Co_2-H_2o Fugacity Modeling Using Neural Network

by

Mohd Syafiq Bin Mohd Hassan

Dissertation submitted in partial fulfillment of
the requirement for the
Bachelor of Engineering (Hons)
(Chemical Engineering)

DECEMBER 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Co_2-H_2o Fugacity Modeling Using Neural Network

by

Mohd Syafiq Bin Mohd Hassan

A project dissertation submitted to the
Chemical Engineering Programme
Universiti Teknology PETRONAS
in partial fulfillment of requirement for the
BACHELOR OF ENGINEERING (Hons)
(CHEMICAL ENGINEERING)

Approved by,

_____

(Ir. Dr. Abdul Halim Shah Maulud)

UNIVERSITY TEKNOLOGI PETRONAS

TRONOH, PERAK

DECEMBER 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(Mohd Syafiq Bin Mohd Hassan)

ABSTRACT

Duan and Sun (2003) have designed a theoretical model for carbon dioxide (CO_2) solubility in pure water. This model is valid for solutions from 273 to 573K and from 0 to 2000 bar, while in the other hand, all the parameters presented in the model can be directly calculated without any iteration, except fugacity coefficient of CO_2 (ϕ_{CO2}) which is a function of temperature (T) and pressure (P). In order to calculate the ϕ_{CO2}, 15 coefficients must be fitted into the equation. Since the P-T diagram of CO_2 is divided into 6 regions, different sets of these coefficients need to be applied for different regions. Hence, there is a need to design a single model to calculate ϕ_{CO2} for the whole regions of P-T diagram which will be done in this project.

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to God for giving me enough time, determination and strength in completing this project. Without assistance from Him, I will not be able to even perform this project outstandingly. Then, my deepest appreciation goes to my supervisor, Ir. Dr. Abdul Halim Shah Maulud for his guidance and assistance throughout the entire duration of Final Year Project I (FYP1) as well as Final Year Project II (FYP2). There are lots of assistance and guidance from him, enabling me to achieve objectives of my project.

Also, special thanks to my colleagues, who are willing to give any ideas and information about my project. Last but not least, I would like to thank my parents for their moral support and encouragement, a 'driving force' for me throughout my life. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Project Background

Carbon dioxide (CO_2) is one of the main undesired by-products in any process mostly in oil and gas industry. Most industries demand for clean and pure natural gas for various applications such as automobile fuel, generation of electricity and domestic use. Thus, this gas should be removed to an acceptable level as natural gas, for example, is highly contaminated by carbon dioxide. Besides, removal of high amount of CO_2 need to be done as worldwide is concerned about carbon emission to atmosphere. In industry such as ammonia production, large quantities of CO_2 are produced and separated, however, most of the gas vented to the open atmosphere instead of captured and stored.

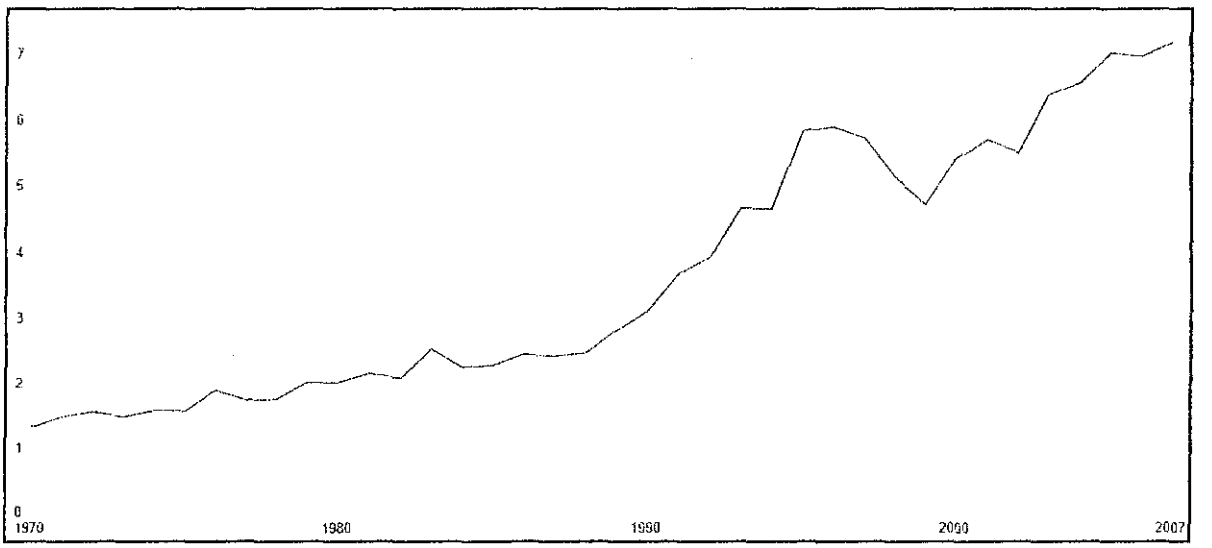


Figure 1.1: CO_2 emissions per capita in Malaysia. 7.18 metric tons per capita in 2007 (World Development Indicator, 2011)

Truly, it is undesired gas in most processes because of its ability to do damages on equipments in the plants. However, CO_2 cannot be easily disposed to the environment as it can cause global warming as well as harmful and hazardous to the surrounding creatures. Thus, there is a need to have a proper CO_2 removal system to tackle this situation. Nowadays, removal of CO_2 is commonly done by using adsorption, absorption, membrane separation and cryogenic. The option of appropriate technology to be used depends on the characteristic of flue gas stream. For instance, in a coal IGCG (integrated gasification combined cycle) process, the CO_2 concentration would be about 35%-40% at pressure of 20 bar or more. In that case, physical solvents such as Selexol could be used for pre-combustion capture of CO_2 with the advantage that it can be released mainly by depressurization. However, this research will only focus on removal of CO_2 by absorption.

By definition, absorption process is a mass transfer operation in which one or more species (solute) is removed from a gaseous stream by dissolution in a liquid (solvent) (Binay. K. D., 2007). Hence, solubility of gas; specifically solubility of CO_2 in solvents is a very important parameter in order to measure the absorption of CO_2.

1.2. Problem Statement

Based on thermodynamic model of CO_2 solubility designed by Duan and Sun (2003), it has precisely predicted the CO_2 solubility in water. Nevertheless, this model is still dependent on fugacity coefficients of CO_2 which is calculated iteratively. Besides, there are 15 parameter coefficients need to be fitted into the model. Also, different P-T regions require different sets of parameter coefficients. Thus, there is a need of a single model to calculate CO_2 fugacity for the whole regions. Since the relationship between fugacity, pressure and temperature are highly non-linear, it is expected that Neural Network (NN) will be able to provide the solution.

1.3. Objectives

- To develop a neural network (NN) modeling for CO_2-H_2O fugacity coefficients.
- To generate values of weights and biases from designed NN model.
- To design a mathematical model to calculate fugacity coefficients of CO_2, ϕ_{CO2} with similar accuracy with available model.

1.4. Scope of Study

- To do literature review on CO_2-H_2O fugacity as well as neural network (NN) modeling.
- To collect appropriate data from literature review.
- To reproduce the original parameter coefficients.
- To develop a single model for CO_2-H_2O system in order to calculate ϕ_{CO2} by achieving the same accuracy with available model.

CHAPTER 2

LITERATURE REVIEW

2.1. Fugacity

2.1.1. Definition

The specific Gibbs function for a simple compressible substance is:

$$dg = v\,dP - s\,dT \tag{1}$$

As in a pure substance the specific Gibbs function equals the chemical potential, we can inscribe for a isothermal process:

$$d\mu_T = v\,dP \tag{2}$$

And by replacing the ideal gas EOS we obtain:

$$d\mu_{T,ideal} = \frac{RT\,dP}{P} = RT\,d\ln P \tag{3}$$

From Eq. (3) we can calculate the chemical potential of a pure material that behaves as an ideal gas. For real gas, we can use an EOS and compute the chemical potential by integration. However, this approach is not followed. Instead, a new thermodynamic property is defined such that the form of Eq. (3) is still valid for a real gas. This new function is the *fugacity f*, defined as:

$$d\mu_{T,real} = RT\,d\ln f \tag{4}$$

Besides, as the real gas and the ideal gas behave the same at very low pressure, it is apparent that:

$$\lim_{P \to 0} \frac{f}{P} = 1 \tag{5}$$

Thus, with the definition of Eq. (4) and with the reference value of f=0 at 0 pressure the fugacity is absolutely well-defined.

2.1.2. Evaluation of Fugacity

Using the definition of the isothermal chemical potential, Eq. (2), and the fugacity, Eq. (4) we can write:

$$d \ln f = \frac{v\,dP}{RT} \tag{6}$$

Eq. (6) in conjunction with an EOS (explicit in the specific volume) can be used to calculate the fugacity. Integrating between two pressures we get:

$$RT \ln \frac{f}{f_0} = \int_{P_0}^{P} v\,dP \tag{7}$$

If the EOS is explicit in pressure, we can use the equation:

$$d(vP) = v\,dP + P\,dv \tag{8}$$

Replacing Eq. (8) in Eq. (6) and integrating we get:

$$RT \ln \frac{f}{f_0} = Pv - P_0 v_0 - \int_{v_0}^{v} P\,dv \tag{9}$$

Using the Redlich-Kwong EOS, $p = \frac{RT}{v-b} - \frac{a}{\sqrt{T}\,v\,(v+b)}$ we can use Eq. (9) to acquire:

$$RT\ln\frac{f}{f_0}=Pv-P_0v_0-RT\ln\frac{v-b}{v_0-b}-\frac{a}{b\sqrt{T}}\ln\frac{(v+b)v_0}{(v_0+b)v} \tag{10}$$

Taking $P_0 \rightarrow 0$ the gas behaves as ideal, we can describe:

$$\ln f=\frac{Pv}{RT}-1+\ln\frac{RT}{v-b}-\frac{a}{bRT^{3/2}}\ln\frac{(v+b)}{v} \tag{11}$$

and substituting by the definition of the pressure we get:

$$\ln f=\frac{b}{v-b}+\ln\frac{RT}{v-b}-\frac{a}{RT^{3/2}}\left[\frac{1}{v+b}+\frac{1}{b}\ln\frac{(v-b)}{v}\right] \tag{12}$$

A similar procedure applying Van der Waals EOS leads to:

$$\ln f=\frac{b}{v-b}+\ln\frac{RT}{v-b}-\frac{2a}{RTv} \tag{13}$$

Assessment of the fugacity from tables or EOS's is usually done using the *fugacity coefficient* φ, defined as:

$$\phi\equiv\frac{f}{p} \tag{14}$$

that can be differentiated to obtain:

$$d\ln\phi=d\ln f-d\ln p \tag{15}$$

and combining Eq. (15) with Eq. (6) we get:

$$d\ln\phi=\left(\frac{v}{RT}-\frac{1}{P}\right)dP \tag{16}$$

Integration of Eq. (16) at constant temperature from 0 pressure (φ = 1) to a state pressure P gives:

$$\ln\frac{f}{P}=\int_0^P\left(\frac{v}{RT}-\frac{1}{P}\right)dP \tag{17}$$

which recounts *PvT* data with the fugacity. Eq. (17) can be integrated numerically from data or an EOS of state can be used to analyze the integral analytically. If we substitute the definition of the Z factor in Eq. (17) we obtain:

$$\ln\frac{f}{P}=\int_0^P\left(\frac{v}{RT}-\frac{1}{P}\right)dP=\int_0^P\left(\frac{Z-1}{P}\right)dP \tag{18}$$

or, in terms of reduced properties,

$$\ln\frac{f}{P}=\int_0^{P_r}\left(\frac{Z-1}{P_r}\right)dP_r \tag{19}$$

Recall that the integral in Eq. (19) has been already solved if we evaluated residual or departure functions for the entropy, Eq. (47) in notes "Thermodynamic Properties". Analysis of the *Z* chart illustrates that for *Pr* smaller than 0.4 the *Z-Pr* curves are straight lines with slope $(Z-1)$ / *Pr*. Then the integral in Eq. (19) can be readily evaluated:

$$\ln\frac{f}{P}=\int_0^{P_r}\left(\frac{Z-1}{P_r}\right)dP_r=\left(\frac{Z-1}{P_r}\right)\int_0^{P_r}dP_r=Z-1 \tag{20}$$

that can be shown as:

$$\frac{f}{P}=e^{Z-1}\cong 1+(Z-1)+\frac{(Z-1)^2}{2!}+\frac{(Z-1)^3}{3!}+\cdots \tag{21}$$

for $Z>0.9$ the series on Eq. (21) can be approximated well as:

$$\frac{f}{P}=Z \tag{22}$$

so at low pressures and for *Z* close to 1 we can use Eq. (22) with reasonable accuracy. To acquire *f* from tabular data we integrate Eq. (4) between a reference state and the state of interest along an isotherm, to get:

$$\mu_{T,P} - \mu_{T,P_{ref}} = RT \ln \frac{f}{f_{ref}} \tag{23}$$

As the chemical potential is the specific Gibbs function for a pure substance, we can describe:

$$\ln \frac{f}{f_{ref}} = \frac{g - g_{ref}}{RT} \tag{24}$$

We can use Eq. (24) to calculate the fugacity of a real gas if we use a reference state with low enough pressure such that the reference fugacity is the pressure (ideal gas) and recalling that $g = h - Ts$. Then Eq. (24) becomes:

$$\ln \frac{f}{P_{ref}} = \frac{1}{R}\left[\frac{h - h_{ref}}{T} - (s - s_{ref})\right] \tag{25}$$

Eq. (25) is used in the following way: a reference state is preferred at the lowest pressure available at the state temperature. If the pressure is not low enough to be in the ideal gas region, extrapolation of the properties to a lower pressure region might be a necessity. Thus, the evaluation of h and s from tables permit the calculation of f. f can be also estimated from a three-parameter principle of corresponding states using Pitzer acentric factor ω. Due to the close relation between Z and f, the fugacity coefficient is tabulated as:

$$\log_{10} \frac{f}{P} = \left(\log_{10} \frac{f}{P}\right)^0 + \omega \left(\log_{10} \frac{f}{P}\right)^1 \tag{26}$$

Eq. (23) for a real gas can be simplified for the case of an ideal gas:

$$\left(\mu_{T,P} - \mu_{T,P_{ref}}\right)_{ig} = RT \ln \frac{P}{P_{ref}} \tag{27}$$

where the reference state is generally chosen at unit pressure, normally 1 atm. This allows us to write the following two relations:

$$\left(\mu_{T,P}\right)_{ig} = \left(g_{T,P}\right)_{ig} = g_T^0 + RT\ln P \tag{28}$$

$$\mu_{T,P} = g_T^0 + RT\ln f \tag{29}$$

Subtracting Eq. (28) from Eq. (29) we obtain:

$$\mu_{T,P} - \left(\mu_{T,P}\right)_{ig} = RT\ln\frac{f}{P} \tag{30}$$

2.2. CO_2-H_2O System

Accurate prediction of CO_2 solubility over a wide range of pressure, temperature as well as types of solution is very important nowadays as it is used to studies of geological CO_2 sequestration, carbonate precipitation, global carbon cycle, as well as fluid inclusions (Kaszuba et al., 2003; Spycher et al., 2003; Xu et al., 2004; Cipolli et al., 2004; Wolf et al., 2004). Upon the introduction of a thermodynamic model of CO_2 solubility in aqueous NaCl solution (Duan and Sun, 2003), it has later been extended to other aqueous electrolyte solutions such as K^+, Mg^{2+}, Ca^{2+}, and SO_4^{2-} (Duan et al., 2005).

