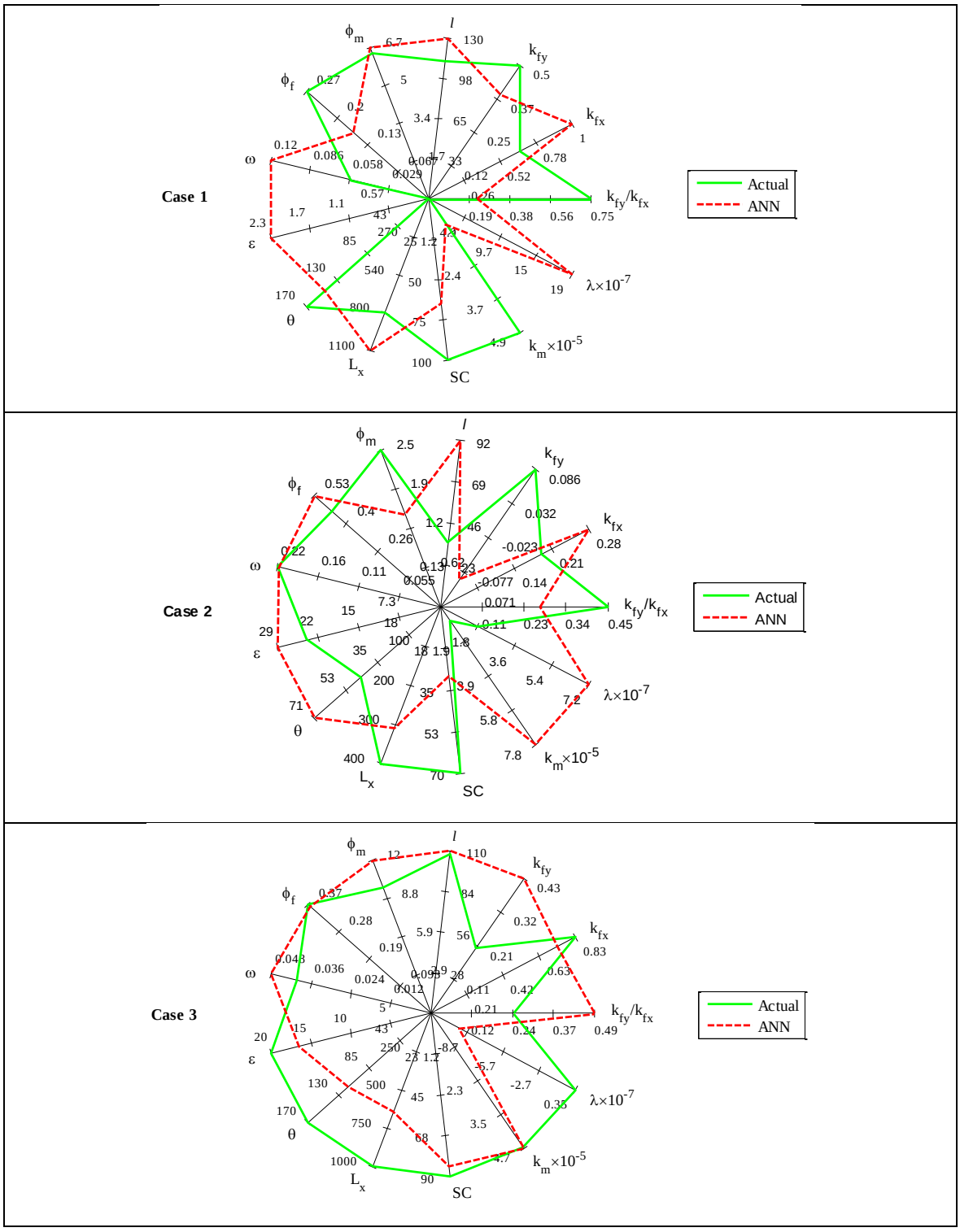


Figure 4.18: Validation results of cases 1, 2, and 3

Chapter 5

Discussions and Conclusions

As discussed in the previous chapters, Artificial Neural Networks (ANNS) are powerful tool in the field of petroleum engineering, capable of mapping a wide range of multidimensional input output relationships without having exact analytical functions. In Chapters 3 and 4 we have described the tools and methods in improving prediction capability of the networks.

The general strategy of growing a network from a single output network to almost two dozen outputs is that the network which performs best is retained. The effect of adding each remaining outputs and effective functional links in turn was assessed. This process was repeated many times until the addition of extra variables and functional links did not result in significant improvement in the network performance. In this study the high level of performance achieved by the neural networks suggests that it would be suitable to use neural networks for retrieval of reservoir characteristics.

In this study, all networks are used with simulated reservoir observations. Comprehensive reservoir proxies are built to model naturally fractured double porosity gas reservoirs. The type of data used is generally composed with the aid of known relationships that have physical basis. It was demonstrated that the distinct advantages in using analytical techniques provide the network with reliable functional links, and helps both inputs and outputs for multivariate ANN models. The underlying physical process has to be understood in order to formulate necessary functional links. Identifying the predictability of each reservoir and fault characteristics helped isolating the strong parameters from weak ones, then the emphasis was placed on increasing the accuracy of weak parameter predictions by coupling the ambiguous ones

with easy-to-predict parameters. Input is also bundled with outputs to steer those hard-to-predict parameters in the right direction. It should be pointed out that inputs should be in established variable ranges to use the trained network.

Application of homogenous anisotropic reservoir system is considered, given the fact that there are no directional properties involved with formation characteristics, fluid properties, and, bottom-hole pressure, when predicting anisotropic formation, the network still gives reasonable output approximations for Stage III. However, a strong degree of uncertainty in the uniqueness of fault characteristics which are inducing certain bottom-hole pressure profile complicates the already challenging task of fault parameter retrieval. Because of the pertinent relationship to be extracted can be subtle and therefore can be easily obscured by noises. To quote Professor Robert Tibshirani, "it is often better to have approximate solution to a real problem than an exact solution to an oversimplified one." [Tibshirani, 1994]. This problematic fact could be solved by incorporating geophysical data.

Having an accurate, comprehensive, and extensive data set remains the key attributes in data preparation. When more data are available, the network will be able to distinguish certain heterogeneity of the outputs. The number of patterns and quality of the data is increased; the prediction capability of the network is increased too. High temporal resolution of bottom-hole pressure profile was maintained throughout this study, and a checking mechanism programmed into the script file generator to guarantee the double porosity feature is present.

As numbers of output variables grow, so does the complexity of the network structure. In comparative analysis among the multivariate ANN models during the testing phase, it was evident that the ANN with the most hidden layers is capable of providing good estimates.

Although every trained ANN investigated in this study is unique, in all network structures, tangent-sigmoidal transfer function (tansig) is proven to be the most effective in the input and hidden layers. The linear transfer function (purelin) is used in the output layers of all ANN architecture.

Considering the potential application of this study, future work will involve further evaluation of the network with heterogeneous reservoir, multiphase flow patterns, as well as horizontal wells.

The general guidelines in the ANN developing process are outlined in Figure 5.1. If there is no improvement in performance, the user can abandon the unsuccessful network at any point and modify it with any new choices in this strategy. The following conclusions can be drawn based on the observations made during the development phases of the network:

1. Analyzing the reservoir properties using neural networks based on the well test data demonstrated the potential application of ANNs in pressure transient analysis.
2. The priori knowledge of the properties and underlying information content about the system can be used to improve the networks.
3. Equalizing the magnitude of data is important for giving equal chance of influence to the components of the input vectors as well as for giving equal chance of predictability to the output variables.
4. Computational efficiency of the networks is improved by keeping the network structure as simple as possible.
5. Choosing the appropriate functional links is beneficial to creating more stable and better conditioned data representation for the networks.
6. Reservoir parameters with small values and fault characteristics present a strong degree of challenge to the neural networks.

7. The level of difficulties in predicting certain parameters can be decreased by representing hard-to-predict variables indirectly bundling with variables of high prediction accuracy.
8. As the number of outputs is increased the size of the hidden layers can accordingly be increased.
9. It is observed that more training patterns with normal distribution result in more accurate predictions.
10. The ANN tool developed in this research provides insight about the pressure transient behaviors of such complex systems.
11. The critical fault communication parameter is identified to be ε_c=10, above which the fault presence becomes undetectable.

