UNIVERSITI
TEKNOLOGI
PETRONAS

Estimating Pressure Drop in Vertical Wells Using Group Method of Data Handling (GMDH) Approach; a Comparative Study

by

Musaab Mohamed Ahmed Alamin

Dissertation submitted in partial fulfilment of

the requirements for the

Bachelor of Engineering (Hons)

(Petroleum Engineering)

May 2013

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Estimating Pressure Drop in Vertical Wells Using Group Method of Data Handling (GMDH) Approach; a Comparative Study

by

Musaab Mohamed Ahmed Alamin

A project dissertation submitted to the
Petroleum Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(PETROLEUM ENGINEERING)

Approved by,

____
(Dr. Mohammed Abdalla Ayoub)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

May 2013

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.

____

MUSAAB MOHAMED AHMED ALAMIN

ABSTRACT

A reliable estimation of the pressure drop in well tubing is essential for the solution of a number of important production engineering and reservoir analysis problems. Different methods have been discussed and analyzed in the literature. This includes the basics of each method, the variables used and the limitations and constraints.

This project aims to construct a tool that can estimate the pressure drop in vertical well using the minimum possible variables. In this project group method of data handling (GMDH) approach is used in order to build the model. And for the optimization of the model, Trend analysis is also used for the sake of having a physically sound model.

The developed model GMDH has shown an outstanding results and it has outperformed all empirical correlations and mechanistic models which have been used in the comparison. The analysis of the results also confirmed with the testing set which has not seen by the GMDH during the development of the model which could still achieve an accurate estimation of the pressure drop.

The GMDH model is developed and the objective is successfully achieved. Moreover, the simplicity and good functionality of the model made it a better choice when it comes to predict a pressure drop in any multiphase vertical well.

ACKNOWLEDGEMENTS

All the thanks firstly go to Allah who helped me completing this project in a proper way, and within the time, my thanks also go to my parents who helped me along during the entire study life. And my deepest gratitude shall be given to my supervisor Dr. Mohammed Abdalla Ayoub who helps me in order to complete this project with his full support, guidance and sharing knowledge, working on this project has given me a very good experience of how to obtain knowledge and work as engineers work. Not to be forgotten all my friends and colleagues, and also Universiti Teknologi PETRONAS which provided the facilities to complete this work.