In addition, the relatively high precise and wide temperature–pressure–composition range (from 273 to 533 K, 0 to 2000 bar, and 0 to 4.5 m NaCl) of that model permits its applications in various numerical simulation programs such as MATLAB. According to Duan and Sun (2003; 2005), the solubility of CO_2 can be calculated by:

$$\ln m_{CO_2} = \ln y_{CO_2}\phi_{CO_2}P - \frac{\mu_{CO_2}^{1(0)}}{RT} - 2\lambda_{CO_2\text{-}Na}(m_{Na} + m_K + 2m_{Ca} + 2m_{Mg}) - \zeta_{CO_2\text{-}Na\text{-}Cl}m_{Cl}(m_{Na} - m_K + m_{Ca} + m_{Mg}) + 0.07m_{SO_4} \quad (31)$$

where T is absolute temperature in Kelvin, P represents the total pressure of the system in bar, R is universal gas constant, m means the molality of components dissolved in water, $y_{CO2}$is the mole fraction of CO_2 in vapor phase, ϕ_{CO2} is the fugacity coefficient of CO_2, $\mu_{CO2}^{1(0)}$ is the standard chemical potential of CO_2 in liquid phase, $\lambda_{CO2\text{-}Na}$ is the interaction parameter between CO_2 and Na^+, $\zeta_{CO2\text{-}Na\text{-}Cl}$ is the interaction parameter between CO2 and Na^+, Cl^-.

Based on Eq. (1), all parameters appeared can be directly calculated without any need of iterations, with the exception of value of ϕ_{CO2}. The ϕ_{CO2} is calculated from the fifth-order virial equation of state (EOS) designed by Duan et al. (1992). It iteratively calculates the value of ϕ_{CO2}, which leads to time-consuming procedure as well as computationally expensive especially for

numerical simulations of large-scale geological sequestration processes that require calculation of CO_2 solubility at each grid and at each time step (Spycher et al., 2003). Below is the EOS to find the value of ϕ_{CO2} (Eq. 32):

$$\phi_{CO_2} = b_1 + \left[b_2 + b_3 T + \frac{b_4}{T} + \frac{b_5}{(T-150)}\right] P + \left[b_6 + b_7 T + \frac{b_8}{T}\right] P^2 + \left[b_9 + b_{10} T + \frac{b_{11}}{T}\right] lnP + \frac{[b_{12} + b_{13} T]}{P} + \frac{b_{14}}{T} + b_{15} T^2 \quad (32)$$

where T and P are in Kelvin and bar respectively. It becomes more complex as the P-T diagram of CO_2 is divided into six (6) sections, as such there 6 different sets of parameters to be fitted in. In addition, there are 15 parameter coefficients needed to be fitted in the EOS in order to find any ϕ_{CO2} value. Below are shown the table of parameter coefficients as well as P-T diagram for CO_2.

	T-P range					
	1	2	3	4	5	6
b_1	1.0	$-7.1734882 \cdot 10^{-1}$	$-6.5129019 \cdot 10^{-2}$	5.0383896	-16.063152	$-1.5693490 \cdot 10^{-1}$
b_2	$4.7586835 \cdot 10^{-3}$	$1.5985379 \cdot 10^{-4}$	$-2.1429977 \cdot 10^{-4}$	$-4.4257744 \cdot 10^{-3}$	$-2.7057990 \cdot 10^{-3}$	$4.4621407 \cdot 10^{-4}$
b_3	$-3.3569963 \cdot 10^{-6}$	$-4.9286471 \cdot 10^{-7}$	$-1.1444930 \cdot 10^{-6}$			$-9.1080591 \cdot 10^{-7}$
b_4	0.0			1.957233	$1.4119239 \cdot 10^{-1}$	
b_5	-1.3179396					
b_6	$-3.8389101 \cdot 10^{-6}$	$-2.7855285 \cdot 10^{-7}$	$-1.1558081 \cdot 10^{-7}$	$2.4223436 \cdot 10^{-6}$	$8.1132965 \cdot 10^{-7}$	$1.0647399 \cdot 10^{-7}$
b_7		$1.1877015 \cdot 10^{-9}$	$1.1952370 \cdot 10^{-9}$			$2.4273357 \cdot 10^{-10}$
b_8	$2.2815104 \cdot 10^{-3}$			$-9.3796135 \cdot 10^{-4}$	$-1.1453082 \cdot 10^{-4}$	
b_9				-1.5026030	2.3895671	$3.5874255 \cdot 10^{-1}$
b_{10}				$3.0272240 \cdot 10^{-3}$	$5.0527457 \cdot 10^{-4}$	$6.3319710 \cdot 10^{-5}$
b_{11}				-31.377342	-17.763460	-249.89661
b_{12}		-96.539512	-221.34306	-12.847063	985.92232	
b_{13}		$4.4774938 \cdot 10^{-1}$				
b_{14}		101.81078	71.820393			888.76800
b_{15}		$5.3783879 \cdot 10^{-6}$	$6.6089246 \cdot 10^{-6}$	$-1.5056648 \cdot 10^{-5}$	$-5.4965256 \cdot 10^{-7}$	$-6.6348003 \cdot 10^{-7}$

Table 2.1: Parameter coefficients for Eq. 2

1: 273 K<T<573 K, P<P_1 (when T<305 K, P_1 equals to the saturation pressure of CO_2; when 305 K<T<405 K, P_1=75+(T-305)-1.25; when T>405 K, P_1=200 bar.); 2: 273 K<T<340 K, P_1<P<1000 bar; 3: 273 K<T<340 K, P >1000 bar; 4: 340 K<T<435 K, P_1<P<1000 bar; 5: 340 K<T<435 K, P>1000 bar; and 6: T>435 K, P>P_1.

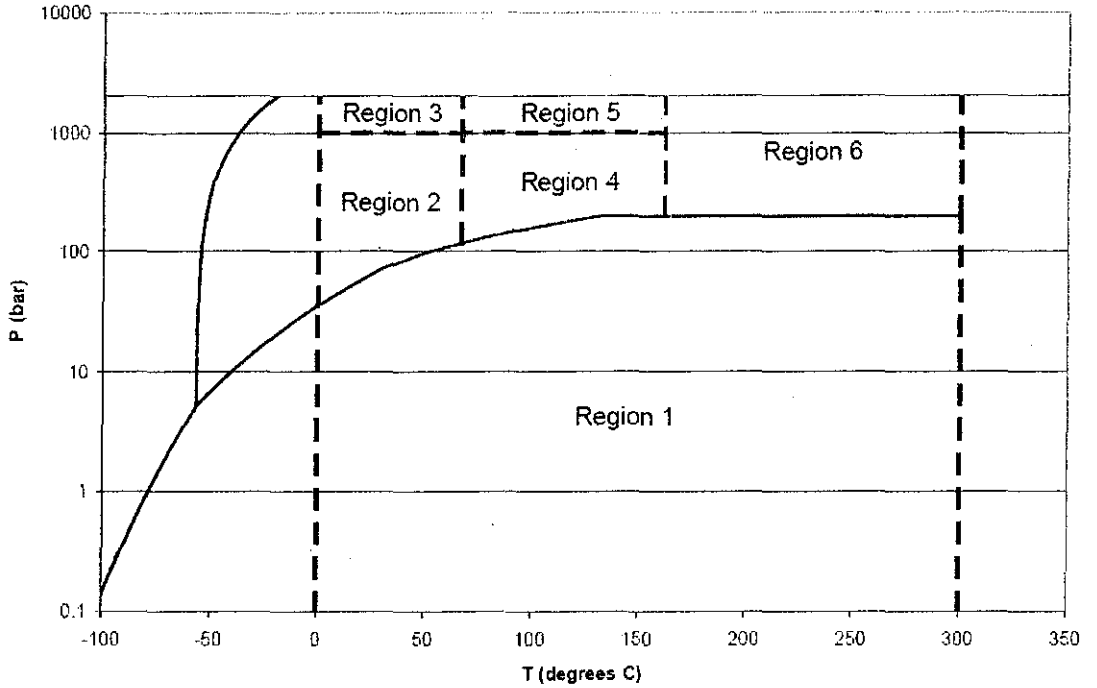


Fig 2.1: P-T diagram for CO_2

1: 273 K<T<573 K, P<P_1 (when T<305 K, P_1 equals to the saturation pressure of CO_2; when 305 K<T<405 K, P_1=75+(T-305)-1.25; when T>405 K, P_1=200 bar.); 2: 273 K<T<340 K, P_1<P<1000 bar; 3: 273 K<T<340 K, P >1000 bar; 4: 340 K<T<435 K, P_1<P<1000 bar; 5: 340 K<T<435 K, P>1000 bar; and 6: T>435 K, P>P_1.

Thus, this project intends to design a single model to calculate the value of ϕ_{CO2} for whole regions. However, since the relationship between fugacity, temperature, and pressure are expected highly non-linear, it is assumed that Neural Network (NN) is able to provide suitable solution for the project.

2.2.1. Comparison with Experimental Data

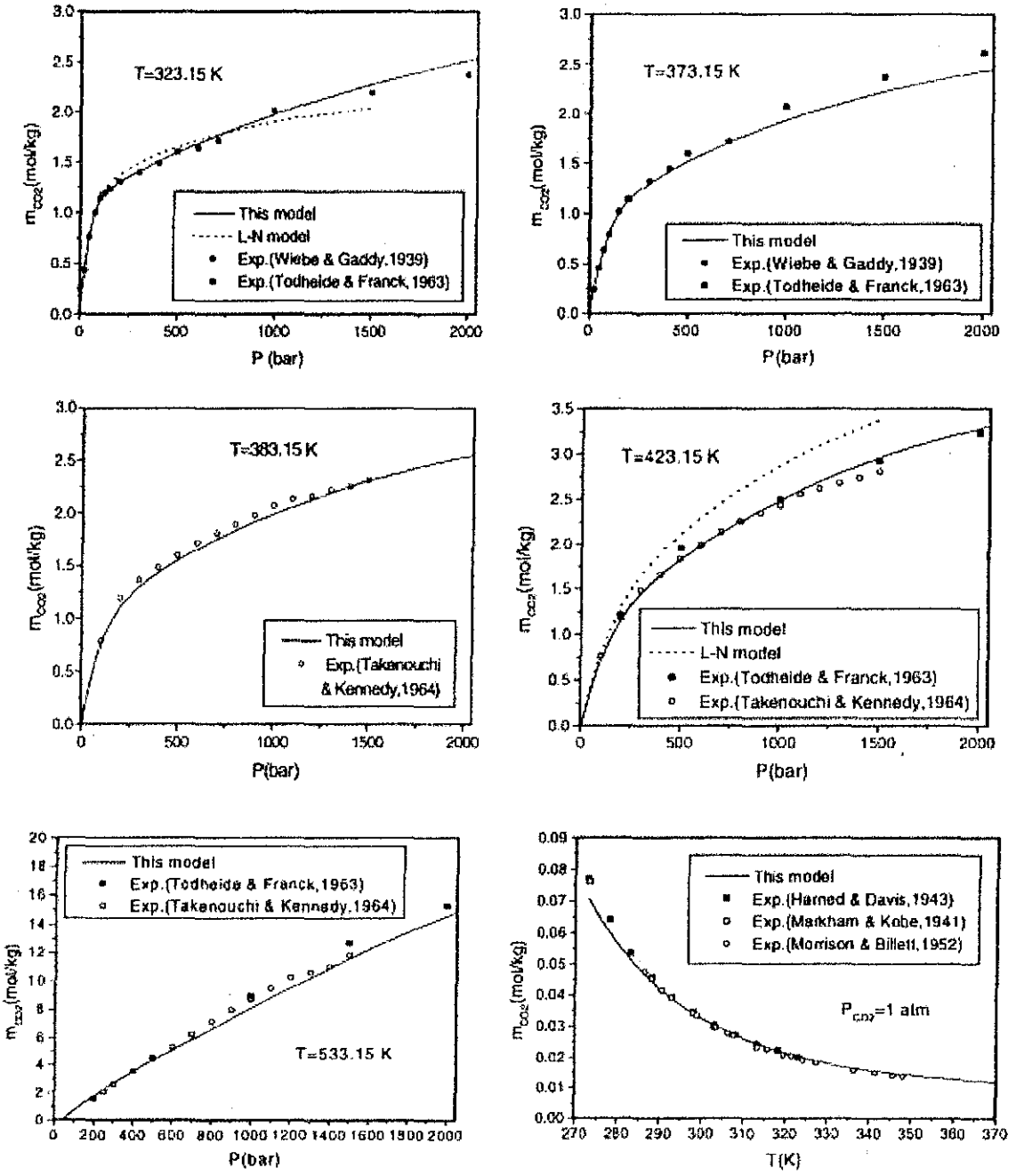


Fig. 2.2: CO2 solubility in pure water (the model of this study vs. experimental data and other models)

Based on the figures above we can conclude that comparison with experimental data demonstrates that this model gives results within or close to experimental uncertainty (about 7%)

in the temperature range from 273 to 533 K, for pressures from 0 to 2000 bar. Following the approach adopted by Duan et al. (1992a), this model is extended to predict CO2 solubility in more complex brines such as seawater with remarkable accuracy.