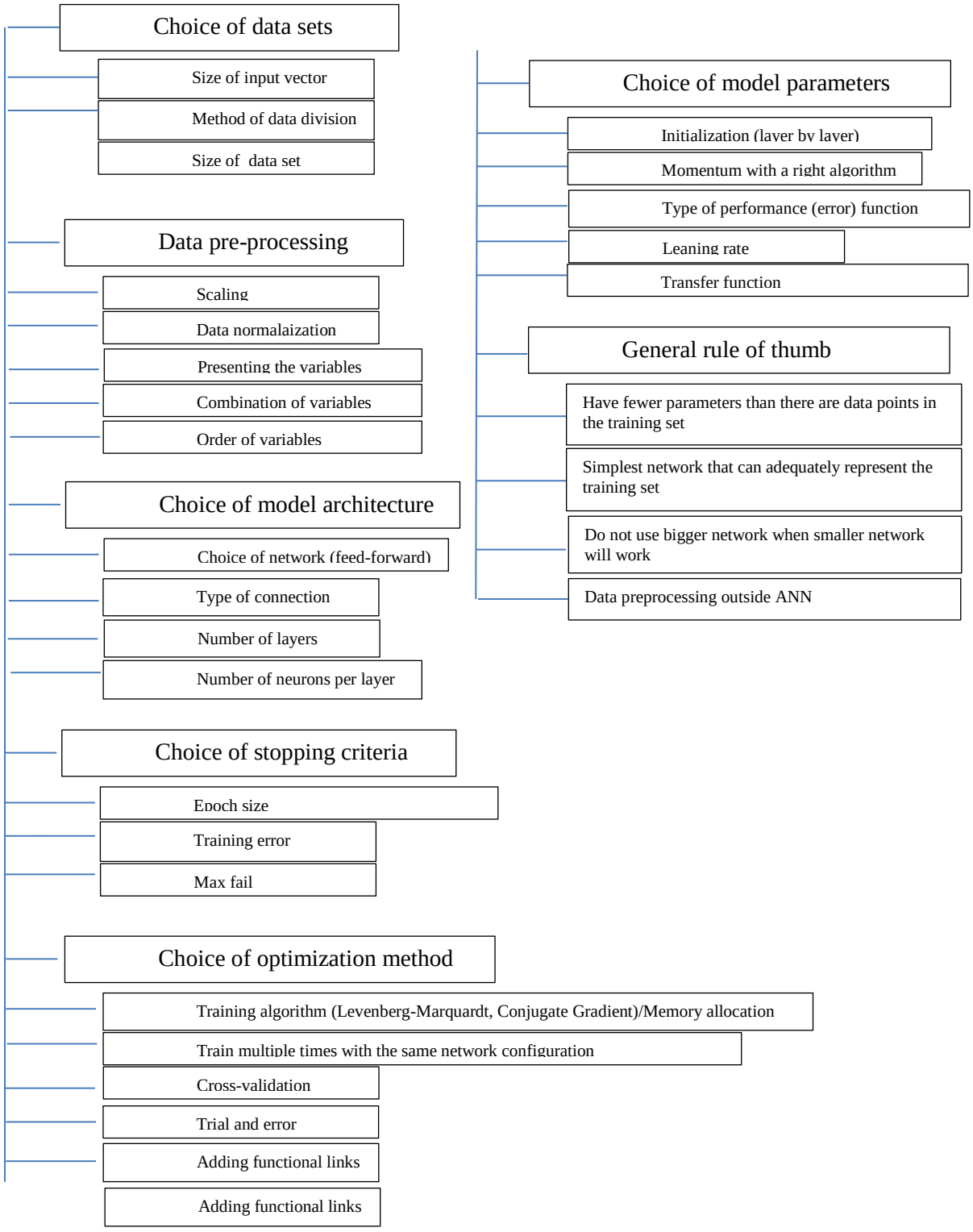


Figure 5.1: Main steps in the development of ANN model

References

Aguilera, Roberto. *Naturally fractured reservoirs*. Tulsa, Okla.: Petroleum Pub. Co.,1980.

Aguilera, Roberto. *Naturally fractured reservoirs*. 2nd ed. Norwood Mass.: Books24x7.com, 2005.

Blackwell, William J., and Frederick W. Chen. *Neural networks in atmospheric remote sensing*. Boston: Artech House, 2009.

Ertekin, Turgay, Jamal H. Kassem, and Gregory R. King. *Basic applied reservoir simulation*. Richardson, Tex.: Society of Petroleum Engineers, 2001.

Govindaraju, Rao S.. *Artificial neural networks in hydrology*. Dordrecht: Kluwer Academic Publishers, 2000.

Hagan, Martin T., Howard B. Demuth, and Mark H. Beale. *Neural network design*. Boston: PWS Pub., 1996.

Haykin, Simon S.. *Neural networks: a comprehensive foundation*. New York: Macmillan ;, 1994.

Hertz, John, Anders Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation*. Redwood City, Calif.: Addison-Wesley Pub. Co., 1991.

Lee, J., and J. B. Rollins. *Pressure transient testing*. Richardson, Tex.: Society of Petroleum Engineers, 2003.

Reiss, Louis Herbert, and Max Creusot. *The Reservoir engineering aspects of fractured formations*. Paris: EÌ • ditions Technip, 1980.

Steeb, W.. *The nonlinear workbook chaos, fractals, cellular automata, neural networks, genetic algorithms, fuzzy logic : with C++, Java, SymbolicC++ and Reduce programs*. Singapore: World Scientific, 20011999.

Theory and practice of the testing of gas wells. 3d ed. Calgary: Energy Resources Conservation Board, 1975.

A. O. Igbokoyi, D. Tiab. *New Method of Well Test Analysis in Naturally Fractured Reservoirs Based on Elliptical Flow.* Petroleum Society of Canada, 2008.

Abdulla Al-Ghamdi, Iraj Ershaghi. *Detection of Communication Cross Faults in Naturally Fractured Reservoirs.* Anchorage Alaska: SPE, 1999.

al, van den Berg et. *Centering, scaling, and transformations: improving the biological information content of metabolomics data.* BMC Genomics, 2006.

Alajmi, Mohammad Naser. *The Development of an Artificial Neural Network as a Pressure Transient Analysis Tool for Applications in Double-Porosity Reservoirs.* SPE, 2003.

Artun, Fazil. Emre. *Optimized design of cyclic pressure pulsing in naturally fractured reservoirs using neural-network based proxy models* . The Pennsylvania State University, 2008.

Aydinoglu, Gokhan. *characteristics of sealing and partially communicating fault-reservoir systems using artificial neural network.* Penn Sate, 2002.

Currie, J.D. Jansen and P.K. *Modelling and Optimisation of Oil and Gas Production Systems*. Delft University of Technology, 2004.

Ershaghi, Robert Khachatoorian and Iraj. *Complexities in the Analysis of Pressure-Transient Response in Faulted Naturally Fractured Reservoirs.* SPE, 1995.

Felipe, Ayala H. Luis. *Compositional modeling of naturally-fractured gas-condensate reservoirs in multi-mechanistic flow domains* . The Pennsylvania State University, 2004.

Guo, B., Schechter, D.S., New Mexico Institute of Mining and Technology, and A., Haliburton Energy Services Banik. "*Use of Single-Well Test Data for Estimating Permeability Anisotropy of the Naturally*." n.d.

Hamid Sarkheil, Hossein Hassani, Firuz Alinia. *The Fracture Network Modeling in Naturally Fractured Reservoirs Using Artificial Neural Network Based on Image Loges and Core Measurements.* Tehran: Amirkabir university of Technology, 2009.

Sevilla, J. Sola and J. "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems." in *Ieee Transactions On Nuclear Science, Vol. 44*. 1997.

Tiab, Alpheus O. Igbokoyi and Djebbar. *Well Test Analysis in Naturally Fractured Reservoirs using Elliptical Flow.* SPE, 2007.

Appendix A

Data Transformation Methods

In order to satisfy the assumption of normality, homogeneity of variance, or linearity, it is possible to increase the predictability of the original variables by transforming them into new variables. The following table summarizes various transformation methods considered in this study and their effects.

Table A: Summary of transformation methods

Method	Formula	Effects
Centering	$\tilde{x}_i = x_i - \bar{x}_i$	Focus on the differences and not the similarities in data
Auto scaling	$\tilde{x}_i = \frac{x_i - \bar{x}_i}{s_i}$	Inflation of the errors, gives equal importance to variables
Range calling	$\tilde{x}_i = \frac{x_i - \bar{x}_i}{x_{i_{max}} - x_{i_{min}}}$	Inflation of the errors, sensitive to outliers
Pareto Scaling	$\tilde{x}_i = \frac{x_i - \bar{x}_i}{\sqrt{s_i}}$	Reduce the relative importance of large values, but keep data structure partially intact, sensitive to large fold changes
Vast scaling	$\tilde{x}_i = \frac{(x_i - \bar{x}_i)}{s_i} \cdot \frac{\bar{x}_i}{s_i}$	Focus on small fluctuations, aim for robustness, not suited for large induced variation without group structure
Level scaling	$\tilde{x}_i = \frac{x_i - \bar{x}_i}{\bar{x}_i}$	Focus on relative response, inflation of errors
Log transformation	$\tilde{x}_{ij} = log_{10}(x_i)$	Correct for heteroscedasticity, make multiplicative models additive, (values with large relative standard deviation and zeros can be problematic, there may be data values which are not mathematically permissible)
Power transformation	$\tilde{x}_i = {x_i}^{\lambda}$	Reduce heteroscedasticity, no problem with small values, used for positive data, and choice for the power is arbitrary, used when symmetry or normality is desired in a data set

Inversion	$\tilde{x}_i = 1/x_i$	Inflate small values, deflate large values
Eigenvalues	$\tilde{x}_{abcd} = eig\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	Focus on the characteristic relationship among variables
Modified dimensionless time	$t_D^i = \frac{0.02637\ k_f}{(\varphi_m c_m + \varphi_f c_f)\ \mu_i\ r_w^2\ L_x\ SC}$	Presents several variables at the same time
Slopes of modified dimensionless pseudo-pressure drop against logarithm of time	$\frac{d\psi_D^i}{d\ log_{10}(t)} = \frac{h(\psi_n - \psi_{n+1})}{1.422 \times 10^6\ T\ q_{sc}(log_{10}(t_{n+1}) - log_{10}(t_n))}$	Captures real gas flow phenomena and slope change
Slopes of pressure against logarithm of time	$\frac{dp}{d\ log_{10}(t)} = \frac{p_{n+1} - p_n}{\log_{10}(t_{n+1}) - \log_{10}(t_n)}$	slope change

The mean is estimated as: $\bar{x}_i = \frac{1}{I}\sum_{i=1}^{I} x_i$ and the standard deviation is estimated as: $\boldsymbol{s}_i = \sqrt{\frac{\sum_{i=1}^{I}(x_i - \bar{x}_i)^2}{\boldsymbol{I-1}}}$. $\tilde{x}$ represents the data after different transformation steps [van den Berg et al, 2006].