2.3. Neural Network

2.3.1. History Perspective

The field of artificial neural networks (they also go by the names of connectionist models, parallel distributed processing systems and neuromorphic systems) is very active nowadays. Although for many people artificial neural networks can be considered a new area of research, which developed in the late eighties, work in this field can be traced back to more than fifty years ago. *McCulloch* and *Pitts*, back in 1943, pointed out the possibility of applying Boolean algebra to nerve net behaviour. They essentially originated neurophysiological automata theory and developed a theory of nets. In 1949 *Donald Hebb*, in his book "*The organization of behavior*" postulated a plausible qualitative mechanism for learning at the cellular level in brains. An extension of his proposals is widely known nowadays as the *Hebbian learning rule*. In 1957 *Rosenblatt* developed the first neurocomputer, the *perceptron*. He proposed a learning rule for this first artificial neural network and proved that, given linearly separable classes, a perceptron would, in a finite number of training trials, develop a weight vector that would separate the classes (the famous *Perceptron convergence theorem)*. His results were summarized in a very interesting book, "*Principles of neurodynamics*".

About the same time *Bemard Widrow* modelled learning from the point of view of minimizing the mean-square error between the output of a different type of ANN processing element, the *ADALINE*, and the desired output vector over the set of patterns. This work has led to modern adaptive filters. Adalines and the *Widrow-Hoff learning rule* were applied to a large number of problems, probably the best known being the control of an inverted pendulum. Although people like Bemard Widrow approached this field from an analytical point of view, most of the research on this field was done from an experimental point of view. In the sixties many sensational promises were made which were not fulfilled. This discredited the research on

artificial neural networks. About the same time Minsky and Papert began promoting the field of artificial intelligence at the expense of neural networks research. The death sentence to ANN was given in a book written by these researchers, *Perceptrons,* where it was mathematically proved that these neural networks were not able to compute certain essential computer predicates like the EXCLUSIVE OR boolean function. Until the 80's, research on neural networks was almost null. Notable exceptions from this period are the works of Amari, Anderson, Fukushima, Grossberg and Kohonen.

Then in the middle 80's interest in artificial neural networks started to rise substantially, making NN one of the most active current areas of research. The work and charisma of *John Hopfield* has made a large contribution to the credibility of NN. With the publication of the *PDP* books the field exploded. Although this persuasive and popular work made a very important contribution to the success of NN, other reasons can be identified for this recent renewal of interest:

- One is the desire is to build a new breed of powerful computers, that can solve problems that are proving to be extremely difficult for current digital computers (algorithmically or rulebased inspired) and yet are easily done by humans in everyday life. Cognitive tasks like understanding spoken and written language, image processing, retrieving contextually appropriate information from memory, are all examples of such tasks.

- Another is the benefit that neuroscience can obtain from NN research. New artificial neural network architectures are constantly being developed and new concepts and theories being proposed to explain the operation of these architectures. Many of these developments can be used by neuroscientists as new paradigms for building functional concepts and models of elements of the brain.

- Also the advances of VLSI technology in recent years, turned the possibility of implementing NN in hardware into a reality. Analog, digital and hybrid electronic implementations are available today, with commercial optical or electrooptical implementations being expected in the future.

- The dramatic improvement in processing power observed in the last few years makes it possible to perform computationally intensive training tasks which would, with older technology, required an unaffordable time.

Research into ANNs has lead to several different architectures being proposed over the years. All of them try, to a greater or lesser extent, to exploit the available knowledge of the mechanisms of the human brain. Before describing the structure of artificial neural networks we should give a brief description of the structure of the archetypal neural network.

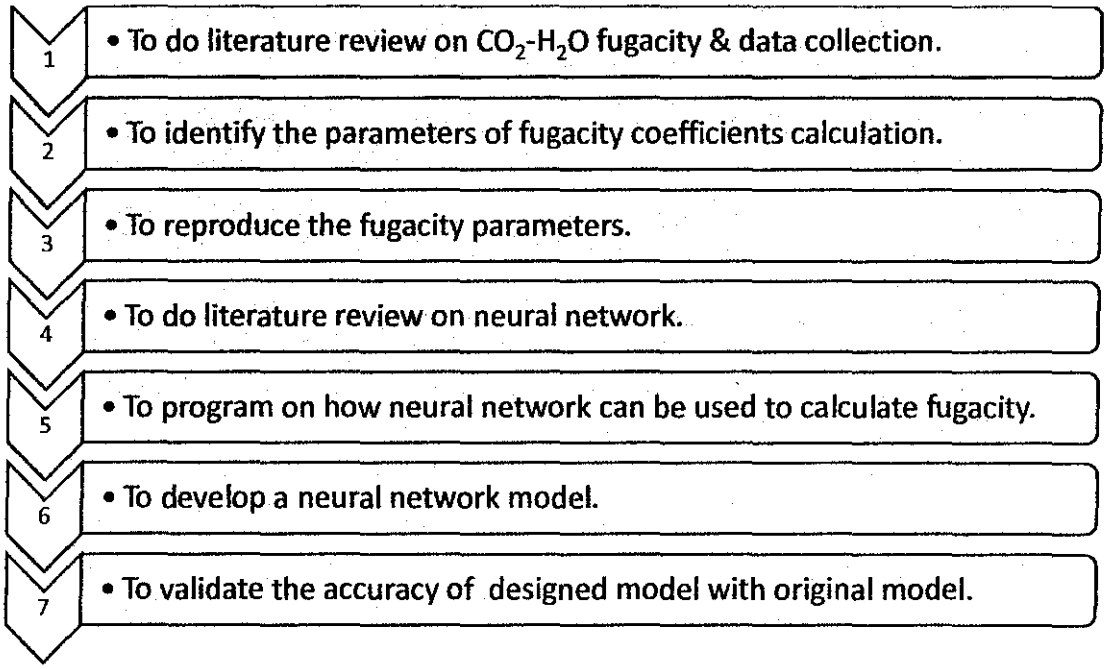
2.3.2. Project Perspective

In this project, Neural Network (NN) will be used to generate a mathematical model which is able to calculate ϕ_{CO2} for the whole regions of P-T diagram for CO_2. Thus, MATLAB is preferable computational software to be used for the project. The NN represents a complex configuration consists of many simple processors (or 'neurones'), arranged in layers (input, hidden and output layer). The proper transformation of information is possible as a result of correctly prepared matrix of weights, numbers attributed for the all interneuron's connections. In the most frequently used feed-forward network, each neurone simply sums up (properly amplified or weaken) signals from the all neurones of the previous layer and, after transformation received result by suitable activation function transmits it to all neurones of the next layer. In addition, the learning process is executed on the basis of input data sets and associated with output data sets. During the learning process (supervised learning) computer compares the values calculated with the expecting ones and adjusts step by step the weight's values to reach the best agreement between the six data sets. In the end of the computational activity, it is expected that a single model to calculate ϕ_{CO2} is able to be generated by MATLAB. Finally, the model value of ϕ_{CO2} is compared with the experimental value ϕ_{CO2} in order to validate the model.

CHAPTER 3

METHODOLOGY

3.1. Methodology



3.2. Gantt Chart

No	Activities	May		June				July				August					September				October					November				December				Jan
		23	30	6	13	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	24	28	5	12	19	26	2
1	Selection of Project Topic																																	
2	Preliminary Research Work																																	
3	Submission of Extended Proposal Defense																																	
4	Proposal Defense																																	
5	Start Project Work																																	
6	Submission of Interim Draft Report																																	
7	Submission of Interim Report																																	
8	Project Work Continues																																	
9	Submission of Progress Report																																	
10	Project Work Continues																																	
11	Pre-EDX																																	
12	Submission of Draft Report																																	
13	Submission of Dissertation (soft bound)																																	
14	Submission of Technical Paper																																	
15	Oral Presentation																																	
16	Submission of Project Dissertation (Hard Bound)																																	

Fig 3.1: Gantt Chart of Project

CHAPTER 4

RESULTS AND DISCUSSION

4.1. Results

After collecting data of temperature, pressure as well as actual CO_2 fugacity coefficients, $\phi_{CO2\ Actual}$, the neural network model has been designed based on try-and-error method. Besides, all data which used for simulation are randomly picked. However, the data are taken from the whole 6 regions in order to preserve the accuracy of the results at the end of the simulation. Recall that the formula used to get the outputs is equation (32) which is:

$$\phi_{CO_2} = b_1 + \left[b_2 + b_3T + \frac{b_4}{T} + \frac{b_5}{(T-150)}\right]P + \left[b_6 + b_7T + \frac{b_8}{T}\right]P^2 + \left[b_9 + b_{10}T + \frac{b_{11}}{T}\right]lnP + \frac{[b_{12} + b_{13}T]}{P} + \frac{b_{14}}{T} + b_{15}T^2$$

Thus, the actual outputs, $\phi_{CO2\ Actual}$ are fugacity coefficients of CO_2-H_2O system calculated based on formula above, while $\phi_{CO2\ Sim}$ are simulation outputs generated by neural network. Based on the table below, there is a "Region" column which represents the region of P-T diagram for CO_2 (Fig 1). Do take note that for region "1" of P-T diagram of CO_2, it has been divided into 3 parts due to conditions (P, T).

No	T (K)	P (bar)	$\phi_{CO2\ Actual}$	Region
1	274	1	0.9932	1 (Part 1)
2	276	1.5	0.9901	1 (Part 1)
3	278	2	0.9871	1 (Part 1)
4	280	2.5	0.9842	1 (Part 1)
5	282	3	0.9815	1 (Part 1)
6	284	3.5	0.9789	1 (Part 1)
7	286	4	0.9765	1 (Part 1)
8	288	4.5	0.9742	1 (Part 1)
9	290	5	0.9720	1 (Part 1)

10	292	5.5	0.9699	1 (Part 1)
11	294	6	0.9679	1 (Part 1)
12	296	6.5	0.9660	1 (Part 1)
13	298	7	0.9642	1 (Part 1)
14	300	7.5	0.9625	1 (Part 1)
15	302	8	0.9608	1 (Part 1)
16	545	143	0.9488	1 (Part 1)
17	547	143.5	0.9498	1 (Part 1)
18	549	144	0.9508	1 (Part 1)
19	551	144.5	0.9517	1 (Part 1)
20	553	145	0.9527	1 (Part 1)
21	555	145.5	0.9536	1 (Part 1)
22	557	146	0.9545	1 (Part 1)
23	559	146.5	0.9554	1 (Part 1)
24	561	147	0.9562	1 (Part 1)
25	563	147.5	0.9571	1 (Part 1)
26	565	148	0.9579	1 (Part 1)
27	567	148.5	0.9587	1 (Part 1)
28	569	149	0.9596	1 (Part 1)
29	571	149.5	0.9603	1 (Part 1)
30	572	150	0.9607	1 (Part 1)
31	306	76	0.6624	1 (Part 2)
32	308	76.5	0.6677	1 (Part 2)
33	310	77	0.6729	1 (Part 2)
34	312	77.5	0.6780	1 (Part 2)
35	314	78	0.6830	1 (Part 2)
36	316	78.5	0.6879	1 (Part 2)
37	318	79	0.6927	1 (Part 2)
38	320	79.5	0.6974	1 (Part 2)
39	322	80	0.7020	1 (Part 2)
40	324	80.5	0.7065	1 (Part 2)
41	326	81	0.7110	1 (Part 2)
42	328	81.5	0.7154	1 (Part 2)
43	330	82	0.7197	1 (Part 2)
44	332	82.5	0.7239	1 (Part 2)
45	396	170	0.7278	1 (Part 2)
46	398	170.5	0.7325	1 (Part 2)
47	400	171	0.7372	1 (Part 2)
48	402	171.5	0.7418	1 (Part 2)
49	404	172	0.7463	1 (Part 2)
50	406	150	0.7772	1 (Part 3)

51	408	150.5	0.7810	1 (Part 3)
52	410	151	0.7847	1 (Part 3)
53	412	151.5	0.7883	1 (Part 3)
54	414	152	0.7919	1 (Part 3)
55	416	152.5	0.7954	1 (Part 3)
56	418	153	0.7989	1 (Part 3)
57	420	153.5	0.8023	1 (Part 3)
58	422	154	0.8057	1 (Part 3)
59	424	154.5	0.8090	1 (Part 3)
60	564	197.5	0.9452	1 (Part 3)
61	566	198	0.9462	1 (Part 3)
62	568	198.5	0.9472	1 (Part 3)
63	570	199	0.9482	1 (Part 3)
64	572	199.5	0.9492	1 (Part 3)
65	274	900	0.1474	2
66	276	900.5	0.1527	2
67	278	901	0.1580	2
68	280	901.5	0.1635	2
69	282	902	0.1690	2
70	284	902.5	0.1747	2
71	286	903	0.1804	2
72	288	903.5	0.1862	2
73	290	904	0.1920	2
74	292	904.5	0.1980	2
75	331	997.5	0.3420	2
76	333	998	0.3497	2
77	335	998.5	0.3574	2
78	337	999	0.3652	2
79	339	999.5	0.3731	2
80	300	500	0.2071	2
81	302	500.5	0.2132	2
82	304	501	0.2193	2
83	306	501.5	0.2256	2
84	308	502	0.2319	2
85	274	200.5	0.1953	2
86	276	201	0.2025	2
87	278	201.5	0.2099	2
88	280	202	0.2173	2
89	282	202.5	0.2247	2
90	331	200.5	0.4411	2
91	333	201	0.4502	2