Appendix B

Viscosity Interpolation

In this study, given the specific gravity of the gas, the pseudo-properties of natural gas are determined using Thomas, Hankinson, and Phillips correlations as shown below:

$$T_c = 170.491 + 307.344\,\gamma_g \quad \text{(B.1)}$$

$$p_c = 709.604 - 58.718\,\gamma_g \quad \text{(B.2)}$$

$$T_{pr} = (460 + T)/T_c \quad \text{(B.3)}$$

$$p_{pr} = p/p_c \quad \text{(B.4)}$$

where T_c, p_c, T_{pr}, p_{pr} represent pseudo-critical temperature, pseudo-critical pressure, pseudo-reduced temperature and pseudo-reduced pressure respectively. The corrections for sour gases and other non-hydrocarbon contents were not considered in this study based on the assumption that the reservoirs produce dry-sweet gases. The graphical correlations for estimating natural gases viscosity as a function of temperature, pressure, and gas gravity [Carr et al, 1954] were depicted in two separate graphs, where the ratio of viscosity at high pressure (μ_g) to the viscosity at the atmospheric pressure (μ_{g1}) was delineated as a function of reduced pressure and reduced temperature. As a convenient way of interpolation, the graphical correlations of viscosity ratio versus pseudo-reduced temperature and viscosity ratio versus pseudo-reduced pressure are merged into a surface of viscosity ratio shown in Table. Gas viscosity at atmospheric pressure and reservoir temperature (μ_{g1}) is calculated using Equation B.5. Finally, the viscosity of natural gas (μ_g) is obtained using Equation B.6.

$$\mu_{g1} = 1.2658 * 10^{-2} - 6.11823 * 10^{-3} * \gamma_g + 1.64574 * 10^{-3} * {\gamma_g}^2 + 1.64574 * 10^{-5} * T * SG^2 - 7.19221 * 10^{-7} * T * \gamma_g - 6.09046 * 10^{-7} * T * {\gamma_g}^2 \quad \text{(B.5)}$$

$$\mu_g = viscotsity\ ratio * \mu_{g1} \quad \text{(B.6)}$$

Table B: Tabulated viscosity ratio (μg/μg1)													
Ppr \ Tpr	1.05	1.1	1.15	1.2	1.3	1.4	1.5	1.6	1.75	2	2.25	2.5	3
0.1	1	1	1	1	1	1	1	1	1	1	1	1	1
0.2	1.012	1.011	1.01	1.009	1.008	1.007	1.006	1.005	1.004	1.003	1.002	1.001	1
0.3	1.025	1.023	1.021	1.019	1.017	1.015	1.013	1.011	1.009	1.007	1.005	1.003	1.001
0.4	1.05	1.043	1.036	1.03	1.027	1.024	1.021	1.018	1.015	1.012	1.009	1.006	1.003
0.5	1.075	1.065	1.055	1.045	1.04	1.035	1.03	1.025	1.021	1.017	1.013	1.009	1.005
0.6	1.1	1.086	1.073	1.06	1.054	1.048	1.042	1.036	1.03	1.024	1.018	1.012	1.007
0.7	1.145	1.12	1.095	1.07	1.063	1.056	1.049	1.042	1.035	1.028	1.021	1.015	1.009
0.8	1.195	1.15	1.12	1.085	1.075	1.067	1.059	1.051	1.043	1.035	1.027	1.019	1.011
0.9	1.285	1.195	1.145	1.11	1.1	1.089	1.078	1.067	1.056	1.045	1.034	1.023	1.013
1	1.415	1.225	1.175	1.135	1.12	1.1	1.1	1.07	1.065	1.055	1.04	1.025	1.015
1.2	1.76	1.435	1.28	1.195	1.155	1.135	1.12	1.095	1.09	1.06	1.045	1.03	1.02
1.4	2.285	1.7	1.42	1.285	1.225	1.185	1.15	1.12	1.11	1.07	1.055	1.04	1.025
1.6	2.865	2.07	1.59	1.425	1.285	1.235	1.185	1.15	1.125	1.08	1.065	1.05	1.03
1.8	3.29	2.465	1.85	1.57	1.36	1.28	1.22	1.18	1.145	1.095	1.075	1.06	1.035
2	3.65	2.8	2.16	1.75	1.46	1.335	1.26	1.215	1.165	1.11	1.085	1.065	1.04
3	4.76	3.85	3.225	2.6	2.02	1.69	1.5	1.385	1.28	1.205	1.145	1.105	1.06
4	5.5	4.655	3.975	3.35	2.56	2.11	1.785	1.595	1.435	1.29	1.21	1.155	1.085
6	6.46	5.72	5.03	4.38	3.5	2.79	2.325	2.03	1.77	1.5	1.34	1.245	1.14
8	7.15	6.5	5.82	5.125	4.185	3.38	2.82	2.425	2.095	1.725	1.485	1.36	1.205
10	7.68	7.06	6.385	5.74	4.755	3.86	3.23	2.77	2.375	1.955	1.665	1.485	1.265
15	8.65	8.1	7.41	6.75	5.79	4.79	4.06	3.49	2.99	2.48	2.085	1.83	1.495
20	9.37	8.88	8.18	7.5	6.5	5.41	4.61	4.025	3.5	2.925	2.46	2.15	1.75

Appendix C

Z Factor

The numerical approximation proposed by Dranchuk and Abu-Kasem is the widely accepted correlation for the gas compressibility factor (*Z*) for non-ideal gas, which is given in the form of an implicit function in terms of Z:

$$f(Z) = Z - b_1 Z^{-1} - b_2 Z^{-2} + b_3 Z^{-5} - (b_4 Z^{-2} + b_6 Z^{-4})\exp(-b_5 Z^{-2}) - 1 = 0 \quad \text{(C.1)}$$

where

$$b_1 = c\left(a_1 + \frac{a_2}{T_{pr}} + \frac{a_3}{T_{pr}^3} + \frac{a_4}{T_{pr}^4} + \frac{a_5}{T_{pr}^5}\right) \quad \text{(C.2)}$$

$$b_2 = c^2\left(a_6 + \frac{a_7}{T_{pr}} + \frac{a_8}{T_{pr}^2}\right) \quad \text{(C.3)}$$

$$b_3 = c^5 a_9\left(\frac{a_7}{T_{pr}} + \frac{a_8}{T_{pr}^2}\right) \quad \text{(C.4)}$$

$$b_4 = c^2 \frac{a_{10}}{T_{pr}^3} \quad \text{(C.5)}$$

$$b_5 = c^2 a_{11} \quad \text{(C.6)}$$

$$b_6 = b_4 b_5 \quad \text{(C.7)}$$

Coefficients α_1 to α_{11} are specified in Table, and c is a function of dimensionless pseudo-reduced pressure and temperature:

$$c = 0.27\frac{p_{pr}}{T_{pr}} \quad \text{(C.8)}$$

Table C: Coefficients for Dranchuk and Abu-Kasem (1975) approximation	
Coefficient	Value
α_1	0.3265
α_2	-1.0700
α_3	-0.5339
α_4	0.01569
α_5	-0.05165
α_6	0.5475
α_7	-0.7361
α_8	0.1844
α_9	0.1056
α_{10}	0.6134
α_{11}	0.7210

Equation C.1 can be solved iteratively using Newton-Raphson algorithm expressed in C.9.

$$Z_{k+1} = Z_k - \frac{f(Z_k)}{f^{'}(Z_k)} \tag{C.9}$$

Here, *k* is the iteration level, and *f'(Z)* is the derivative of *f(Z)* with respect to *Z* given by

$$f(Z) = 1 + b_1 Z^{-2} + 2b_2 Z^{-3} - 5b_3 Z^{-6} + (2b_4 Z^{-3} - 2b_4 b_5 Z^{-5} + 4b_6 Z^{-5} - 2b_5 b_6 Z^{-7})\exp(-b_5 Z^{-2}). \tag{C.10}$$

A good starting value for Z is provided by the explicit approximation of Papay as:

$$Z = 1 - \frac{3.52 p_{pr}}{10^{0.9813 T_{pr}}} + \frac{0.274 p_{pr}^2}{10^{0.8157 T_{pr}}}. \tag{C.11}$$

This Dranchuk and Abu-Kasem approximation closely resembles the Z factor values represented in graphical form by Standing and Katz in 1942. This method has been programmed into a Matlab subroutine '**`Z_fator_subroutine.m`**', where the ranges of pseudo-reduced pressure and pseudo-reduced temperature are given as:

$0.2 \leq p_{pr} \leq 30$ and $1.05 \leq T_{pr} \leq 3.0$

Appendix D

Computer Programs

This appendix contains listing of following programs:

1. Viscosity Interpolation Subroutine
2. Gas Compressibility Calculation Subroutine (Z factor)
3. Reservoir Proxy Generator
4. Compressibility of Gas
5. Production Rate Specification using ψ approach
6. PVT table for CMG inputs
7. Tabulation of generated reservoir data
8. Reservoir data into reservoir simulator
9. Physical accurability checker
10. Input and output data organization for the ANN
11. Example of Matlab Neural Network Toolbox