92	335	201.5	0.4592	2
93	337	202	0.4684	2
94	339	202.5	0.4775	2
95	331	1000.5	0.3416	3
96	333	1001	0.3493	3
97	335	1001.5	0.3570	3
98	337	1002	0.3648	3
99	339	1002.5	0.3727	3
100	274	1995	0.3725	3
101	276	1995.5	0.3830	3
102	278	1996	0.3936	3
103	280	1996.5	0.4043	3
104	282	1997	0.4150	3
105	284	1997.5	0.4259	3
106	286	1998	0.4368	3
107	288	1998.5	0.4479	3
108	290	1999	0.4590	3
109	292	1999.5	0.4702	3
110	330	1500	0.4641	3
111	332	1500.5	0.4736	3
112	334	1501	0.4833	3
113	336	1501.5	0.4930	3
114	338	1502	0.5028	3
115	341	900	0.3647	4
116	343	900.5	0.3719	4
117	345	901	0.3791	4
118	347	901.5	0.3863	4
119	349	902	0.3936	4
120	351	902.5	0.4009	4
121	353	903	0.4082	4
122	355	903.5	0.4156	4
123	357	904	0.4230	4
124	359	904.5	0.4304	4
125	426	997.5	0.7013	4
126	428	998	0.7088	4
127	430	998.5	0.7162	4
128	432	999	0.7236	4
129	434	999.5	0.7310	4
130	341	200.5	0.4929	4
131	343	201	0.5005	4
132	345	201.5	0.5081	4

133	347	202	0.5156	4
134	349	202.5	0.5230	4
135	426	200.5	0.7650	4
136	428	201	0.7689	4
137	430	201.5	0.7727	4
138	432	202	0.7765	4
139	434	202.5	0.7801	4
140	416	1995	1.1779	5
141	418	1995.5	1.1886	5
142	420	1996	1.1993	5
143	422	1996.5	1.2100	5
144	424	1997	1.2206	5
145	426	1997.5	1.2312	5
146	428	1998	1.2417	5
147	430	1998.5	1.2523	5
148	432	1999	1.2627	5
149	434	1999.5	1.2732	5
150	341	1000.5	0.3796	5
151	343	1001	0.3875	5
152	345	1001.5	0.3955	5
153	347	1002	0.4034	5
154	349	1002.5	0.4113	5
155	341	1500	0.5169	5
156	343	1500.5	0.5267	5
157	345	1501	0.5365	5
158	347	1501.5	0.5463	5
159	349	1502	0.5560	5
160	523	1995	2.3264	6
161	525	1995.5	2.3351	6
162	527	1996	2.3438	6
163	529	1996.5	2.3524	6
164	531	1997	2.3609	6
165	533	1997.5	2.3694	6
166	535	1998	2.3778	6
167	537	1998.5	2.3862	6
168	539	1999	2.3945	6
169	541	1999.5	2.4027	6
170	562	1997	2.4764	6
171	564	1997.5	2.4840	6
172	566	1998	2.4916	6
173	568	1998.5	2.4992	6

174	570	1999	2.5067	6
175	572	1999.5	2.5142	6
176	436	1800	1.6953	6
177	438	1800.5	1.7068	6
178	440	1801	1.7182	6
179	442	1801.5	1.7295	6
180	444	1802	1.7407	6
181	480	1900	2.0266	6
182	482	1900.5	2.0364	6
183	484	1901	2.0463	6
184	486	1901.5	2.0560	6
185	488	1902	2.0657	6
186	436	1600	1.5005	6
187	438	1600.5	1.5113	6
188	440	1601	1.5221	6
189	442	1601.5	1.5328	6
190	444	1602	1.5434	6
191	480	1700	1.8098	6
192	482	1700.5	1.8192	6
193	484	1701	1.8284	6
194	486	1701.5	1.8376	6
195	488	1702	1.8467	6
196	564	950	1.4150	6
197	566	950.5	1.4199	6
198	568	951	1.4249	6
199	570	951.5	1.4298	6
200	572	952	1.4347	6
201	436	200.5	0.8464	6
202	438	201	0.8501	6
203	440	201.5	0.8539	6
204	442	202	0.8576	6
205	444	202.5	0.8613	6
206	564	200.5	1.0276	6
207	566	201	1.0295	6
208	568	201.5	1.0315	6
209	570	202	1.0334	6
210	572	202.5	1.0353	6

Table 4.1: Data collection consists of temperature, pressure & fugacity coefficient

As mention before, the neural network is basically built based on try-and-error approach, in term of the number of neurons, layers as well as type of transfer functions. Below are shown several results of this project, which will be discussed in 'Discussion' section:

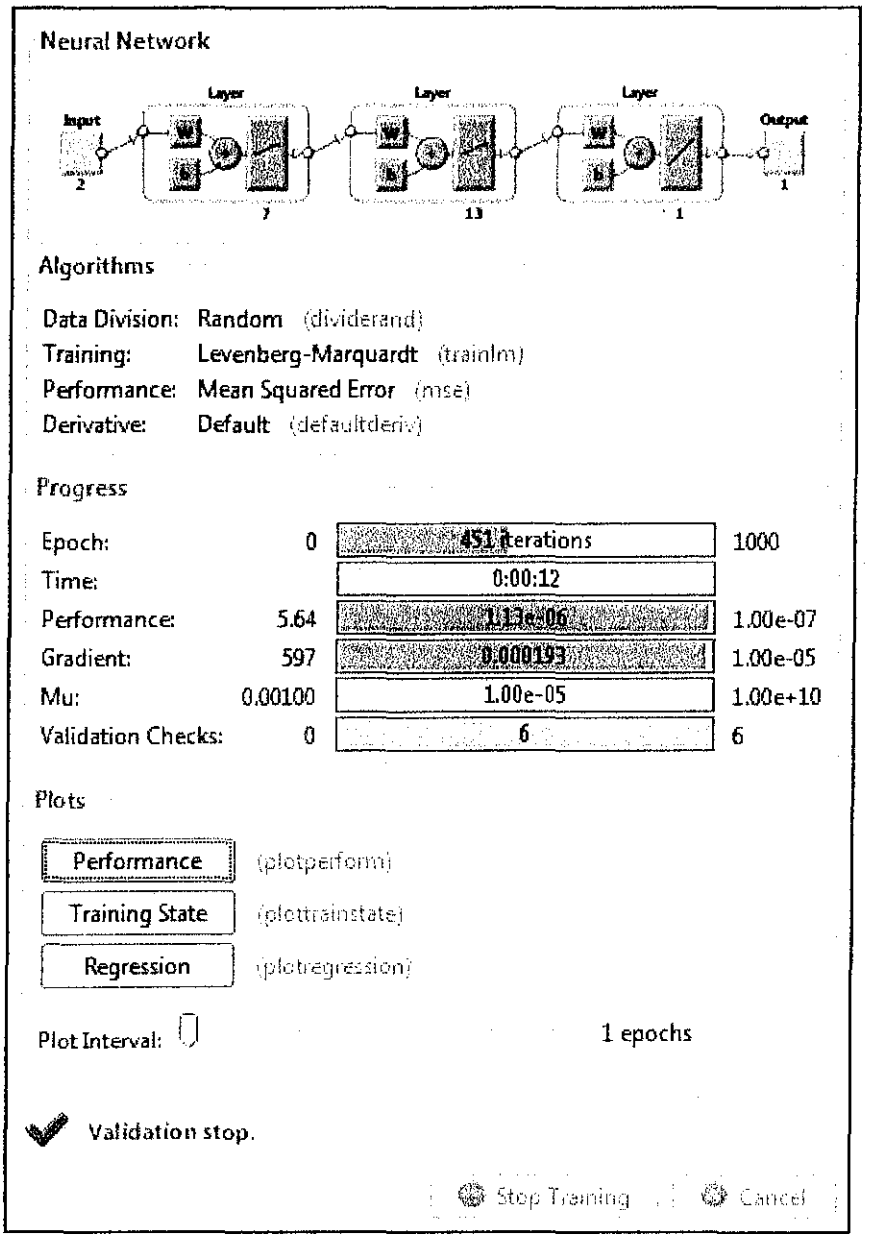


Fig 4.1: MATLAB neural network

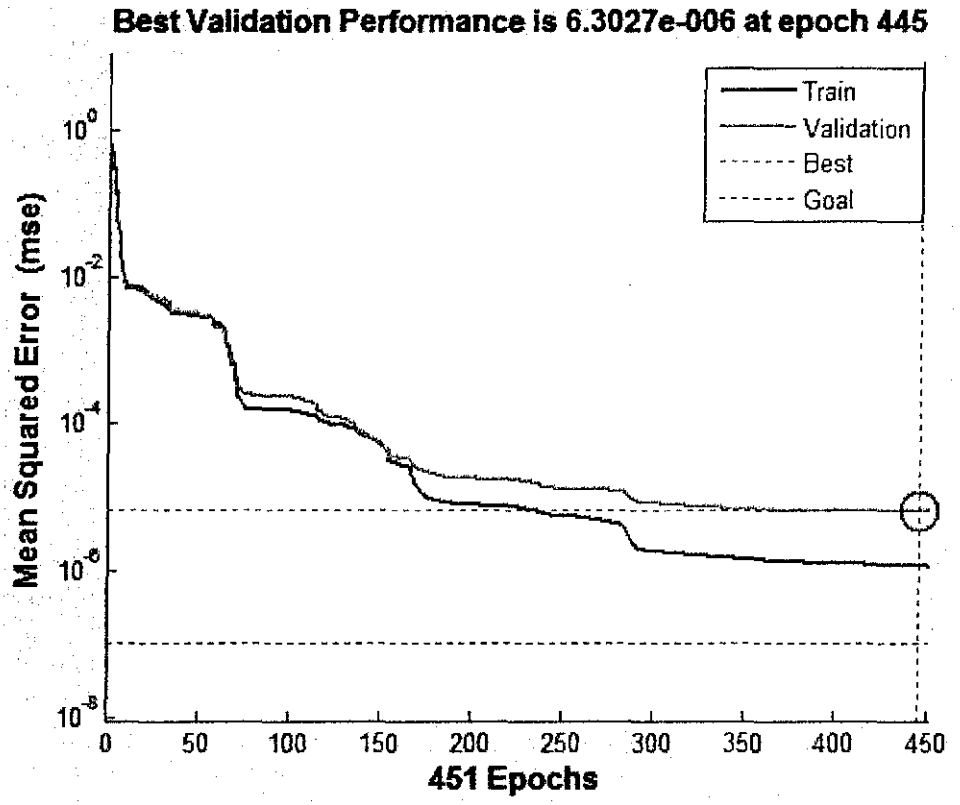


Fig 4.2: Performance of neural network

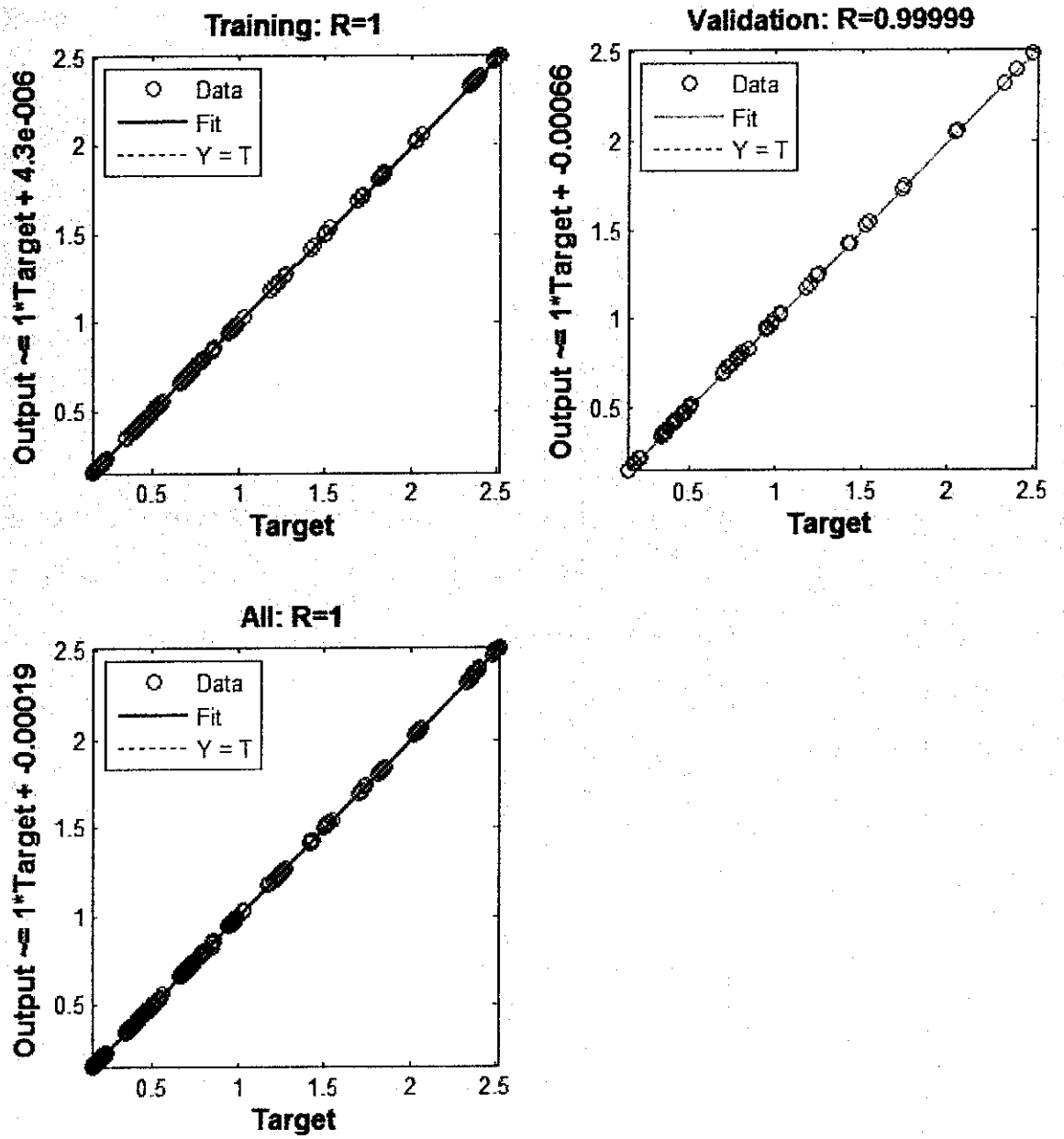


Fig 4.3: Regression plot of neural network

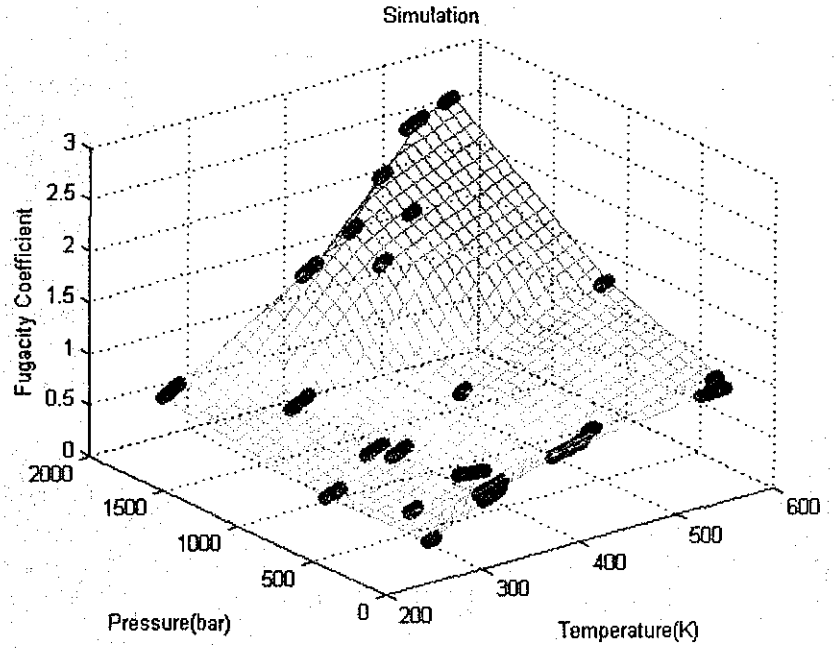
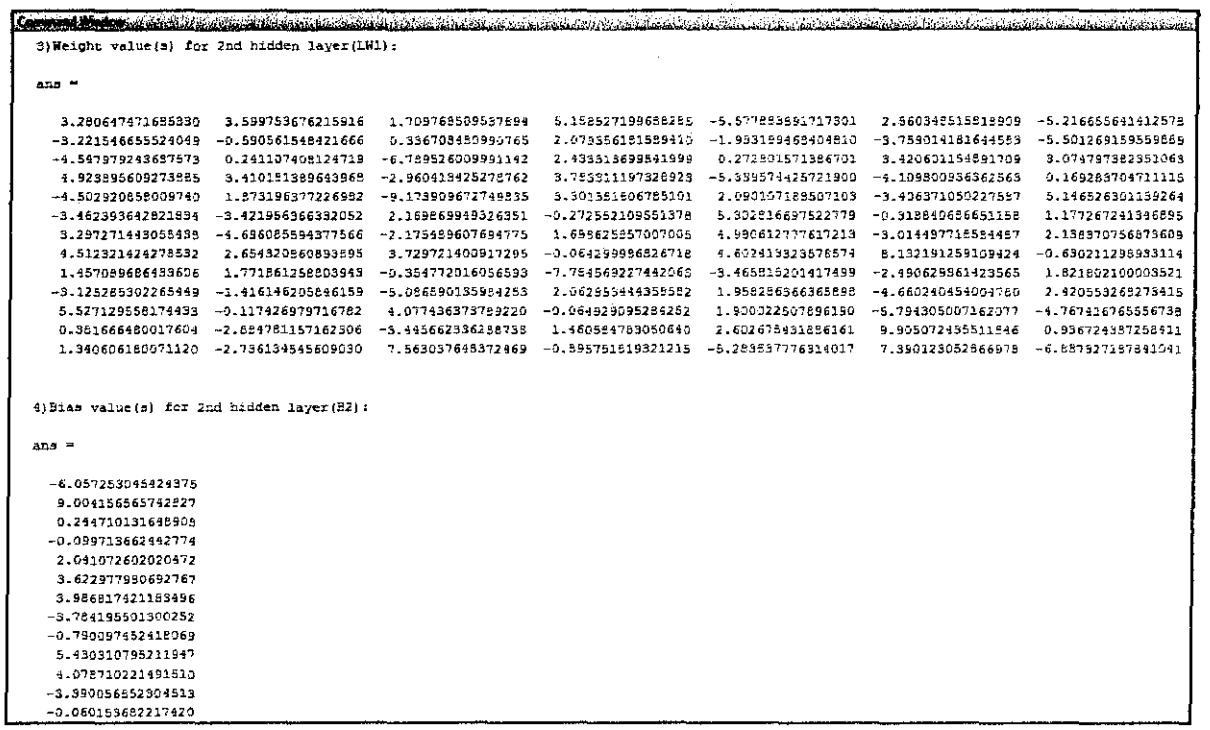


Fig 4.4: 3D plot of P-T- $\phi_{CO2\ Sim}$

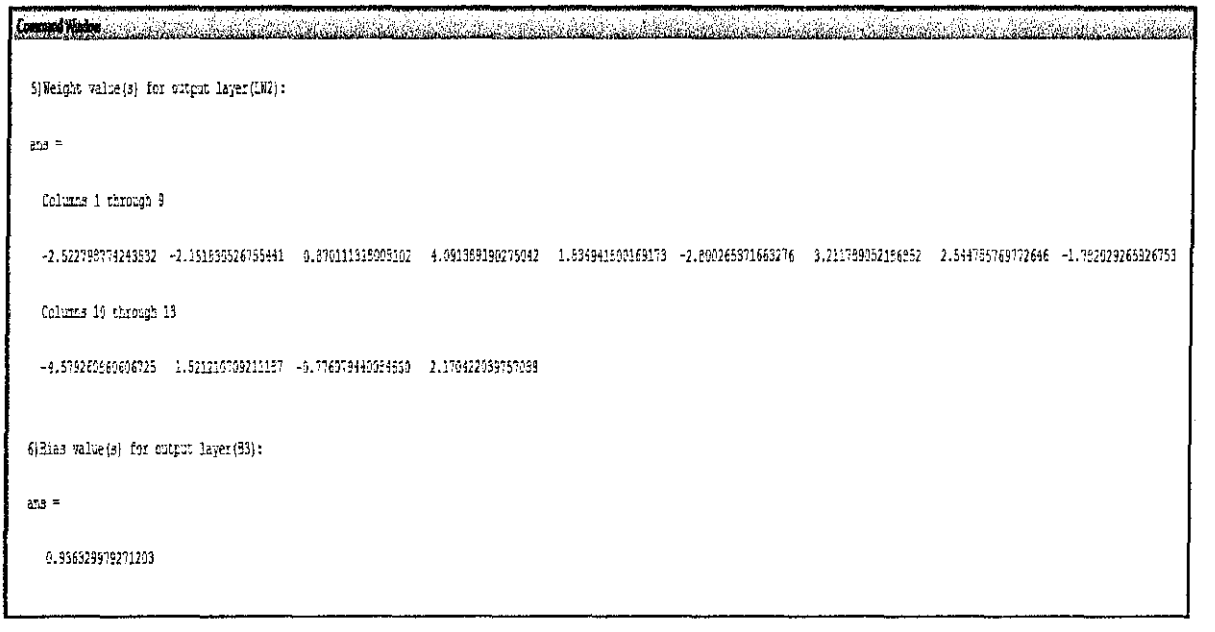
```
Command Window
Warning: NEWFF used in an obsolete way.
> In obs_use at 18
  In newff>create_network at 127
  In newff at 102
  In svafig6 at 26
        See help for NEWFF to update calls to the new argument list.

1)Weight value(s) for hidden layer (IW):

ans =

   0.024314886198726   0.009201900343075
   0.027122406187101  -0.028571475158764
   0.103436929228812  -0.136804741773623
   0.030422688264001  -0.003984725895334
  -0.060150865352321   0.088549019057763
   0.018744145443540   0.002423235438437
  -0.400073802631029   0.091304690342938


2)Bias value(s) for 1st hidden layer(B1):

ans =

 -29.671673019489301
 -27.659270012547863
 -17.212749648478820
  -7.716523562434156
  16.349496547615362
 -10.585679290653822
  -8.219902175068270
```

Fig 4.5: Weights & biases for 1st hidden layer



```
3)Weight value(s) for 2nd hidden layer(LW1):

ans =

   3.280647471685330   3.599753676215916   1.709768509537894   5.158527199658285  -5.577858991717301   2.560348515818909  -5.216655641412578
  -3.221546655524049  -0.590561548421666   5.336708480990765   2.073556181589410  -1.983189468404810  -3.758014181644559  -5.501269159559869
  -4.547979243687573   0.241107408124719  -6.789526009991142   2.433513699841998   0.272801571386701   3.420631154891709   3.074797382351063
   4.923895609273885   3.410151389643968  -2.960419425278762   3.755911197328923  -5.339574425721900  -4.109800956362565   0.169283704711115
  -4.502920858009740   1.873196377226982  -9.173909677749835   5.301581606785101   2.090157189507103  -3.426371050227557   5.146526301159264
  -3.462393642821894  -3.421956366332052   2.169869949326351  -0.272552109551378   5.302816697522779  -0.318840636651158   1.177267241346895
   3.297271443055438  -4.696085594377566  -2.175489607694775   1.695625857007005   4.990612737617219  -3.014497718584457   2.138370756873609
   4.512321424278532   2.654320860893895   3.729721400917295  -0.064299986826718   4.602419323578574   8.132191259109424  -0.630211298933114
   1.457089686483606   1.771861258803949  -0.354772016056593  -7.754569227442063  -3.465815201417499  -2.490629961423565   1.821802100003521
  -3.125285302265449  -1.416146205646159  -5.086890135984253   2.062955444355582   1.958286566365898  -4.660240454004760   2.420553269275415
   5.527129558174493  -0.117426979716782   4.077436373789220  -0.064929095286252   1.930022507896190  -5.794305007162077  -4.767416765556738
   0.381666480017604  -2.884781157162306  -3.445662336288758   1.460584783050640   2.602678431886161   9.905072455511946   0.956724387258411
   1.940606180071120  -2.736134545609030   7.563037648372469  -0.895751819321215  -5.289537776314017   7.350123052866978  -6.887527187841041


4)Bias value(s) for 2nd hidden layer(B2):

ans =

  -6.057253045424375
   9.004156565742227
   0.244710131648906
  -0.099713662442774
   2.041072602020472
   3.622977980692767
   3.986817421183496
  -3.784195501300252
  -0.790097452418069
   5.430310795211947
   4.078710221491510
  -3.990056852304513
  -0.060153682217420
```

Fig 4.6: Weights & biases for 2nd hidden layer



```
5)Weight value(s) for output layer(LW2):

ans =

  Columns 1 through 9

  -2.522798774243832  -2.151833526755441   0.870111319005102   4.091389190275042   1.534941500169173  -2.800265971663276   3.211789052156852   2.544755769772646  -1.752029265326753

  Columns 10 through 13

  -4.579262660606725   1.521210709211157  -9.776379440094659   2.170422039757099


6)Bias value(s) for output layer(B3):

ans =

   0.936329979271203
```

Fig 4.7: Weights & biases for output neuron

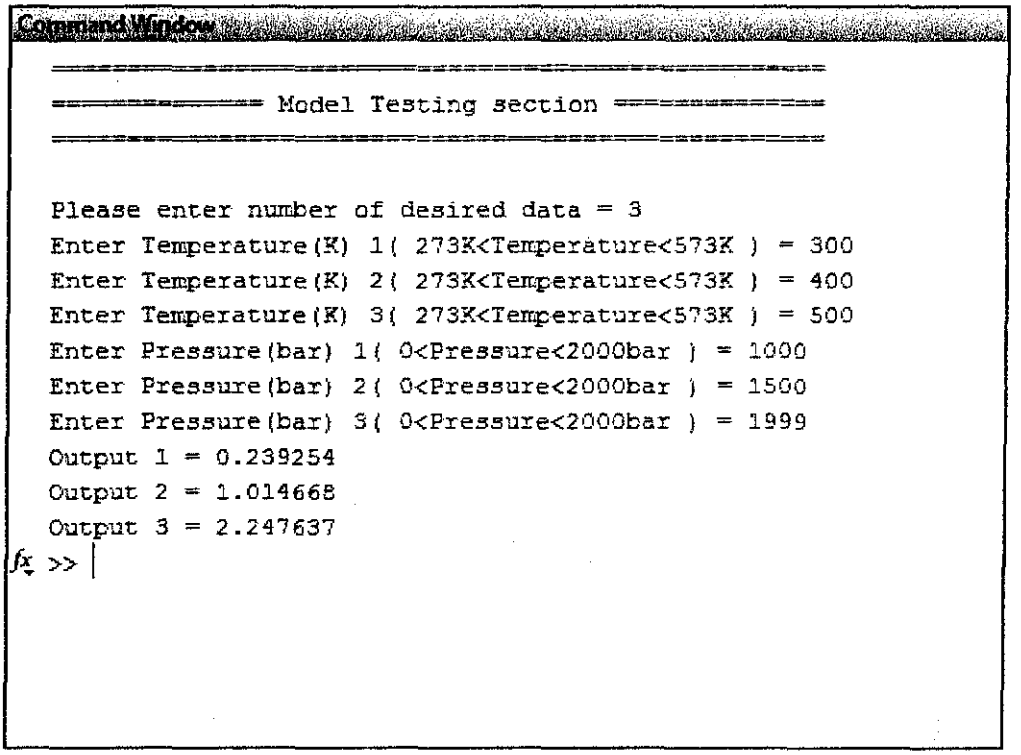


Fig 4.8: Model testing

No	T (K)	P (bar)	$\phi_{CO2\ Actual}$	$\phi_{CO2\ Sim}$	Error (%)	Region
1	274	1	0.9932	0.9930	0.0197	1 (Part 1)
2	276	1.5	0.9901	0.9902	0.0123	1 (Part 1)
3	278	2	0.9871	0.9873	0.0226	1 (Part 1)
4	280	2.5	0.9842	0.9844	0.0183	1 (Part 1)
5	282	3	0.9815	0.9816	0.0063	1 (Part 1)
6	284	3.5	0.9789	0.9789	0.0076	1 (Part 1)
7	286	4	0.9765	0.9763	0.0188	1 (Part 1)
8	288	4.5	0.9742	0.9739	0.0243	1 (Part 1)
9	290	5	0.9720	0.9717	0.0228	1 (Part 1)
10	292	5.5	0.9699	0.9697	0.0151	1 (Part 1)
11	294	6	0.9679	0.9678	0.0037	1 (Part 1)
12	296	6.5	0.9660	0.9660	0.0078	1 (Part 1)
13	298	7	0.9642	0.9643	0.0145	1 (Part 1)
14	300	7.5	0.9625	0.9626	0.0112	1 (Part 1)
15	302	8	0.9608	0.9608	0.0074	1 (Part 1)
16	545	143	0.9488	0.9469	0.2083	1 (Part 1)
17	547	143.5	0.9498	0.9488	0.1045	1 (Part 1)
18	549	144	0.9508	0.9505	0.0337	1 (Part 1)