1. Viscosity Interpolation Subroutine

```
 function [VISG] = viscositysubroutine(TEM, SPG, pressure)
% SPG---specific gravity of the natural gas
% TEM---Temperature of the natural gas at reservoir condition
% pressure--- any desired pressure for calculating viscosity
% VISGR---viscosity ratio
% VISGU---viscosity at 1 atmospheric pressure
% VISG--- Viscosity of the natural gas
% Pseudocritical_pressure---Pseudocritical_pressure
% Pseudocritical_temperature---Pseudo-reduced pressure
% Ppr---pseudo-reduced pressure
% Tpr---pseudo-reduced temperature
% HIPRS--- Higher bound of pseudo-reduced pressure table
% LOPRS---lower bound of pseudo-reduced pressure table
% HITEM--- Higher bound of pseudo-reduced temperature table
% LOTEM---lower bound of pseudo-reduced temperature table
% PprTab--- pseudo-reduced pressure table for viscosity ratio
% TprTab--- Pseudo-reduced temperature table for viscosity ratio
% VisTab--- Tabulated viscosity ratio surface values on pseudo-reduced pressure and Pseudo-reduced temperature plane
Pseudocritical_pressure=709.604-58.718* SPG;
Pseudocritical_temperature=170.491+307.344* SPG;
TprTab=[1.05; 1.1; 1.15; 1.2; 1.3; 1.4; 1.5; 1.6; 1.75; 2; 2.25; 2.5; 3];
HITEM=max(TprTab); LOTEM=min(TprTab);
Ppr= pressure/Pseudocritical_pressure;
Tpr=(TEM+460)/Pseudocritical_temperature;
PprTab=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.2 1.4 1.6 1.8 2 3 4 6 8 10 15 20];
HIPRS=max(PprTab); LOPRS=min(PprTab);
VisTab=[

     1      1      1      1      1      1      1      1      1      1      1      1      1   ;
 1.012  1.011   1.01  1.009  1.008  1.007  1.006  1.005  1.004  1.003  1.002  1.001      1   ;
 1.025  1.023  1.021  1.019  1.017  1.015  1.013  1.011  1.009  1.007  1.005  1.003  1.001   ;
  1.05  1.043  1.036   1.03  1.027  1.024  1.021  1.018  1.015  1.012  1.009  1.006  1.003   ;
 1.075  1.065  1.055  1.045   1.04  1.035   1.03  1.025  1.021  1.017  1.013  1.009  1.005   ;
   1.1  1.086  1.073   1.06  1.054  1.048  1.042  1.036   1.03  1.024  1.018  1.012  1.007   ;
 1.145   1.12  1.095   1.07  1.063  1.056  1.049  1.042  1.035  1.028  1.021  1.015  1.009   ;
 1.195   1.15   1.12  1.085  1.075  1.067  1.059  1.051  1.043  1.035  1.027  1.019  1.011   ;
 1.285  1.195  1.145   1.11    1.1  1.089  1.078  1.067  1.056  1.045  1.034  1.023  1.013   ;
 1.415  1.225  1.175  1.135   1.12    1.1    1.1   1.07  1.065  1.055   1.04  1.025  1.015   ;
  1.76  1.435   1.28  1.195  1.155  1.135   1.12  1.095   1.09   1.06  1.045   1.03   1.02   ;
 2.285    1.7   1.42  1.285  1.225  1.185   1.15   1.12   1.11   1.07  1.055   1.04  1.025   ;
 2.865   2.07   1.59  1.425  1.285  1.235  1.185   1.15  1.125   1.08  1.065   1.05   1.03   ;
  3.29  2.465   1.85   1.57   1.36   1.28   1.22   1.18  1.145  1.095  1.075   1.06  1.035   ;
  3.65    2.8   2.16   1.75   1.46  1.335   1.26  1.215  1.165   1.11  1.085  1.065   1.04   ;
```

```
   4.76    3.85    3.225    2.6    2.02    1.69    1.5    1.385    1.28    1.205    1.145    1.105    1.06    ;
    5.5   4.655    3.975    3.35   2.56    2.11    1.785  1.595    1.435   1.29     1.21     1.155    1.085   ;
   6.46    5.72    5.03     4.38   3.5     2.79    2.325  2.03     1.77    1.5      1.34     1.245    1.14    ;
   7.15     6.5    5.82     5.125  4.185   3.38    2.82   2.425    2.095   1.725    1.485    1.36     1.205   ;
   7.68    7.06    6.385    5.74   4.755   3.86    3.23   2.77     2.375   1.955    1.665    1.485    1.265   ;
   8.65     8.1    7.41     6.75   5.79    4.79    4.06   3.49     2.99    2.48     2.085    1.83     1.495   ;
   9.37    8.88    8.18     7.5    6.5     5.41    4.61   4.025    3.5     2.925    2.46     2.15     1.75    ;]

VISGU=0.126585e-1-0.611823e-2*SPG+0.164574e-2*SPG^2+0.164574e-4*TEM-0.719221e-6*SPG*TEM-0.609046e-6*SPG^2*TEM;
    if Tpr>LOTEM && Tpr<HITEM && Ppr>LOPRS && Ppr<HIPRS
        % using spline interpolation method
        VISGR=interp2(TprTab,PprTab,VisTab,Tpr,Ppr,'spline');
        VISG=VISGU* VISGR;
    elseif Ppr<=LOPRS
        VISG= VISGU;
    end
end
```

2. Gas Compressibility Calculation Subroutine (Z factor)

```
function [Z] = Z_fator_subroutine(TEM, SPG, pressure)
% SPG-specific gravity of the natural gas
% TEM---Temperature of the natural gas at reservoir condition
% pressure--- any desired pressure for calculating viscosity
% Pseudocritical_pressure---Pseudo-critical pressure
% Pseudocritical_temperature---Pseudo-reduced pressure
% Ppr---pseudo-reduced pressure
% Tpr---pseudo-reduced temperature
% Dranchuk and Abu-Kassem (1975) approximation. Coefficient Value
aa1 =0.3265; aa2 =-1.0700; aa3= -0.5339; aa4 =0.01569; aa5 =-0.05165;
aa6 =0.5475; aa7 =-0.7361; aa8 =0.1844; aa9 =0.1056; aa10 =0.6134; aa11 =0.7210;
Pseudocritical_pressure=709.604-58.718* SPG;
Pseudocritical_temperature=170.491+307.344* SPG;
Ppr= pressure/Pseudocritical_pressure;
Tpr=(TEM+460)/Pseudocritical_temperature;
c=0.27*Ppr/Tpr;
b1=c*(aa1+ aa2/Tpr+ aa3/Tpr^3+ aa4/Tpr^4+ aa5/Tpr^5);
b2=c^2*(aa6+ aa7/Tpr+ aa8/Tpr^2);
b3=c^5*aa9*(aa7/Tpr+ aa8/Tpr^2);
b4=c^2*aa10/Tpr^3;
b5=c^2*aa11;
b6=b4*b5;

% Intitial Guess for Z factor
Z=1-3.52*Ppr/(10^0.9813*Tpr)+0.274*Ppr^2/(10^0.8157*Tpr);
```

```
% Initial Znew
Znew=0;
% Initializing the difference
delta=1;
% Newton-Rapson algorithm
while delta>0.000001
    fz=Z-b1*Z^-1-b2*Z^-2+b3*Z^-5-(b4*Z^-2+b6*Z^-4)*exp(-b5*Z^-2)-1;
    fzder=1+b1*Z^-2+2*b2*Z^-3-5*b3*Z^-6+(2*b4*Z^-3-2*b4*b5*Z^-5+4*b6*Z^-5-2*b5*b6*Z^-7)*exp(-b5*Z^-2);
    Z=Z-fz/fzder;
    delta=abs(Znew-Z);
    Znew=Z;
End
```

3. Reservoir Proxy Generator

```
clear
clc
field_size=10000;
num_case = 1000;  % number of cases
rw=0.25;
%% Range of reservoir data to be generated
gas_gravity=[0.56;0.9];
Temperature_range=[100;200];      %F
frac_space_range=[20;150];
k_matrix_range=[0.000001; 0.0001];
k_frac_range_i=[0.001; 1];
por_matrix_range=[0.01; 0.15];
por_frac_range=[0.0001; 0.005];
Earth_Pressuregradient_range=[0.55;0.99];
h_range=[20; 100];  % thickness
Initial_pressure_range=[1500;5000];
Surface_gas_rate_range=[1e6;4e6];
C_Matrix_range=[2e-6;15e-6];
C_Fracture_range=[72e-6;96e-6];

%% Random reservoir data within the range
Gas_specific_gravity= gas_gravity(1) + (gas_gravity(2)-gas_gravity(1))*rand(num_case,1);
T=Temperature_range(1)+(Temperature_range(2)-Temperature_range(1))*rand(num_case,1);
Initial_Pressure=(Initial_pressure_range(1) + (Initial_pressure_range(2)-Initial_pressure_range(1))*rand(num_case,1));
thickness=h_range(1) + (h_range(2)-h_range(1))*rand(num_case,1);
ReferenceDepth=Initial_Pressure./(Earth_Pressuregradient_range(1)+(Earth_Pressuregradient_range(2)-
Earth_Pressuregradient_range(1))*rand(num_case,1));
Gridtop=ReferenceDepth-thickness/2;
por_matrix=por_matrix_range(1) + (por_matrix_range(2)-por_matrix_range(1))*rand(num_case,1);
por_frac=por_frac_range(1) + (por_frac_range(2)-por_frac_range(1))*rand(num_case,1);
k_matrix=k_matrix_range(1) + (k_matrix_range(2)-k_matrix_range(1))*rand(num_case,1);
```

```
k_frac_i=k_frac_range_i(1) + (k_frac_range_i(2)-k_frac_range_i(1))*rand(num_case,1);
k_frac_j_composition=0.05*randi(19,1,num_case)';
k_frac_j=k_frac_i.*k_frac_j_composition;
k_frac=sqrt(k_frac_i.*k_frac_j);
frac_space=frac_space_range(1) + (frac_space_range(2)-frac_space_range(1))*rand(num_case,1);
Sealing_Capacity=5*randi([14 20],1,num_case)';
% Distance to the fault
R=100*randi([3 15],1,num_case)';
% angle between pricipale flow direction and fault line
fault_angle=5*randi([6 34],1,num_case)';
% matrix rock compressibility
C_Matrix=C_Matrix_range(1) + (C_Matrix_range(2)-C_Matrix_range(1))*rand(num_case,1);
% fracture rock compressibility
C_Fracture=C_Fracture_range(1) + (C_Fracture_range(2)-C_Fracture_range(1))*rand(num_case,1);
%% According to Warren and Root
% Storitivity ratio
w=por_frac.*C_Fracture./(por_frac.*C_Fracture+por_matrix.*C_Matrix);
% Shape factor using sugar cube model
Shape_factor=60./frac_space.^2;
% Inter-porosity flow parameter
lambda=Shape_factor.*k_matrix./k_frac*rw^2; % 10^-3 < lambda < 10^-9
```