19	551	144.5	0.9517	0.9518	0.0093	1 (Part 1)
20	553	145	0.9527	0.9529	0.0297	1 (Part 1)
21	555	145.5	0.9536	0.9539	0.0331	1 (Part 1)
22	557	146	0.9545	0.9547	0.0247	1 (Part 1)
23	559	146.5	0.9554	0.9555	0.0099	1 (Part 1)
24	561	147	0.9562	0.9562	0.0066	1 (Part 1)
25	563	147.5	0.9571	0.9569	0.0204	1 (Part 1)
26	565	148	0.9579	0.9577	0.0278	1 (Part 1)
27	567	148.5	0.9587	0.9585	0.0255	1 (Part 1)
28	569	149	0.9596	0.9594	0.0114	1 (Part 1)
29	571	149.5	0.9603	0.9605	0.0163	1 (Part 1)
30	572	150	0.9607	0.9608	0.0103	1 (Part 1)
31	306	76	0.6624	0.6622	0.0383	1 (Part 2)
32	308	76.5	0.6677	0.6679	0.0256	1 (Part 2)
33	310	77	0.6729	0.6732	0.0510	1 (Part 2)
34	312	77.5	0.6780	0.6783	0.0499	1 (Part 2)
35	314	78	0.6830	0.6832	0.0320	1 (Part 2)
36	316	78.5	0.6879	0.6879	0.0054	1 (Part 2)
37	318	79	0.6927	0.6925	0.0230	1 (Part 2)
38	320	79.5	0.6974	0.6970	0.0476	1 (Part 2)
39	322	80	0.7020	0.7016	0.0636	1 (Part 2)
40	324	80.5	0.7065	0.7061	0.0671	1 (Part 2)
41	326	81	0.7110	0.7106	0.0548	1 (Part 2)
42	328	81.5	0.7154	0.7152	0.0241	1 (Part 2)
43	330	82	0.7197	0.7198	0.0274	1 (Part 2)
44	332	82.5	0.7239	0.7246	0.1016	1 (Part 2)
45	396	170	0.7278	0.7277	0.0036	1 (Part 2)
46	398	170.5	0.7325	0.7328	0.0401	1 (Part 2)
47	400	171	0.7372	0.7375	0.0492	1 (Part 2)
48	402	171.5	0.7418	0.7412	0.0764	1 (Part 2)
49	404	172	0.7463	0.7436	0.3544	1 (Part 2)
50	406	150	0.7772	0.7767	0.0685	1 (Part 3)
51	408	150.5	0.7810	0.7807	0.0275	1 (Part 3)
52	410	151	0.7847	0.7847	0.0003	1 (Part 3)
53	412	151.5	0.7883	0.7884	0.0154	1 (Part 3)
54	414	152	0.7919	0.7921	0.0213	1 (Part 3)
55	416	152.5	0.7954	0.7956	0.0189	1 (Part 3)
56	418	153	0.7989	0.7990	0.0093	1 (Part 3)
57	420	153.5	0.8023	0.8022	0.0069	1 (Part 3)
58	422	154	0.8057	0.8054	0.0291	1 (Part 3)
59	424	154.5	0.8090	0.8085	0.0571	1 (Part 3)

60	564	197.5	0.9452	0.9457	0.0549	1 (Part 3)
61	566	198	0.9462	0.9461	0.0171	1 (Part 3)
62	568	198.5	0.9472	0.9466	0.0635	1 (Part 3)
63	570	199	0.9482	0.9478	0.0482	1 (Part 3)
64	572	199.5	0.9492	0.9497	0.0593	1 (Part 3)
65	274	900	0.1474	0.1491	1.1811	2
66	276	900.5	0.1527	0.1538	0.7760	2
67	278	901	0.1580	0.1587	0.4275	2
68	280	901.5	0.1635	0.1637	0.1314	2
69	282	902	0.1690	0.1688	0.1166	2
70	284	902.5	0.1747	0.1741	0.3207	2
71	286	903	0.1804	0.1795	0.4846	2
72	288	903.5	0.1862	0.1850	0.6121	2
73	290	904	0.1920	0.1907	0.7069	2
74	292	904.5	0.1980	0.1964	0.7725	2
75	331	997.5	0.3420	0.3437	0.4939	2
76	333	998	0.3497	0.3509	0.3580	2
77	335	998.5	0.3574	0.3582	0.2157	2
78	337	999	0.3652	0.3655	0.0676	2
79	339	999.5	0.3731	0.3728	0.0856	2
80	300	500	0.2071	0.2079	0.4095	2
81	302	500.5	0.2132	0.2137	0.2410	2
82	304	501	0.2193	0.2196	0.1074	2
83	306	501.5	0.2256	0.2256	0.0095	2
84	308	502	0.2319	0.2317	0.0519	2
85	274	200.5	0.1953	0.1974	1.0994	2
86	276	201	0.2025	0.2030	0.2442	2
87	278	201.5	0.2099	0.2088	0.4918	2
88	280	202	0.2173	0.2148	1.1151	2
89	282	202.5	0.2247	0.2211	1.6320	2
90	331	200.5	0.4411	0.4417	0.1345	2
91	333	201	0.4502	0.4514	0.2661	2
92	335	201.5	0.4592	0.4609	0.3586	2
93	337	202	0.4684	0.4703	0.4109	2
94	339	202.5	0.4775	0.4795	0.4221	2
95	331	1000.5	0.3416	0.3443	0.7907	3
96	333	1001	0.3493	0.3515	0.6428	3
97	335	1001.5	0.3570	0.3588	0.4881	3
98	337	1002	0.3648	0.3660	0.3272	3
99	339	1002.5	0.3727	0.3733	0.1611	3
100	274	1995	0.3725	0.3731	0.1651	3

101	276	1995.5	0.3830	0.3832	0.0645	3
102	278	1996	0.3936	0.3935	0.0148	3
103	280	1996.5	0.4043	0.4040	0.0704	3
104	282	1997	0.4150	0.4146	0.1004	3
105	284	1997.5	0.4259	0.4255	0.1037	3
106	286	1998	0.4368	0.4365	0.0796	3
107	288	1998.5	0.4479	0.4477	0.0280	3
108	290	1999	0.4590	0.4592	0.0504	3
109	292	1999.5	0.4702	0.4709	0.1549	3
110	330	1500	0.4641	0.4641	0.0057	3
111	332	1500.5	0.4736	0.4738	0.0367	3
112	334	1501	0.4833	0.4833	0.0118	3
113	336	1501.5	0.4930	0.4927	0.0524	3
114	338	1502	0.5028	0.5022	0.1200	3
115	341	900	0.3647	0.3624	0.6247	4
116	343	900.5	0.3719	0.3700	0.5113	4
117	345	901	0.3791	0.3775	0.4022	4
118	347	901.5	0.3863	0.3852	0.2957	4
119	349	902	0.3936	0.3928	0.1902	4
120	351	902.5	0.4009	0.4006	0.0842	4
121	353	903	0.4082	0.4083	0.0240	4
122	355	903.5	0.4156	0.4161	0.1355	4
123	357	904	0.4230	0.4240	0.2515	4
124	359	904.5	0.4304	0.4320	0.3728	4
125	426	997.5	0.7013	0.7011	0.0382	4
126	428	998	0.7088	0.7084	0.0491	4
127	430	998.5	0.7162	0.7159	0.0378	4
128	432	999	0.7236	0.7236	0.0010	4
129	434	999.5	0.7310	0.7314	0.0643	4
130	341	200.5	0.4929	0.4898	0.6261	4
131	343	201	0.5005	0.4987	0.3611	4
132	345	201.5	0.5081	0.5075	0.1246	4
133	347	202	0.5156	0.5160	0.0831	4
134	349	202.5	0.5230	0.5244	0.2619	4
135	426	200.5	0.7650	0.7678	0.3596	4
136	428	201	0.7689	0.7673	0.2095	4
137	430	201.5	0.7727	0.7689	0.4907	4
138	432	202	0.7765	0.7741	0.3044	4
139	434	202.5	0.7801	0.7839	0.4886	4
140	416	1995	1.1779	1.1778	0.0077	5
141	418	1995.5	1.1886	1.1885	0.0080	5

142	420	1996	1.1993	1.1993	0.0032	5
143	422	1996.5	1.2100	1.2100	0.0024	5
144	424	1997	1.2206	1.2206	0.0043	5
145	426	1997.5	1.2312	1.2312	0.0005	5
146	428	1998	1.2417	1.2417	0.0018	5
147	430	1998.5	1.2523	1.2524	0.0116	5
148	432	1999	1.2627	1.2628	0.0004	5
149	434	1999.5	1.2732	1.2732	0.0003	5
150	341	1000.5	0.3796	0.3802	0.1592	5
151	343	1001	0.3875	0.3875	0.0020	5
152	345	1001.5	0.3955	0.3949	0.1434	5
153	347	1002	0.4034	0.4023	0.2654	5
154	349	1002.5	0.4113	0.4098	0.3684	5
155	341	1500	0.5169	0.5153	0.3026	5
156	343	1500.5	0.5267	0.5254	0.2628	5
157	345	1501	0.5365	0.5358	0.1420	5
158	347	1501.5	0.5463	0.5466	0.0625	5
159	349	1502	0.5560	0.5580	0.3519	5
160	523	1995	2.3264	2.3266	0.0092	6
161	525	1995.5	2.3351	2.3352	0.0058	6
162	527	1996	2.3438	2.3438	0.0021	6
163	529	1996.5	2.3524	2.3523	0.0011	6
164	531	1997	2.3609	2.3608	0.0032	6
165	533	1997.5	2.3694	2.3693	0.0041	6
166	535	1998	2.3778	2.3777	0.0032	6
167	537	1998.5	2.3862	2.3862	0.0008	6
168	539	1999	2.3945	2.3946	0.0034	6
169	541	1999.5	2.4027	2.4030	0.0090	6
170	562	1997	2.4764	2.4763	0.0035	6
171	564	1997.5	2.4840	2.4842	0.0050	6
172	566	1998	2.4916	2.4919	0.0089	6
173	568	1998.5	2.4992	2.4994	0.0079	6
174	570	1999	2.5067	2.5068	0.0018	6
175	572	1999.5	2.5142	2.5139	0.0098	6
176	436	1800	1.6953	1.6943	0.0570	6
177	438	1800.5	1.7068	1.7068	0.0019	6
178	440	1801	1.7182	1.7193	0.0640	6
179	442	1801.5	1.7295	1.7317	0.1306	6
180	444	1802	1.7407	1.7442	0.2030	6
181	480	1900	2.0266	2.0263	0.0113	6
182	482	1900.5	2.0364	2.0365	0.0030	6

183	484	1901	2.0463	2.0465	0.0107	6
184	486	1901.5	2.0560	2.0563	0.0118	6
185	488	1902	2.0657	2.0658	0.0062	6
186	436	1600	1.5005	1.4983	0.1429	6
187	438	1600.5	1.5113	1.5113	0.0049	6
188	440	1601	1.5221	1.5236	0.0958	6
189	442	1601.5	1.5328	1.5353	0.1613	6
190	444	1602	1.5434	1.5464	0.1936	6
191	480	1700	1.8098	1.8091	0.0427	6
192	482	1700.5	1.8192	1.8185	0.0353	6
193	484	1701	1.8284	1.8281	0.0168	6
194	486	1701.5	1.8376	1.8378	0.0134	6
195	488	1702	1.8467	1.8477	0.0562	6
196	564	950	1.4150	1.4147	0.0181	6
197	566	950.5	1.4199	1.4201	0.0098	6
198	568	951	1.4249	1.4252	0.0226	6
199	570	951.5	1.4298	1.4301	0.0210	6
200	572	952	1.4347	1.4347	0.0061	6
201	436	200.5	0.8464	0.8294	2.0028	6
202	438	201	0.8501	0.8453	0.5660	6
203	440	201.5	0.8539	0.8563	0.2860	6
204	442	202	0.8576	0.8609	0.3866	6
205	444	202.5	0.8613	0.8591	0.2498	6
206	564	200.5	1.0276	1.0253	0.2285	6
207	566	201	1.0295	1.0293	0.0263	6
208	568	201.5	1.0315	1.0320	0.0553	6
209	570	202	1.0334	1.0339	0.0460	6
210	572	202.5	1.0353	1.0351	0.0206	6

Table 4.2: Generic fugacity coefficients & errors

4.2. Discussion

After several tries, a quite precise neural network (NN) is able to be designed. This NN model basically built from 210 data (table 4.1), each taken from temperature, pressure, and fugacity coefficient of CO_2. From these 210 data, 70% of them (147 data) are randomly used for train purpose and remaining 30% (63 data) are used for validation purpose. Based on fig 4.1, the NN model consists of 2 inputs (temperature and pressure), 2 layers (1st and 2nd hidden layers), and 1 output (fugacity coefficient). It is understood generally that an improper set-up for the hidden neurons may be a difficulty for the ANN to converge to the best state in application. Besides, there are 7 neurons have been used in 1st hidden layer and they are all connected to 13 neurons in 2nd hidden layer.

There are no rigorously heretical principles for choosing the proper network topology, so different structures were tested in order to obtain the optimal hidden neurons and training cycles. In addition, both hidden layers have applied similar transfer function in order for signal processing, which is log-sigmoid transfer function. One of the reasons for using these transformation functions is the ease of evaluating the derivatives that is required for minimization of the error function. By this function, output values have the range from 0 to 1 and therefore output data have been normalized in the range of [0, 1] by following equation:

$$T_{new} = \frac{T_{old} - T_{\min}}{T_{\max} - T_{\min}}$$

Trainlm (Levenberg-Marquardt backpropagation) is used in this project, combined with learngdm (gradient descent with momentum). It is because it is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. After the NN is completely trained, it achieves its highest performance at epoch is equal to 445, which is shown in term of mean-squared error (MSE). Its value is about 6.3027x10^-6, shown in fig 4.2. Also, the performance of NN can also be measured by interpreting fig 4.3, which are regression plots for the outputs. Based on these figures, the simulation outputs can be defined as 'well-trained' since the value of regression is approaching one.

In the other hand, fig 4.4 shows 3D plot for 2 inputs-1 output system. Also, there are 'dots' in this plot which represent the 210 data of fugacity coefficients at specific temperature and pressure.

The next step after completion of model is to get values for weights as well as biases for each layer. This step is vital in order to achieve the final objective of this project which is to produce a mathematical model. This is because the relationship of the weights and biases with generic output can be described as (in MATLAB command):

Output=purelin∑(LW2.Sigma(logsig∑(logsig∑(LW1(IW.p+B1)))+B2+B3)

Where:

p=input (Temperature & Pressure)

IW=Weight for 1st hidden layer

LW1=Weight for 2nd hidden layer

LW2=Weight for output layer

B1=Bias for 1st hidden neuron

B2=Bias for 2nd hidden neuron

B3=Bias for output neuron

Logsig=Log-sigmoid transfer function

Purelin=Linear transfer function

These all values of parameters above can be referred to fig 4.5-4.7. The complete model can be referred to Appendix 2 in 'Appendices' section. Also, the simulation outputs for this final mathematical model as well as percentage of error can be found in table 4.2. Based on percentage of error, this model generates average error of about 0.16% and max error is 2.00%. Besides, as mentioned before, MSE for this model is about 6.3027x10^-6.

CHAPTER 5

RECOMMENDATION AND CONCLUSION

5.1. Recommendation

Several suggestions can be done to further improve the scope of the project and accuracy of the results:

1. Neural Network modeling of CO_2 fugacity coefficient using MATLAB can be improved in term of error by applying more inputs into the model. However, there will be a significant increment on time consumption.
2. Ranges of temperature and pressure should be reduced in order to increase performance and accuracy. This, of course, will reduce scope of study however, will benefit the accuracy and performance of model
3. Other softwares aside of MATLAB, should be given a try to do modeling on Neural Network. Hence, we can compare the Neural Network model itself as well as the softwares used.

5.2. Conclusion

Measuring the absorption of CO_2 gas in solvents, (in this case pure water) is a vital requirement for most industries in current time. Even though model proposed by Duan and Sun (2003) is able to predict the absorption of CO_2 in term of molality (eqn 31), it is dependent on value of fugacity coefficient of CO_2, ϕ_{CO2}. This matter will becomes a huge problem as ϕ_{CO2} need to be calculated iteratively, based on eqn 32. Thus, the intention of this project is to solve the due problem, which by producing a mathematical model, as a replacement for eqn 32. In order to achieve this target, there 3 objectives need to be done completely. For the 1st objective, the author has to build a Neural Network model, to correlate the inputs and outputs of eqn 32. This objective has been successfully done by using MATLAB software, which this Neural Network model manage to set its performance at 6.3027x10^-6 in term of mean-squared error (MSE) based on 210 data. Next, the correlations of this model need to be generated in order to produce a mathematical model of ϕ_{CO2}. These correlations are represented by values of weights as well as biases inside Neural Network model. Lastly, the 3rd objective is the main objective of this project, which is to design a mathematical model to calculate ϕ_{CO2} with similar accuracy with available model which is eqn 32. Ultimately, the final objective is successfully achieved at the end of the project. The author manages to correlate the weights and biases to produce output, which is based on this MATLAB formula:

$$\text{Output}=\text{purelin}\sum(\text{LW2.Sigma}(\text{logsig}\sum(\text{logsig}\sum(\text{LW1}(\text{IW.p}+\text{B1})))+\text{B2}+\text{B3})$$

Where, all the parameters have been elaborated before in 'Discussion' section. Also, this final mathematical can be referred to Appendix 2 in 'Appendices' section, the last part of this report.

References

Bibliography

Binay, K. D. (2007). Gas Absorption and Stripping. In *Principles of Mass Transfer and Separation Process* (pp. 257-261). New Delhi: Prentice-Hall of India.

Duan, Z., & Sun, R. (2003). An improved model calculating CO2 solubility in pure water and aqueous NaCl solutions from 273 to 533 K and from 0 to 2000 bar. *Chem. Geol. 193* , 257–271.

Duan, Z., Sun, R., & Chen Zhu Chou, I.-M. (2005). An improved model for the calculation of CO2 solubility in aqueous solutions containing Na+, K+, Ca2+, Mg2+, Cl-, and SO42-. *Marine Chemistry 98 (2006)* , 131–139.

Kaszuba, J., Janecky, D., & Snow, M. (2003). Carbon dioxide reaction processes in a model brine aquifer at 200 °C and 200 bars: implications for geologic sequestration of carbon. *Appl. Geochem. 18* , 1065–1080.

Piotrowski, K., Piotrowski, J., & Schlesinger, J. (2003). Modelling of complex liquid/vapour equilibria in the urea synthesis process with the use of artificial neural network. *Chemical Engineering and Processing 42* , 285-289.

S.J.T.Hangx. (2005). Behaviour of the CO2-H2O system and preliminary mineralisation model and experiments. *CATO Workpackage WP 4.1* .

Spycher, N., Pruess, K., & Ennis-King, J. (2003). CO2–H2O mixtures in the geological sequestration of CO2: I. Assessment and calculation of mutual solubilities from 12 to 100 8C and up to 600 bar. *Geochim. Cosmochim. Acta 67* , 3015-3031.

Xu, T., Apps, J., & Pruess, K. (2004). Numerical simulation of CO2 disposal by mineral trapping in deep aquifers. *Appl. Geochem. 19* , 917-936.