4. Compressibility of Gas

```
for i=1:num_case
        dp=0.1; % pressure change
        SPG=Gas_specific_gravity(i,1);
        TEM=T(i,1);
        pressure=Initial_Pressure(i,1);
        [VISG] = viscositysubroutine(TEM, SPG, pressure);
        Visc_init(i,1)=VISG;
        [Z] = Z_fator_subroutine(TEM, SPG, pressure);
        Z_init(i,1)=Z;
        [Zn] = Z_fator_subroutine(TEM, SPG, pressure+dp);
        cg(i,1)=1/pressure-(Zn-Z)/(Z*dp); %Gas compressibility
end
```

5. Production Rate Specification using ψ approach

```
D180days=k_frac*180*24./((por_matrix.*C_Matrix+por_frac.*C_Fracture).*Visc_init*rw^2);
rd=exp(0.5*(log(tD180days)+0.809))*rw;
tD=k_frac*5*365*24./((por_matrix.*C_Matrix+por_frac.*C_Fracture).*Visc_init*rw^2);
%% Making sure the measurable pressure transient will not reach the outer boundary in 10 years
    PwD=0.5*(log(tD)+0.809); % Dimensionless pressure drop at the wellbore
    BHPmin=14.7; % Minimum allowable bottom-hole pressure
```

```
    P_smallincrement=BHPmin*0.1/14.7;
    P_largeincrement=5;
for i=1:num_case
    ct1=0;
    ct=0;
    x=0;
    while x < Initial_Pressure(i,1)
        ct=ct+1;
        SPG=Gas_specific_gravity(i,1);
        TEM=T(i,1);
        pressure=x;
        [VISG] = viscositysubroutine(TEM, SPG, pressure);
        [Z] = Z_fator_subroutine(TEM, SPG, pressure);
        p_miu_z(ct,1)=2*x/(VISG*Z); % ψ value calculation

        p(ct,1)=x; % recorded pressure values in intervals
        % bottom-hole pressure can be set to any desired pressure lager than 14.7 and smaller than Initial pressure
        if x<BHPmin
            x=x+P_smallincrement; % small step
            ct1=ct1+1;
        else
            x=x+P_largeincrement; % large step
        end
    end
        % trapezoidal integration of ψ from 0 to 14.7 psi
        ThyBHPmin(i,1)=P_smallincrement*trapz(p_miu_z(1:ct1,1));
        % trapezoidal integration of ψ from 14.7 psi to Initial pressure of the reservoir
        Thyi(i,1)=P_smallincrement*trapz(p_miu_z(1:ct1,1))+P_largeincrement*trapz(p_miu_z(ct1:ct,1));
        % Specifying production rate MMscf/day
        qsc(i,1)=(Thyi(i,1)-ThyBHPmin(i,1))*k_frac(i,1)*thickness(i,1)/(1.422e6*(T(i,1)+520)*PwD(i,1));
end
```

6. PVT table for CMG inputs

```
PVT_data_Pressure_range=[14.7,(120:120:6000)]';
for i=1:num_case
    for j=1:length(PVT_data_Pressure_range)
        SPG=Gas_specific_gravity(i,1);
        TEM=T(i,1);
        pressure=PVT_data_Pressure_range(j,1);
        [VISG] = viscositysubroutine(TEM, SPG, pressure);
        ViscTable(j,1)=VISG;
        [Z] = Z_fator_subroutine(TEM, SPG, pressure);
        Zfactor(j,1)=Z;
    end
```

```
    PVTZ(:,i)=Zfactor;
    PVTV(:,i)=ViscTable;
end
```

7. Tabulating generated reservoir data

```
CMGfeed(:,1)=Initial_Pressure;
CMGfeed(:,2)=por_matrix;
CMGfeed(:,3)=por_frac;
CMGfeed(:,4)=k_matrix;
CMGfeed(:,5)=k_frac_i;
CMGfeed(:,6)=k_frac_j;
CMGfeed(:,7)=frac_space;
CMGfeed(:,8)=T;
CMGfeed(:,9)=thickness;
CMGfeed(:,10)=Gridtop;
CMGfeed(:,11)=ReferenceDepth;
CMGfeed(:,12)=Gas_specific_gravity;
CMGfeed(:,13)=qsc;
CMGfeed(:,14)=w;
CMGfeed(:,15)=lambda;
CMGfeed(:,16)=C_Matrix;
CMGfeed(:,17)=C_Fracture;
CMGfeed(:,18)=cg;
CMGfeed(:,19)=Z_init;
CMGfeed(:,20)=Visc_init;
CMGfeed(:,21)=Sealing_Capacity;
CMGfeed(:,22)=R;
CMGfeed(:,23)=fault_angle;
%% Saving generated data
save CMGfeed.mat CMGfeed
```