APPENDICES

Appendix 1: MATLAB Neural Network Coding

```
%=========================================================================
%=========================================================================
%..... CO2-H2O FUGACITY MODELING USING NEURAL NETWORK .....
%..... Final Year Project .....
%..... Author: Mohd Syafiq Bin Mohd Hassan .....
%..... ID: 10785 .....
%..... Supervised by: Ir Dr Abdul Halim Shah Maulud .....
%=========================================================================
%=========================================================================

clear all;
clc;
close all;
rand('twister',20);
echo off;
format long

% Data retrieved:
load fypdataset6;
p1 = temperature;
p2 = pressure;
p3 = [p1;p2];
t = fugacitycoefficients;

% NN model designed:
net = newff(minmax(p3),[7 13 1],{'logsig' 'logsig'
'purelin'},'trainlm','learngdm');
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.3;
net.divideParam.testRatio = 0;
net.performFcn = 'mse';
net.trainFcn = 'trainlm';
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-7;
net.trainParam.show = 1;

% NN model trained:
[net,tr] = trainlm(net,p3,t);
a = sim(net,p3);
plotperform(tr);
plotregression(t,a);

%=========================================================================

% Plotting(mesh) 3d graph for target val(t):
% Determine the minimum and the maximum x and y values:
xmin = min(p1); ymin = min(p2);
xmax = max(p1); ymax = max(p2);
```

```
% Define the resolution of the grid:
xres=30;
yres=30;

% Define the range and spacing of the x- and y-coordinates,
% and then fit them into X and Y
xv = linspace(xmin, xmax, xres);
yv = linspace(ymin, ymax, yres);
[Xinterp,Yinterp] = meshgrid(xv,yv);

% Calculate Z in the X-Y interpolation space, which is an
% evenly spaced grid:
Zinterp = griddata(p1,p2,t,Xinterp,Yinterp);

% Generate the mesh plot:
figure
mesh(Xinterp,Yinterp,Zinterp)
colormap(cool(8))
xlabel('Temperature(K)')
ylabel('Pressure(bar)')
zlabel('Fugacity Coefficient')
title('Experiment')

% Generate data point on the same axes with specified properties:
hold on
plot3(p1,p2,t,'marker','o','markerfacecolor','r','linestyle','none')
hidden off

%=========================================================================

%=========================================================================

% Plotting(mesh) 3d graph for simulation val(a):
% Determine the minimum and the maximum x and y values:
xmin = min(p1); ymin = min(p2);
xmax = max(p1); ymax = max(p2);

% Define the resolution of the grid:
xres=30;
yres=30;

% Define the range and spacing of the x- and y-coordinates,
% and then fit them into X and Y
xv = linspace(xmin, xmax, xres);
yv = linspace(ymin, ymax, yres);
[Xinterp,Yinterp] = meshgrid(xv,yv);

% Calculate Z in the X-Y interpolation space, which is an
% evenly spaced grid:
Zinterp = griddata(p1,p2,a,Xinterp,Yinterp);

% Generate the mesh plot:
figure
```

```
mesh(Xinterp,Yinterp,Zinterp)
colormap(cool(8))
xlabel('Temperature(K)')
ylabel('Pressure(bar)')
zlabel('Fugacity Coefficient')
title('Simulation')

% Generate data point on the same axes with specified properties:
hold on
plot3(p1,p2,a,'marker','o','markerfacecolor','r','linestyle','none')
hidden off
```

```
%=========================================================================
```

```
%=========================================================================

% Plotting(mesh) 3d graph for error val(MSE):
% Determine error values:
e = t-a;
mse = ((e).^2)./90;

% Determine the minimum and the maximum x and y values:
xmin = min(p1); ymin = min(p2);
xmax = max(p1); ymax = max(p2);

% Define the resolution of the grid:
xres=30;
yres=30;

% Define the range and spacing of the x- and y-coordinates,
% and then fit them into X and Y
xv = linspace(xmin, xmax, xres);
yv = linspace(ymin, ymax, yres);
[Xinterp,Yinterp] = meshgrid(xv,yv);

% Calculate Z in the X-Y interpolation space, which is an
% evenly spaced grid:
Zinterp = griddata(p1,p2,mse,Xinterp,Yinterp);

% Generate the mesh plot:
figure
mesh(Xinterp,Yinterp,Zinterp)
colormap(cool(8))
xlabel('Temperature(K)')
ylabel('Pressure(bar)')
zlabel('MSE')
title('MSE')
pause(3)
drawnow;
commandwindow;
pause(2)
```

```
%=========================================================================
```

```
%==========================================================================

% Show values for weights and bias for each layer(used for designing eqn):
fprintf('1)Weight value(s) for hidden layer (IW):\n');
net.iw{1}
fprintf('\n');
pause(2)
fprintf('2)Bias value(s) for 1st hidden layer(B1):\n');
net.b{1}
fprintf('\n');
pause(2)
fprintf('3)Weight value(s) for 2nd hidden layer(LW1):\n');
net.lw{2}
fprintf('\n');
pause(2)
fprintf('4)Bias value(s) for 2nd hidden layer(B2):\n');
net.b{2}
fprintf('\n');
pause(2)
fprintf('5)Weight value(s) for output layer(LW2):\n');
net.lw{3,2}
fprintf('\n');
pause(2)
fprintf('6)Bias value(s) for output layer(B3):\n');
net.b{3}
fprintf('\n');
pause(2)
fprintf('These values can be applied to general equation in order to build
the model.\n');
% General
fprintf('General Equation:\n\n');
fprintf('          Output =
purelinSigma(LW2.Sigma(logsigSigma(logsigSigma(LW1(IW.p + b1))) +
b2)+b3).\n\n');
fprintf('Applying these parameters into the equation, the final equation
becomes:\n\n');
fprintf('    Output = purelin((-2.522798774243832)*nb1+(-
2.151830526755441)*nb2+(0.870111318008102)*nb3+(4.091389190275042)*nb4+(1.834
941800169173)*nb5+(-2.800265371663276)*nb6+(3.211789052186852)\n');
fprintf('              *nb7+(2.544785769772646)*nb8+(-
1.782029265926753)*nb9+(-4.579260860606725)*nb10+(1.521210709211187)*nb11+(-
0.776079440084530)*nb12+(2.170422039757039)*nb13+0.936329979271203)\n\n');
fprintf('    where :\n\n');

fprintf('          nb1  =
logsig((3.280647471685330*na1)+(3.599753676215916*na2)+(1.709768509537894*na3
)+(5.158527199688285*na4)+(-5.577883891717301*na5)+(2.560348515818909*na6)+(-
5.216655641412578*na7)+(-6.057253045424375))\n');
fprintf('          nb2  = logsig((-3.221546655524049*na1)+(-
0.590561548421666*na2)+(0.336708480990765*na3)+(2.079356181589410*na4)+(-
1.933199468404810*na5)+(-3.759014181644583*na6)+(-
5.501269159559889*na7)+(9.004156563742827))\n');
fprintf('          nb3  = logsig((-
4.547979243687573*na1)+(0.241107408124719*na2)+(-
```

```
6.789526009991142*na3)+(2.433313699841999*na4)+(0.272801571386701*na5)+(3.420
601154891709*na6)+(3.074797382351063*na7)+(0.244710131648908))\n');
fprintf('          nb4  =
logsig((4.923895609273885*na1)+(3.410181389643968*na2)+(-
2.960413425278762*na3)+(3.783311197328923*na4)+(-5.339574425721900*na5)+(-
4.109800936362563*na6)+(0.169283704711115*na7)+(-0.099713662442774))\n');
fprintf('          nb5  = logsig((-
4.502920858009740*na1)+(1.873196377226982*na2)+(-
9.173909672749835*na3)+(3.301381806785101*na4)+(2.090107188507103*na5)+(-
3.406371050227587*na6)+(5.146526301139264*na7)+(2.041072602020472))\n');
fprintf('          nb6  = logsig((-3.462393642821834*na1)+(-
3.421956366332052*na2)+(2.169869949326351*na3)+(-
0.272552109551378*na4)+(5.302816697522779*na5)+(-
0.318840686651158*na6)+(1.177267241346895*na7)+(3.622977980692767))\n');
fprintf('          nb7  = logsig((3.297271443058438*na1)+(-
4.696085594377566*na2)+(-
2.175489607694775*na3)+(1.693625857007005*na4)+(4.990612777617213*na5)+(-
3.014497718584487*na6)+(2.138370756873609*na7)+(3.986817421183496))\n');
fprintf('          nb8  =
logsig((4.512321424278532*na1)+(2.654320860893895*na2)+(3.729721400917295*na3
)+(-0.064299986826718*na4)+(4.602413323578574*na5)+(8.132191259109424*na6)+(-
0.630211298933114*na7)+(-3.784195501300252))\n');
fprintf('          nb9  =
logsig((1.457089686433606*na1)+(1.771861258803943*na2)+(-
0.354772016056593*na3)+(-7.784569227442063*na4)+(-3.465815201417499*na5)+(-
2.490629361423565*na6)+(1.821802100003521*na7)+(-0.790097452418069))\n');
fprintf('          nb10 = logsig((-3.125285302265449*na1)+(-
1.416146205846159*na2)+(-
5.086890135984253*na3)+(2.062993444358582*na4)+(1.958286366365898*na5)+(-
4.660240454004780*na6)+(2.420553268273415*na7)+(5.430310795211947))\n');
fprintf('          nb11 = logsig((5.527129558174433*na1)+(-
0.117426979716782*na2)+(4.077436373789220*na3)+(-
0.064929095284252*na4)+(1.900022507896190*na5)+(-5.794305007162077*na6)+(-
4.767416765556738*na7)+(4.078710221491510))\n');
fprintf('          nb12 = logsig((0.381666480017604*na1)+(-
2.884781157162306*na2)+(-
3.445662336288738*na3)+(1.460584783050640*na4)+(2.602678431886161*na5)+(9.905
072435811846*na6)+(0.936724387258411*na7)+(-3.390056852304513))\n');
fprintf('          nb13 = logsig((1.340606180071120*na1)+(-
2.736134545609030*na2)+(7.563037648372469*na3)+(-0.895751819321215*na4)+(-
5.283837776314017*na5)+(7.390123052866978*na6)+(-6.887327187841041*na7)+(-
0.060153682217420))\n\n');

fprintf('          na1  =
logsig((0.024314886198726*T)+(0.009201900343075*P)+(-
29.671673019489301))\n');
fprintf('          na2  = logsig((0.027122406187101*T)+(-
0.028571475158764*P)+(-27.658270012547863))\n');
fprintf('          na3  = logsig((0.103436929228812*T)+(-
0.136804741773623*P)+(-17.212749648478820))\n');
fprintf('          na4  = logsig((0.030422688264001*T)+(-
0.003984725895334*P)+(-7.716523562434156))\n');
fprintf('          na5  = logsig((-
0.060150865352321*T)+(0.088349019057763*P)+(16.348496547615362))\n');
fprintf('          na6  =
logsig((0.018744145443540*T)+(0.002423238438437*P)+(-
10.585679290653822))\n');
```

```
fprintf('         na7  = logsig((-
0.400073802631029*T)+(0.091304690342938*P)+(-8.219902175068270))\n\n');
pause(3)
fprintf('
==========================================================================
==\n');
fprintf('         ============== Please press "Enter" to proceed for model
testing ==============\n');
fprintf('
==========================================================================
==\n');
pause

%==========================================================================

% Model testing
clc;
fprintf('=================================================\n');
fprintf('========== Model Testing section ==============\n');
fprintf('=================================================\n\n');
pause(1.5)

count = input('Please enter number of desired data = ');
for i = 1:count
    fprintf('Enter Temperature(K) %d( 273K<Temperature<573K )',i);
    T(i) = input(' = ');
    while (T(i)<=273) || (T(i)>=573)
        T(i) = input('Temperature not within valid range. Please re-enter =
');
    end
end

for i = 1:count
    fprintf('Enter Pressure(bar) %d( 0<Pressure<2000bar )',i);
    P(i) = input(' = ');
    while (P(i)<=0) || (P(i)>=2000)
        P(i) = input('Pressure not within valid range. Please re-enter = ');
    end
end

for i = 1:count
    na1(i) = logsig((0.024314886198726*T(i))+(0.009201900343075*P(i))+(-
29.671673019489301));
    na2(i) = logsig((0.027122406187101*T(i))+(-0.028571475158764*P(i))+(-
27.658270012547863));
    na3(i) = logsig((0.103436929228812*T(i))+(-0.136804741773623*P(i))+(-
17.212749648478820));
    na4(i) = logsig((0.030422688264001*T(i))+(-0.003984725895334*P(i))+(-
7.716523562434156));
    na5(i) = logsig((-
0.060150865352321*T(i))+(0.088349019057763*P(i))+(16.348496547615362));
    na6(i) = logsig((0.018744145443540*T(i))+(0.002423238438437*P(i))+(-
10.585679290653822));
    na7(i) = logsig((-0.400073802631029*T(i))+(0.091304690342938*P(i))+(-
8.219902175068270));
```

```
    nb1(i) =
logsig((3.280647471685330*na1(i))+(3.599753676215916*na2(i))+(1.7097685095378
94*na3(i))+(5.158527199688285*na4(i))+(-
5.577883891717301*na5(i))+(2.560348515818909*na6(i))+(-
5.216655641412578*na7(i))+(-6.057253045424375));
    nb2(i) = logsig((-3.221546655524049*na1(i))+(-
0.590561548421666*na2(i))+(0.336708480990765*na3(i))+(2.079356181589410*na4(i
))+(-1.933199468404810*na5(i))+(-3.759014181644583*na6(i))+(-
5.501269159559889*na7(i))+(9.004156563742827));
    nb3(i) = logsig((-4.547979243687573*na1(i))+(0.241107408124719*na2(i))+(-
6.789526009991142*na3(i))+(2.433313699841999*na4(i))+(0.272801571386701*na5(i
))+(3.420601154891709*na6(i))+(3.074797382351063*na7(i))+(0.244710131648908))
;
    nb4(i) = logsig((4.923895609273885*na1(i))+(3.410181389643968*na2(i))+(-
2.960413425278762*na3(i))+(3.783311197328923*na4(i))+(-
5.339574425721900*na5(i))+(-
4.109800936362563*na6(i))+(0.169283704711115*na7(i))+(-0.099713662442774));
    nb5(i) = logsig((-4.502920858009740*na1(i))+(1.873196377226982*na2(i))+(-
9.173909672749835*na3(i))+(3.301381806785101*na4(i))+(2.090107188507103*na5(i
))+(-
3.406371050227587*na6(i))+(5.146526301139264*na7(i))+(2.041072602020472));
    nb6(i) = logsig((-3.462393642821834*na1(i))+(-
3.421956366332052*na2(i))+(2.169869949326351*na3(i))+(-
0.272552109551378*na4(i))+(5.302816697522779*na5(i))+(-
0.318840686651158*na6(i))+(1.177267241346895*na7(i))+(3.622977980692767));
    nb7(i) = logsig((3.297271443058438*na1(i))+(-4.696085594377566*na2(i))+(-
2.175489607694775*na3(i))+(1.693625857007005*na4(i))+(4.990612777617213*na5(i
))+(-
3.014497718584487*na6(i))+(2.138370756873609*na7(i))+(3.986817421183496));
    nb8(i) =
logsig((4.512321424278532*na1(i))+(2.654320860893895*na2(i))+(3.7297214009172
95*na3(i))+(-
0.064299986826718*na4(i))+(4.602413323578574*na5(i))+(8.132191259109424*na6(i
))+(-0.630211298933114*na7(i))+(-3.784195501300252));
    nb9(i) = logsig((1.457089686433606*na1(i))+(1.771861258803943*na2(i))+(-
0.354772016056593*na3(i))+(-7.784569227442063*na4(i))+(-
3.465815201417499*na5(i))+(-
2.490629361423565*na6(i))+(1.821802100003521*na7(i))+(-0.790097452418069));
    nb10(i) = logsig((-3.125285302265449*na1(i))+(-
1.416146205846159*na2(i))+(-
5.086890135984253*na3(i))+(2.062993444358582*na4(i))+(1.958286366365898*na5(i
))+(-
4.660240454004780*na6(i))+(2.420553268273415*na7(i))+(5.430310795211947));
    nb11(i) = logsig((5.527129558174433*na1(i))+(-
0.117426979716782*na2(i))+(4.077436373789220*na3(i))+(-
0.064929095284252*na4(i))+(1.900022507896190*na5(i))+(-
5.794305007162077*na6(i))+(-4.767416765556738*na7(i))+(4.078710221491510));
    nb12(i) = logsig((0.381666480017604*na1(i))+(-
2.884781157162306*na2(i))+(-
3.445662336288738*na3(i))+(1.460584783050640*na4(i))+(2.602678431886161*na5(i
))+(9.905072435811846*na6(i))+(0.936724387258411*na7(i))+(-
3.390056852304513));
    nb13(i) = logsig((1.340606180071120*na1(i))+(-
2.736134545609030*na2(i))+(7.563037648372469*na3(i))+(-
0.895751819321215*na4(i))+(-
5.283837776314017*na5(i))+(7.390123052866978*na6(i))+(-
6.887327187841041*na7(i))+(-0.060153682217420));
```

```
    o1(i) = purelin((-2.522798774243832)*nb1(i)+(-
2.151830526755441)*nb2(i)+(0.870111318008102)*nb3(i)+(4.091389190275042)*nb4(
i)+(1.834941800169173)*nb5(i)+(-
2.800265371663276)*nb6(i)+(3.211789052186852)*nb7(i)+(2.544785769772646)*nb8(
i)+(-1.782029265926753)*nb9(i)+(-
4.579260860606725)*nb10(i)+(1.521210709211187)*nb11(i)+(-
0.776079440084530)*nb12(i)+(2.170422039757039)*nb13(i)+0.936329979271203);
end
for i = 1:count
    fprintf('Output %d = %f\n',i,o1(i));
end
```

APPENDICES

Appendix 2: Mathematical Model for ϕ_{CO2}

ϕ_{CO2} = purelin((-2.522798774243832)*nb1 + (-2.151830526755441)*nb2 + (0.870111318008102)*nb3 + (4.091389190275042)*nb4 + (1.834941800169173)*nb5 + (-2.800265371663276)*nb6 + (3.211789052186852)*nb7 + (2.544785769772646)*nb8 + (-1.782029265926753)*nb9 + (-4.579260860606725)*nb10 + (1.521210709211187)*nb11 + (-0.776079440084530)*nb12 + (2.170422039757039)*nb13 + 0.936329979271203)

Where:

nb1 = logsig((3.280647471685330*na1) + (3.599753676215916*na2) + (1.709768509537894*na3) + (5.158527199688285*na4) +(-5.577883891717301*na5)+(2.560348515818909*na6)+(-5.216655641412578*na7) + (-6.057253045424375))

nb2 = logsig((-3.221546655524049*na1) + (-0.590561548421666*na2) + (0.336708480990765*na3) + (2.079356181589410*na4)+(-1.933199468404810*na5)+(-3.759014181644583*na6)+(-5.501269159559889*na7) + (9.004156563742827))

nb3 = logsig((-4.547979243687573*na1) + (0.241107408124719*na2) + (-6.789526009991142*na3) + (2.433313699841999*na4) + (0.272801571386701*na5) + (3.420601154891709*na6) + (3.074797382351063*na7) + (0.244710131648908))

```
nb4 = logsig((4.923895609273885*na1) + (3.410181389643968*na2) + (-2.960413425278762*na3) +
(3.783311197328923*na4) + (-5.339574425721900*na5) + (-4.109800936362563*na6) + (0.169283704711115*na7) +
(-0.099713662442774))

nb5 = logsig((-4.502920858009740*na1) + (1.873196377226982*na2) + (-9.173909672749835*na3) +
(3.301381806785101*na4) + (2.090107188507103*na5) + (-3.406371050227587*na6) + (5.146526301139264*na7) +
(2.041072602020472))

nb6 = logsig((-3.462393642821834*na1) + (-3.421956366332052*na2) + (2.169869949326351*na3) +
(-0.272552109551378*na4) + (5.302816697522779*na5) + (-0.318840686651158*na6) + (1.177267241346895*na7) +
(3.622977980692767))

nb7 = logsig((3.297271443058438*na1) + (-4.696085594377566*na2) + (-2.175489607694775*na3) +
(1.693625857007005*na4) + (4.990612777617213*na5) + (-3.014497718584487*na6) + (2.138370756873609*na7) +
(3.986817421183496))

nb8 = logsig((4.512321424278532*na1) + (2.654320860893895*na2) + (3.729721400917295*na3) +
(-0.064299986826718*na4) + (4.602413323578574*na5) + (8.132191259109424*na6) + (-0.630211298933114*na7) +
(-3.784195501300252))

nb9 = logsig((1.457089686433606*na1) +( 1.771861258803943*na2) + (-0.354772016056593*na3) +
(-7.784569227442063*na4) + (-3.465815201417499*na5) + (-2.490629361423565*na6) + (1.821802100003521*na7) +
(-0.790097452418069))
```

```
nb10 = logsig((-3.125285302265449*na1) + (-1.416146205846159*na2) + (-5.086890135984253*na3) +
(2.062993444358582*na4) + (1.958286366365898*na5) + (-4.660240454004780*na6) + (2.420553268273415*na7) +
(5.430310795211947))

nb11 = logsig((5.527129558174433*na1) + (-0.117426979716782*na2) + (4.077436373789220*na3) +
(-0.064929095284252*na4) + (1.900022507896190*na5) + (-5.794305007162077*na6) + (-4.767416765556738*na7) +
(4.078710221491510))

nb12 = logsig((0.381666480017604*na1) + (-2.884781157162306*na2) + (-3.445662336288738*na3) +
(1.460584783050640*na4) + (2.602678431886161*na5) + (9.905072435811846*na6) + (0.936724387258411*na7) +
(-3.390056852304513))

nb13 = logsig((1.340606180071120*na1) + (-2.736134545609030*na2) + (7.563037648372469*na3) +
(-0.895751819321215*na4) + (-5.283837776314017*na5) + (7.390123052866978*na6) + (-6.887327187841041*na7) +
(-0.060153682217420))

na1 = logsig((0.024314886198726*T) + (0.009201900343075*P) + (-29.671673019489301))

na2 = logsig((0.027122406187101*T) + (-0.028571475158764*P) + (-27.658270012547863))

na3 = logsig((0.103436929228812*T) + (-0.136804741773623*P) + (-17.212749648478820))

na4 = logsig((0.030422688264001*T) + (-0.003984725895334*P) + (-7.716523562434156))

na5 = logsig((-0.060150865352321*T) + (0.088349019057763*P) + (16.348496547615362))
```

```
na6 = logsig((0.018744145443540*T) + (0.002423238438437*P) + (-10.585679290653822))

na7 = logsig((-0.400073802631029*T) + (0.091304690342938*P) + (-8.219902175068270))

P = Pressure (bar)

T = Temperature (K)

Purelin() = MATLAB-command for linear transfer function

Logsig() = MATLAB-command for log-sigmoid transfer function
```