8. Reservoir data into reservoir simulator

```
a=field_size/2; b=-a;
k_tan=tan((fault_angle-90)*pi/180);
X0=(a+a*k_tan.^2+sqrt((1+k_tan.^2).*R.^2))./(1+k_tan.^2); Y0=k_tan.*(X0-a)+b;
Fault_slope=tan(fault_angle*pi/180);
X1=-Y0./Fault_slope+X0;
for i=1:num_case
if X1(I,1)<0
   X1(I,1)=0; end
if X1(I,1)>field_size
   X1(I,1)=field_size; end
end
Y1=Fault_slope.*(X1-X0)+Y0;
X2=(-field_size-Y0)./Fault_slope+X0;
```

```
for i=1:num_case
if X2(I,1)<0
   X2(I,1)=0; end
if X2(I,1)>field_size
   X2(I,1)=field_size; end
end
Y2=Fault_slope.*(X2-X0)+Y0;
Fault_length=sqrt((X2-X1).^2+(Y2-Y1).^2);
X1=abs(X1); X2=abs(X2); Y1=abs(Y1); Y2=abs(Y2);
%% Writing dat file
dx=200; dy=dx; refinesize=50;
nam2 = 'BHP.bat';
fidbatch = fopen(nam2,'w');


for i=1:size(CMGfeed,1)
    k=Fault_slope(I,1);
    theta=fault_angle(I,1);
    x1=X1(I,1);
    x2=X2(I,1);
    y1=Y1(I,1);
    y2=Y2(I,1);
    if theta==90
        blocknum_left=round(min(x1,x2)/dx);
        refine_middleblock_x=round(dx/refinesize);
        blocknum_right=round((field_size-max(x1,x2))/dx);


        blocknum_up=round(field_size/dy);
        refine_middleblock_y=0;
        blocknum_down=0;


    else
        blocknum_left=round(min(x1,x2)/dx);
        refine_middleblock_x=round(abs(x2-x1)/refinesize);
        blocknum_right=round((field_size-max(x1,x2))/dx);


        blocknum_up=round(min(y1,y2)/dy);
        refine_middleblock_y=round(abs(y2-y1)/refinesize);
        blocknum_down=round((field_size-max(y1,y2))/dy);
    end
    blocknum_x=blocknum_left+refine_middleblock_x+blocknum_right;
    blocknum_y=blocknum_up+refine_middleblock_y+blocknum_down;
    %% placing the well block at the centre
    if min(x1,x2)<field_size/2 && max(x1,x2)<field_size/2
        wellblock_x=blocknum_left+refine_middleblock_x+blocknum_right-round(field_size/2/dx);
    elseif min(x1,x2)<field_size/2 && max(x1,x2)>field_size/2
        wellblock_x=blocknum_left+round((field_size/2-min(x1,x2))/refinesize);
```

```
elseif min(x1,x2)>field_size/2 && max(x1,x2)>field_size/2
    wellblock_x=round(field_size/2/dx);
end

if theta==90
    wellblock_y=round(field_size/2/dy);
else

    if min(y1,y2)<field_size/2 && max(y1,y2)<field_size/2
        wellblock_y=blocknum_up+refine_middleblock_y+blocknum_down-round(field_size/2/dx);
    elseif min(y1,y2)<field_size/2 && max(y1,y2)>field_size/2
        wellblock_y=blocknum_up+round((field_size/2-min(y1,y2))/refinesize);
    elseif min(y1,y2)>field_size/2 && max(y1,y2)>field_size/2
        wellblock_y=round(field_size/2/dy);
    end
end

if theta==90
    for xx=1:blocknum_y
        fault_x(x,1)=round(refine_middleblock_x/2);
        fault_y(x,1)=xx;
    end

else
    for x=1㉿refine_middleblock_x)*refinesize
        fault_x(x,1)=fix(x/refinesize);
        fault_y(x,1)=fix(abs(k*x)/refinesize);
    end
end



faultblock_x= blocknum_left+fault_x;
if k<0
    faultblock_y=blocknum_up+fault_y;
elseif k>0
    faultblock_y=blocknum_up+refine_middleblock_y-fault_y;
end
for g=1:length(faultblock_x)
    if faultblock_x(g,1)<=blocknum_left
        faultblock_x(g,1)=blocknum_left+1;
    end
    if faultblock_x(g,1)>=blocknum_left+refine_middleblock_x
        faultblock_x(g,1)=blocknum_left+refine_middleblock_x;
    end
    if faultblock_y(g,1)<=blocknum_up
```

```
            faultblock_y(g,1)= blocknum_up+1;
        end
        if faultblock_y(g,1)>=blocknum_up+refine_middleblock_y
            faultblock_y(g,1)=blocknum_up+refine_middleblock_y;
        end
    end
    fault=unique([faultblock_x faultblock_y],'rows');
    %% Constants in CMG files
    BWI=1; % Water formation volume factor at P=Prw (BWI keyword) 1.00731
    CVW=1.3e-6; % cp/psi Pressure dependence of water viscosity (viscosity units/pressure units).Usually CVW is 0
    CW=3.17716e-006; % 1/psi CW indicates the input of water compressibility (for a PVT region).Max 6.89e-3 Min 0
    DENSITY_WATER=62.3805; % lb/ft3
    REFPW=14.696; % psi Reference pressure
    VWI=0.715187;  % Water viscosity at reference pressure
    DWGC=11000; % ft gas water contact set deep enough that simulation will not produce water/
    %we do not change this parameter throughout the reservoir.
    %% Water saturation relative permeability of gas
SWT=[0.150  0.0; 0.175  0.0; 0.2 0.0; 0.225 0.0; 0.250 0.0; 0.275 0.000103; 0.300 0.000668; 0.325 0.001995; 0.350
0.004338; 0.375 0.007925; 0.400 0.012965; 0.425 0.019658; 0.450 0.028191; 0.475 0.038746; 0.500 0.051496; 0.525
0.066609; 0.550 0.084248; 0.575 0.104573; 0.600 0.127737; 0.625 0.153893; 0.650 0.183188; 0.675 0.215767; 0.700
0.251773; 0.725 0.291345; 0.750 0.334621; 0.775 0.381737; 0.8 0.432827; 0.825 0.488020; 0.85 0.547448; 0.875 0.611238;
0.9 0.679518; 0.925 0.75241; 0.950 0.830041; 0.975 0.912530; 1.0 1.0];
    %% Gas saturation relative permeability of gas
SGT=[0.0 0.0; 0.025 0.0; 0.050  0.0; 0.075  0.0; 0.1    0.000001; 0.125 0.000007; 0.150 0.000031; 0.175 0.000093; 0.200
0.000232; 0.225   0.000501; 0.250 0.000977; 0.275 0.001760; 0.300 0.002980; 0.325 0.004800; 0.350 0.007416; 0.375
0.011065; 0.400 0.016028; 0.425 0.022631; 0.450   0.031250; 0.475 0.042315; 0.500 0.056314; 0.525 0.073794; 0.550
0.095367; 0.575 0.121716; 0.600 0.153590; 0.625 0.191818; 0.650 0.237305; 0.675   0.291038; 0.700 0.354093; 0.725
0.427631; 0.750 0.512909; 0.775   0.611280; 0.800 0.724196; 0.825 0.853215; 0.850 1.0];
  z=1;
  nam = ['BHP' num2str(i) '.dat'];
    fid = fopen(nam,'w');
    fprintf(fid,'RESULTS SIMULATOR IMEX 200900\n');
    fprintf(fid,'\n');
    fprintf(fid,'INUNIT FIELD\n');
    fprintf(fid,'WSRF WELL 1\n');
    fprintf(fid,'WSRF GRID TIME\n');
    fprintf(fid,'WSRF SECTOR TIME\n');
    fprintf(fid,'OUTSRF WELL LAYER NONE\n');
    fprintf(fid,'OUTSRF RES ALL\n');
    fprintf(fid,'OUTSRF GRID SG SW PRES WINFLUX\n'); % Extra outputs for gas well is omitted
    fprintf(fid,'WPRN GRID 0 \n');
    fprintf(fid,'OUTPRN GRID NONE \n');
    fprintf(fid,'OUTPRN RES NONE \n');
    fprintf(fid,'**$  Distance units: ft \n');
    fprintf(fid,'RESULTS XOFFSET           0.0000\n');
    fprintf(fid,'RESULTS YOFFSET           0.0000\n');
```

```
    fprintf(fid,'RESULTS ROTATION           0.0000  **$  (DEGREES)\n');
    fprintf(fid,'RESULTS AXES-DIRECTIONS 1.0 -1.0 1.0\n');
    fprintf(fid,'**$ ********************************************************************************\n');
    fprintf(fid,'**$ Definition of fundamental 103odeling103 grid\n');
    fprintf(fid,'**$ ********************************************************************************\n');
    fprintf(fid,'GRID VARI %u %u %u \n',blocknum_x,blocknum_y,z);
    fprintf(fid,'KDIR DOWN \n');
    fprintf(fid,'DI IVAR  \n');
    if blocknum_left==0 && blocknum_right==0 && refine_middleblock_x~=0
        fprintf(fid,' %d%c%g \n', refine_middleblock_x,'*',refinesize);
    end
    if blocknum_right==0 && blocknum_left~=0 && refine_middleblock_x~=0
        fprintf(fid,' %d%c%g %d%c%g \n',blocknum_left,'*',dx, refine_middleblock_x,'*',refinesize);
    end
    if blocknum_right~=0 && blocknum_left==0 &&  refine_middleblock_x~=0
        fprintf(fid,' %d%c%g %d%c%g \n',refine_middleblock_x,'*',refinesize, blocknum_right,'*',dx);
    end
    if blocknum_left~=0 && refine_middleblock_x~=0 && blocknum_right~=0
        fprintf(fid,' %d%c%g %d%c%g %d%c%g \n',blocknum_left,'*',dx, refine_middleblock_x,'*',refinesize,
blocknum_right,'*',dx);
    end
    if blocknum_left~=0 && refine_middleblock_x==0 && blocknum_right==0
        fprintf(fid,' %d%c%g \n',blocknum_left,'*',dx);
    end
    fprintf(fid,'DJ JVAR \n');
    if blocknum_up==0 && blocknum_down==0 && refine_middleblock_y~=0
        fprintf(fid,' %d%c%g \n',refine_middleblock_y,'*',refinesize);
    end
    if blocknum_up==0 && blocknum_down~=0 && refine_middleblock_y~=0
        fprintf(fid,' %d%c%g %d%c%g \n',refine_middleblock_y,'*',refinesize, blocknum_down,'*',dx);
    end
    if blocknum_up~=0 && blocknum_down==0 && refine_middleblock_y~=0
        fprintf(fid,' %d%c%g %d%c%g \n',blocknum_up,'*',dx, refine_middleblock_y,'*',refinesize);
    end
    if blocknum_up~=0 && blocknum_down~=0 && refine_middleblock_y~=0
        fprintf(fid,' %d%c%g %d%c%g %d%c%g \n',blocknum_up,'*',dx, refine_middleblock_y,'*',refinesize,
blocknum_down,'*',dx);
    end
    if blocknum_up~=0 && blocknum_down==0 && refine_middleblock_y==0
        fprintf(fid,' %d%c%g \n',blocknum_up,'*',dx);
    end
    fprintf(fid,'DK ALL \n');
    fprintf(fid,' %d%c%g \n',z*blocknum_x*blocknum_y,'*',thickness(I,1));
    fprintf(fid,'DTOP \n');
    % Grid top is set at the beginning of the progamme
    fprintf(fid,' %d%c%d \n',blocknum_x*blocknum_y,'*',Gridtop(I,1));
```

```
    fprintf(fid,'DUALPOR \n');
    fprintf(fid,'SHAPE WR \n');
    fprintf(fid,'**$ Property: NULL Blocks  Max: 1  Min: 1 \n');
    fprintf(fid,'**$  0 = null block, 1 = active block \n');
    fprintf(fid,'NULL MATRIX CON            1 \n');
    fprintf(fid,'**$ Property: NULL Blocks  Max: 1  Min: 1 \n');
    fprintf(fid,'**$  0 = null block, 1 = active block \n');
    fprintf(fid,'NULL FRACTURE CON            1 \n');
    % matrix porosity
    fprintf(fid,'**$ Property: Porosity  Max: %g  Min: %g \n',por_matrix(I,1),por_matrix(I,1));
    fprintf(fid,'POR MATRIX CON          %f \n',por_matrix(I,1));
    % Fracture porosity
    fprintf(fid,'**$ Property: Porosity  Max: %g  Min: %g \n',por_frac(I,1),por_frac(I,1));
    fprintf(fid,'POR FRACTURE CON          %g \n',por_frac(I,1));

    % I direction matrix permeanility
    fprintf(fid,'**$ Property: Permeability I (md)   Max: %g  Min: %g \n',k_matrix(I,1),k_matrix(I,1));
    fprintf(fid,'PERMI MATRIX CON          %g \n',k_matrix(I,1));

    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
Sealing_Capacity(I,1)/100);
    end
    % I direction fracture permeability
    fprintf(fid,'**$ Property: Permeability I (md)   Max: %g  Min: %g \n',k_frac_i(I,1),k_frac_i(I,1));
    fprintf(fid,'PERMI FRACTURE CON          %g \n',k_frac_i(I,1));
    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
Sealing_Capacity(I,1)/100);
    end
    % J direction matrix permeanility
    fprintf(fid,'**$ Property: Permeability J (md)   Max: %g  Min: %g \n',k_matrix(I,1),k_matrix(I,1));
    fprintf(fid,'PERMJ MATRIX CON         %g \n',k_matrix(I,1));
    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
Sealing_Capacity(I,1)/100);
    end
    % J direction fracture permeability
    fprintf(fid,'**$ Property: Permeability J (md)   Max: %g  Min: %g \n',k_frac_j(I,1),k_frac_j(I,1));
    fprintf(fid,'PERMJ FRACTURE CON          %d\n',k_frac_j(I,1));
    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
```

```
Sealing_Capacity(I,1)/100);
    end
    % K direction matrix permeanility
    fprintf(fid,'**$ Property: Permeability K (md)   Max: %g  Min: %g \n',0.000001,0.000001);
    fprintf(fid,'PERMK MATRIX CON        %g \n',0.000001);
    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
Sealing_Capacity(I,1)/100);
    end
    % K direction fracture permeanility
    fprintf(fid,'**$ Property: Permeability K (md)   Max: %g  Min: %g \n',0.0001,0.0001);
    fprintf(fid,'PERMK FRACTURE CON        %g \n',0.0001);
    fprintf(fid,'*MOD \n');
    for r=1:length(fault)
        fprintf(fid,'%d:%d       %d:%d       1:1     = %g\n',fault(r,1),fault(r,1),fault(r,2),fault(r,2),1-
Sealing_Capacity(I,1)/100);
    end
    % Fracture spacing I
    fprintf(fid,'**$ Property: Fracture Spacing I (ft)   Max: %g  Min: %g \n',frac_space(I,1),frac_space(I,1));
    fprintf(fid,'DIFRAC CON        %g \n',frac_space(I,1));
    % Fracture spacing J
    fprintf(fid,'**$ Property: Fracture Spacing J (ft)   Max: %g  Min: %g \n',frac_space(I,1),frac_space(I,1));
    fprintf(fid,'DJFRAC CON        %g \n',frac_space(I,1));
    % Fracture spacing K
    fprintf(fid,'**$ Property: Fracture Spacing K (ft)   Max: %g  Min: %g \n',frac_space(I,1),frac_space(I,1));
    fprintf(fid,'DKFRAC CON        %g \n',frac_space(I,1));
    fprintf(fid,'**$ Property: Pinchout Array  Max: 1  Min: 1\n');
    fprintf(fid,'**$  0 = pinched block, 1 = active block\n');
    fprintf(fid,'PINCHOUTARRAY CON            1\n');
    % Fracture rock compressibility
    fprintf(fid,'CPOR FRACTURE            %g \n', C_Fracture(I,1));
    % Matrix rock compressibility
    fprintf(fid,'CPOR MATRIX            %g \n', C_Matrix(I,1));
    % DPCONNECT keyword allows the grid module to produce extra connection between blocks in a dual
porosity/permeability
    % system when either the matrix or the fracture is missing from one of the two blocks being connected.
    Fprintf(fid,'DPCONNECT 1 \n');
    fprintf(fid,'MODEL GASWATER \n');
    fprintf(fid,'TRES %d\n', T(i));   % TRES= reservoir temperature
    fprintf(fid,'PVTG ZG 1\n'); % *GASWATER=PVTG option Use a two phase gas and water model, with no oil phase
105odeling.
    % Only two equations are solved simultaneously.
    % Gas PVT properties are entered using the *PVTG option. Krow, Krog, So, Pb, Co and Cvo are not input.
    % PCGW, if required, is input on the *SWT table and is properly handled in both *BLOCK_CENTER and *DEPTH_AVE
*TRANZONE initialization methods.
```

```
Fprintf(fid,'\n');
% PVT=PVT_rtable' is done in order to rotate the PVT_rtable to match the fprintf array writing sequence
% which starts from first row fill '%g  %f %f\n' fills first raw into first column
fprintf(fid,'**$           p              Z          visg\n');
for nn=1:length(PVT_data_Pressure_range)
    fprintf(fid,'           %g           %g       %g\n',[PVT_data_Pressure_range(nn,1) PVTZ(nn,i) PVTV(nn,i)]);
end
fprintf(fid,'BWI %g \n',BWI);
fprintf(fid,'CVW %g \n',CVW);
fprintf(fid,'CW %g \n',CW);
fprintf(fid,'DENSITY WATER %g \n',DENSITY_WATER);
fprintf(fid,'REFPW %g \n',REFPW);
fprintf(fid,'VWI %g \n',VWI);
fprintf(fid,'GRAVITY GAS %g \n',Gas_specific_gravity(i));

fprintf(fid,'ROCKFLUID\n');
fprintf(fid,'RPT 1\n');
fprintf(fid,'**$        Sw         krw\n');
fprintf(fid,'SWT\n');
for mm=1:length(SWT)
    fprintf(fid,'           %g           %g \n',SWT(mm,1),SWT(mm,2));
end
fprintf(fid,'**$        Sg         krg \n');
fprintf(fid,'SGT\n');
for ll=1:length(SGT)
    fprintf(fid,'           %g           %g \n',SGT(ll,1),SGT(ll,2));
end
fprintf(fid,'INITIAL\n');
fprintf(fid,'VERTICAL DEPTH_AVE WATER_GAS EQUIL NOTRANZONE\n');
fprintf(fid,'\n');
fprintf(fid,'REFDEPTH %g \n',ReferenceDepth(i));
fprintf(fid,'REFPRES %g \n',Initial_Pressure(i)); % Reference pressure is calculated using pressure gradient
fprintf(fid,'DWGC %d\n',DWGC); % Gas water contact
fprintf(fid,'\n');
%   fprintf(fid,'DATUMDEPTH 3100 INITIAL\n');
%% Numerical subroutine
fprintf(fid,'NUMERICAL\n');
fprintf(fid,'DTMAX 300\n');
fprintf(fid,'DTMIN 0.0000001\n');
fprintf(fid,'SOLVER AIMSOL\n');
fprintf(fid,'RUN\n');
fprintf(fid,'DATE 2011 9 10.00000\n');
%% Well entry starts here
fprintf(fid,'%s\n','WELL  ''Well-1''');
fprintf(fid,'%s\n','PRODUCER ''Well-1''');
fprintf(fid,'OPERATE  MAX  STG  %g  CONT\n',qsc(I,1)*1e6);
```

```
    %    fprintf(fid,'MONITOR  MIN  BHP  1000.  SHUTIN\n');
    fprintf(fid,'**$          rad  geofac  wfrac  skin\n');
    fprintf(fid,'GEOMETRY  K  0.25  0.37  1.  0.\n');
    fprintf(fid,'%s\n','PERF  GEO  ''Well-1''');
    fprintf(fid,'**$ UBA     ff  Status  Connection     \n');
    fprintf(fid,'%g %g %g %s\n',    wellblock_x, wellblock_y, z,'  1.  OPEN    FLOW-TO ''SURFACE''');
    fprintf(fid,'   \n');
    for n=1:10:60
        fprintf(fid,'*TIME %g\n',n/60/24);
    end
    for n=2:23
        fprintf(fid,'*TIME %g\n',n/24);
    end
    for n=1:30
        fprintf(fid,'*TIME %g\n',n);
    end
    for n=1:66
        fprintf(fid,'*TIME %g\n',30+5*n);
    end
    for n=1:72
        fprintf(fid,'*TIME %g\n',360+10*n);
    end
    for n=1:84
        fprintf(fid,'*TIME %g\n',1080+30*n);
    end
    fprintf(fid,'STOP   \n');
    fclose(fid);
% Locate the folder containing mx201010.exe or proper version of which then copy the directory to following address
    fprintf(fidbatch,'%s %s\n','"C:\Program Files (x86)\CMG\IMEX\2010.10\Win_x64\EXE\mx201010.exe" -f', nam);
end
 fclose(fidbatch);


%% Data retrieve
 fordata = ('CMG_BHP_RWD.bat');
 fidrwd = fopen(fordata,'wt');
 for i=1:size(CMGfeed,1)
    numb = num2str(i);
    dataext = ['BHP' numb '.rwd' ];
    fid = fopen(dataext,'wt');
    numb = num2str(i);
    fprintf(fid,'%s','FILE ''BHP',numb);
    fprintf(fid,'%s','.irf''');
    fprintf(fid,'\nLINES-PER-PAGE 10000\n');
    fprintf(fid,'\nTIME ON\n');
    fprintf(fid,'\n*TIMES-FOR\n');
    for n=1:10:60
```

```
        fprintf(fid,' %g\n',n/60/24);
    end
    for n=2:23
        fprintf(fid,' %g\n',n/24);
    end
    for n=1:30
        fprintf(fid,' %g\n',n);
    end
    for n=1:66
        fprintf(fid,' %g\n',30+5*n);
    end
    for n=1:72
        fprintf(fid,' %g\n',360+10*n);
    end
    for n=1:84
        fprintf(fid,' %g\n',1080+30*n);
    end
    fprintf(fid,'SPREADSHEET\n');     %GIVES ONLY ONE TABLE AND STRING DATA IS WRITTEN ONLY ONCE(FIVE LINES)
    fprintf(fid,'TABLE-FOR\n');
    fprintf(fid,'%s\n','COLUMN-FOR *PARAMETERS ''Well Bottom-hole Pressure'' *WELLS ''well-1''');
    %*PARAMETERS 'Well Bottom-hole Pressure' *WELLS 'Well-1'
    fprintf(fid,'%s\n','TABLE-END');
    fclose(fid);
    % Locate the folder containing report.exe then copy the directory to following address
    fprintf(fidrwd,'%s','call "C:\Program Files (x86)\CMG\BR\2010.12\Win_x64\EXE\report.exe" -f "BHP');
    fprintf(fidrwd,num2str(i));
    fprintf(fidrwd,'%s','.rwd"');
    fprintf(fidrwd,'%s',' -o "BHP',numb,'.txt"');
    fprintf(fidrwd,'\n');
end
fclose(fidrwd);
fclose('all');
% beep
Execute BHP.bat. This may take several hours to simulate all the constructed reservoirs in orderly manner.
When BHP.bat has completed simulating all the reservoirs, please run CMG_BHP_RWD.bat to extract Bottom-hole pressure of
each reservoir by generating BHP*.txt files.
```

9. Physical accurability checker

```
clear
clc
load CMGfeed.mat
load qscnew.mat
load qscmodified.mat
% file number to be modified
i=501;
correctionfactor=1.15;
```

```
qsc_correction=correctionfactor*qscnew(i);
qscmodified(i,1)= qsc_correction;
save qscmodified.mat qscmodified
str = ['OPERATE  MAX  STG  ',num2str(qsc_correction),'  CONT                                                    '];
dataext = ['BHP' num2str(i) '.dat' ];
dataextnew = ['BHP' num2str(i) 'new.dat' ];
%%%---------------Rewriting the corrected prodation rate to dat file
fid=fopen(dataext) ; % the original file
fidd=fopen(dataextnew,'w') ; % the new file
while ~feof(fid) ; % reads the original till last line
    tline=fgets(fid) ; %
    [m s e]=regexp(tline, 'STG','match','start','end');
    if isempty(m)==0
        fwrite(fidd,str)
    else
        fwrite(fidd,tline) ;
    end
end
fclose all ;
%%% --------------------END %%%%%%%%%%%%%%
fid = fopen(['BHP' num2str(i) 'new.rwd' ],'wt');
numb = num2str(i);
fprintf(fid,'%s','FILE ''BHP',num2str(i));
fprintf(fid,'%s','new.irf''');
fprintf(fid,'\nLINES-PER-PAGE 10000\n');
fprintf(fid,'\nTIME ON\n');
fprintf(fid,'\n*TIMES-FOR\n');
for n=1:10:60
    fprintf(fid,' %g\n',n/60/24);
end
for n=2:23
    fprintf(fid,' %g\n',n/24);
end
for n=1:30
    fprintf(fid,' %g\n',n);
end
for n=1:66
    fprintf(fid,' %g\n',30+5*n);
end
for n=1:72
    fprintf(fid,' %g\n',360+10*n);
end
for n=1:84
    fprintf(fid,' %g\n',1080+30*n);
end
fprintf(fid,'SPREADSHEET\n');     %GIVES ONLY ONE TABLE AND STRING DATA IS WRITTEN ONLY ONCE(FIVE LINES)
```

```
fprintf(fid,'TABLE-FOR\n');
fprintf(fid,'%s\n','COLUMN-FOR *PARAMETERS ''Well Bottom-hole Pressure'' *WELLS ''well-1''');
fprintf(fid,'%s\n','TABLE-END');
fclose(fid);
nam2 = 'BHPnew.bat';
fidbatch = fopen(nam2,'w');
fprintf(fidbatch,'%s %s\n','"C:\Program Files (x86)\CMG\IMEX\2010.10\Win_x64\EXE\mx201010.exe" -f', dataextnew);
fprintf(fidbatch,'\n');
fprintf(fidbatch,'%s','call "C:\Program Files (x86)\CMG\BR\2010.12\Win_x64\EXE\report.exe" -f "BHP');
fprintf(fidbatch,num2str(i));
fprintf(fidbatch,'%s','new.rwd"');
fprintf(fidbatch,'%s',' -o "BHP',num2str(i),'new.txt"');
fclose('all');
!BHPnew.bat
fileName=['BHP',num2str(i),'new.txt'];
A=importdata(fileName);
B = getfield(A,'data');
C=B(:,2);
% by looking at the last value in bottom-hole pressure reaching the desired minimum one can adjust the production rate.
BHPlast=C
```

10. Input and output data organization for the ANN

```
load CMGfeed.mat
Initial_Pressure=CMGfeed(:,1);
por_matrix=CMGfeed(:,2)*100;
por_frac=CMGfeed(:,3)*100;
k_matrix=CMGfeed(:,4);
k_frac_i=CMGfeed(:,5);
k_frac_j=CMGfeed(:,6);
frac_space=CMGfeed(:,7);
T=CMGfeed(:,8);
thickness=CMGfeed(:,9);
Gridtop=CMGfeed(:,10);
ReferenceDepth=CMGfeed(:,11);
Gas_specific_gravity=CMGfeed(:,12);
qsc=CMGfeed(:,13);
w=CMGfeed(:,14);
lambda=CMGfeed(:,15);
C_Matrix=CMGfeed(:,16);
C_Fracture=CMGfeed(:,17);
cg=CMGfeed(:,18);
Z_init=CMGfeed(:,19);
Visc_init=CMGfeed(:,20);
Sealing_Capacity=CMGfeed(:,21);
L=CMGfeed(:,22);
```

```
fault_angle=CMGfeed(:,23);

rw=0.25;

stepup=1;

for i = 1:500
    fileName=['BHP',num2str(i),'.txt'];

    A=importdata(fileName);

    B = getfield(A,'data');

    ct=0;

    for j=1:stepup:length(B)-1
        ct=ct+1;

        DelP(ct,1)=B(j+stepup,2)-B(j,2);

        Delt(ct,1)=log10(24*B(j+stepup,1))-log10(24*B(j,1));

    end
    DelP_Delt(:,i)=DelP./Delt;

    Time=B(:,1);

    k_frac(i,1)=sqrt(k_frac_i(i,1)*k_frac_j(i,1));



      inputs(:,i)=[-DelP_Delt(:,i);

         C_Matrix(i,1); C_Fracture(i,1); cg(i,1); (C_Matrix(i,1)+C_Fracture(i,1)+cg(i,1));

         C_Matrix(i,1)^0.5; C_Fracture(i,1)^0.5;cg(i,1)^0.5;(C_Matrix(i,1)+C_Fracture(i,1)+cg(i,1))^0.5;

         C_Matrix(i,1)^0.02; C_Fracture(i,1)^0.02; cg(i,1)^0.02; (C_Matrix(i,1)+C_Fracture(i,1)+cg(i,1))^0.02;

         Gas_specific_gravity(i,1);Gas_specific_gravity(i,1)^0.5;

         Visc_init(i,1);Visc_init(i,1)^0.5;

         Z_init(i,1);Z_init(i,1)^0.5;

         qsc(i,1);qsc(i,1)^0.5;1/qsc(i,1);

         T(i,1);

         thickness(i,1);

         Initial_Pressure(i,1)];


     k_frac(i,1)=sqrt(k_frac_i(i,1)*k_frac_j(i,1));


       targets(:,i)=[k_frac_j(i,1)/k_frac_i(i,1)*100;

                  k_frac_i(i,1)*100;

                  k_frac_j(i,1)*100;

                  k_frac(i,1)*100;

                  qsc(i,1)*k_matrix(i,1)*1e5;

                  k_matrix(i,1)*1e5;

                  Sealing_Capacity(i,1)*qsc(i,1)/10;

                  Sealing_Capacity(i,1)/10;

                  L(i,1)*qsc(i,1)/100;

                  L(i,1)/100;

                  fault_angle(i,1)*qsc(i,1)/10;

                  fault_angle(i,1)/10;

                  qsc(i,1)*(1-Sealing_Capacity(i,1)/100)*L(i,1)/100;

                  (1-Sealing_Capacity(i,1)/100)*L(i,1)/100;

                  qsc(i,1);

                  por_matrix(i,1)*10;
```

```
                por_frac(i,1)*100;
                frac_space(i,1)/10;
                lambda(i,1)*1e7;
                w(i,1)*100];

   end
   save inputs.mat inputs
   save targets.mat targets
 plot(log10(Time*24),C(:,1:10:500))
```

11. Example of Matlab Neural Network Toolbox

```
% Design hidden layer sizes by changing the numbers in the 'hiddenLayerSize'
HiddenLayerSize = [240 120 60 40];
% feedforwardnet
net = feedforwardnet(HiddenLayerSize);
% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};
% Create connection from inputs to subsequent layers
 net.inputConnect(1:length(HiddenLayerSize)+1) = 1;
% Designate transfer fuctions for layers, however, transfer functions can be assigned individually
 net.layers{1:length(HiddenLayerSize)}.transferFcn = 'tansig';
% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 10/100;
% Network derivative function can be modified here
% In order to change the derivative function of the ANN uncomment this line and derivative function in the quotation
mark
% net.derivFcn = 'fpderiv'; %  and change the derivative

% training algorithm
net.trainFcn = 'trainscg';
%% if I use different learning function uncomment following line
% net.inputWeights{1:length(HiddenLayerSize)}.learnFcn = 'learnwh';
%%
% Choose a Performance Function
% For a list of all performance functions type: help nonperformance
net.performFcn = 'msereg';  % Mean squared error msereg
net.performParam.ratio = 0.9;
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};
net.trainParam.goal = 1e-4; %accuracy within this range
net.trainParam.epochs = 1000000; % number of iteration sets
```

```
%% This parameter can be modified but with proper training function
% net.trainParam.searchFcn = 'srchbac'; %srchbac, srchbre, srchcha, srchgol, srchhyb
net.trainParam.show = 5;
net.trainParam.max_fail = 1000;
net.trainParam.lr=0.01;
% net.trainParam.mc = 0.9;
net.trainParam.min_grad=1e-30;
% net.initFcn = 'initlay';
net=init(net);
% Train the Network
[net,tr] = train(net,inputs,targets);
% Test the Network
outputs=net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs);
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};

trainPerformance = perform(net,trainTargets,outputs);
valPerformance = perform(net,valTargets,outputs);
testPerformance = perform(net,testTargets,outputs);
% Getting indices of randomly divided inputs and outputs
[Pn_train,Pn_val,Pn_test,trainInd,valInd,testInd] = divideind(inputs,tr.trainInd,tr.valInd,tr.testInd);
[Tn_train,Tn_val,Tn_test] = divideind(targets,tr.trainInd,tr.valInd,tr.testInd);
Tn_test_ann = sim(net,Pn_test);
A = abs(Tn_test-Tn_test_ann)./Tn_test*100;
inputweight=net.IW;
layerweight=net.LW;
biasvalue=net.b;

C1=[Tn_test(1,:);Tn_test_ann(1,:)]/100; % Permeability ratio
C2=[Tn_test(2,:);Tn_test_ann(2,:)]/100; % Fracture permeability in x direction
C3=[Tn_test(3,:);Tn_test_ann(3,:)]/100; % Fracture permeability in y direction
C4=[Tn_test(6,:);Tn_test_ann(6,:)]/1e5; % Matrix permeability
C5=[Tn_test(8,:);Tn_test_ann(8,:)]*10; % Sealing capacity
C6=[Tn_test(10,:);Tn_test_ann(10,:)]*100; % Distance to the fault
C7=[Tn_test(12,:);Tn_test_ann(12,:)]*10; % Fault angle
C8=[Tn_test(14,:);Tn_test_ann(14,:)]*10; % Communication index
C9=[Tn_test(16,:);Tn_test_ann(16,:)]/10; % Matrix porosity
C10=[Tn_test(17,:);Tn_test_ann(17,:)]/100; % Fracture porosity
C11=[Tn_test(18,:);Tn_test_ann(18,:)]*10; % Fracture spacing
C12=[Tn_test(19,:);Tn_test_ann(19,:)]/1e7; % Inter-porosity flow parameter
C13=[Tn_test(20,:);Tn_test_ann(20,:)]/100; % Storativity ratio
```