TABLE OF CONTENTS

Chapter 1 1

INTRODUCTION 1

1.1 Project Background 1

1.2 Group method of data handling (GMDH) Algorithms 2

1.3 Selection of Independent variables 3

1.4 Problem statement 3

1.5 Objectives 5

1.6 Feasibility of the study 5

Chapter 2 6

LITERATURE REVIEW 6

2.1 Overview 6

2.2 Empirical Correlations 8

2.3 Mechanistic Models 10

2.4 Artificial Neural Networks 11

Chapter 3 13

RESEARCH METHODOLOGY 13

3.1 Overview 13

3.2 Data Gathering & Processing 14

3.2.1 Partitioning 15

3.3 Building GMDH Model 16

3.4 Software Used 16

3.5 Trend analysis 17

3.6 Statistical Error Analysis 17

3.7 Graphical Error Analysis 17

3.7.1 Cross-plots 18

3.7.2 Error Distribution 18

3.8 Limitations of the Model 18

Chapter 4 19

RESULTS AND DISCUSSION 19

4.1 Development of the GMDH model 19

4.1.1 Introduction 19

4.1.2 Summary of the Model’s Equation 20

4.1.3 GMDH Model Optimization 21

4.2 Trend Analysis for the Proposed GMDH Mode 22

4.3 Statistical Error Analysis for the Proposed GMDH Model against Other Investigated Models 25

4.4 Graphical Error Analysis for the Proposed GMDH Model against Other Investigated Models 25

4.4.1 Cross Plots of GMDH Model against Investigated Models 26

4.4.2 Error Distribution of GMDH Model against Investigated Models 33

4.5 Discussion of the Results 35

CHAPTER 5 39

CONCLUSION AND RECOMMENDATION 39

5.1 Conclusions 39

5.2 Recommendations 40

REFERENCES 41

APPENDIX A - Statistical Error Equation 46

APPENDIX B- GMDH CODE 47

LIST OF FIGURES

Figure 3.1 Modelling Construction Process 14

Figure 4.1 Schematic Diagram Of The Proposed GMDH Model Topology 20

Figure 4.2 Effect of Gas Rate on Pressure Drop 22

Figure 4.3 Effect of Water Rate on Pressure Drop 23

Figure 4.4 Effect of Oil Rate on Pressure Drop 23

Figure 4.5 Effect of Depth “Pipe Length” on Pressure Drop 24

Figure 4.6 Effect of Oil Gravity on Pressure Drop 24

Figure 4.7 Cross plot of pressure drop for Training set (GMDH Model) 27

Figure 4.8 Cross plot of pressure drop for Validation set (GMDH Model) 27

Figure 4.9 Cross plot of pressure drop for Testing set (GMDH Model) 28

Figure 4.10 Cross plot of pressure drop for Hagedorn & Brown Correlation 28

Figure 4.11 Cross plot of pressure drop for Beggs Brill Correlation 29

Figure 4.12 Cross Plot of Pressure Drop for Orkiszewski Correlation 29

Figure 4.13 Cross plot of pressure drop for Gray Correlation 30

Figure 4.14 Cross plot of pressure drop for Duns & Ros Correlation 30

Figure 4.15 Cross plot of pressure drop for Ansari et al Model 31

Figure 4.16 Cross plot of pressure drop for Mukherjee & Brill Correlation 31

Figure 4.17 Cross plot of pressure drop for Aziz et al Model 32

Figure 4.18 Cross plot of pressure drop for Ayoub Model 32

Figure 4.19 Error Distribution for Training Set 33

Figure 4.20 Error Distribution for Validation Set 34

Figure 4.21 Error Distribution for Testing Set 34

Figure 4.22 Error Distribution for All models including GMDH model 35

Figure 4.23 Average Absolute Relative Error for All Models 36

Figure 4.24 Root Mean Square Error for All Models 37

Figure 4.25 Coefficient of Determination for All Models 37

Figure 4.26 Root Mean Square Error against Standard Deviation for All Models 38

Figure 4.27 Average Absolute Relative Errors against the Confident Of

Determination for All Models 38

<u>LIST OF TABLES</u>

Table 3.1 Statistical Analysis of the Used Data 15

Table 4.1 Statistical Analysis Results of the Proposed GMDH Model 25

Table 4.2 Statistical Analysis Results of the Proposed GMDH Model and Investigate Models 26

Chapter 1

INTRODUCTION

1.1 Project Background

The main factors that cause the pressure drop in any vertical pipe can be a summation of three terms: gravity, friction loss and a momentum pressure drop. (Griffith et al, 1975):

$\Delta P = \Delta P_f + \Delta P_g + \Delta P_m$ ……………………….…………….equation (1.1)

The pressure loss due to the fiction loss, the gravity loss and the momentum loss. The momentum loss (ΔP_m) can be negligible in the literature (Griffith et al, 1975; Hasan and Kabir, 1992; Ansari et al., 1994; Abdul-Majeed & Al-Mashat, 2000). Several methods have been proposed to estimate the pressure drop in vertical wells which produce a mixture of oil and gas. Early researchers used laboratory and/or field data to develop empirical correlation to investigate pressure drop in multi-phase flow. (Duns & Ros, 1963 and Orkiszewski, 1967). And currently, some researchers are using artificial intelligence such as the artificial neural networks to directly predict the pressure drop in any vertical pipe (Ayoub, 2004).

One of the challenges in pressure drop calculation is the determination of the flow regime in vertical pipe. Due to the complexity of multiphase flow, several flow regimes may exist that depend on different factors. Zavareh et al investigation in 1988 using a multiphase flow loop and a photograph showed that when the well is vertical all flow regimes detected is bubble flow for all conditions tested. This bubble flow can be further subdivided as being bubble, dispersed bubble, inverted bubble or Inverted dispersed bubble, based on the bubble size and depending on which phase is continuous.

1.2 Group method of data handling (GMDH) Algorithms

GMDH is a heuristic self-organization method that models the input-output relationship of a complex system. In 1966, the Russian cyberneticist, Prof. Alexey G. Ivakhnenko in the Institute of Cybernetics in Kiev (Ukraine) introduced a technique for constructing an extremely high-order regression-type polynomial. The algorithm, the Group Method of Data Handling (GMDH), builds a multinomial of degree in the hundreds, whereas standard multiple regression becomes bogged down in computation and linear dependence.

The author great generosity of open code sharing has made this method quickly settled in the large number of scientific laboratories worldwide. The basic GMDH algorithm is a procedure of constructing a high order polynomial of the form:

$$y = a + \sum_{i=1}^{m} b_i\, x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} d_{ijk}\, x_i x_j x_k + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m} e_{ijkl}\, x_i x_j x_k x_l + \cdots \text{ .... equation (1.2)}$$

GMDH modelling can be an alternative to artificial neural networks modelling approach that helps overcome many of the artificial neural networks limitations is that based on the self-organizing Group Method of Data Handling (GMDH). Based on the self-organizing group method of data handling (GMDH), this technique uses well-proven optimization criteria for automatically determining the network size and connectivity, and element types and coefficients for the optimum model, thus reducing the modelling effort and the need for user intervention.

The mechanisation of model creation not only lessens the burden on the analyst but also safeguards the model generated from being influenced by human biases and misjudgements. The GMDH model automatically selects influential input parameters and the input-output relationship can be expressed in polynomial form. This enhances explanation capabilities and allows comparison of the resulting data-based machine learning models with existing first principles or empirical models. (Farlow, 1981; Osman & Abdel-Aal, 2002).

1.3 Selection of Independent variables

The main problem with estimation the pressure drop in vertical well is the number & type of independent variables that can affect the pressure drop. The model suggested by Aziz et al (1972) has a total of fifteen independent variables which must be specified as input data in order to calculate the pressure at the wellbore.

A simple linear model has been presented by Gregory (1974) which required only four independent variables to be specified under certain conditions. It can be used in straightforward hand calculations where it gives easier way to calculate the pressure drop than Aziz et al model. Empirical models could not completely take into account all the variables and complexities of multiphase flow and resulted in limited range of applicability. (Takacs, 2001).

Early methods used very rude physical models and many simplifying assumptions, and were usually based on experimental data gathered from laboratory or field measurements. These empirical models could not completely take into account all the variables and complexities of multiphase flow and resulted in correlations of limited applicability

In this project, the number of independent variables will be evaluated by the software to minimize the number of independent variables need to be used in the proposed model.

1.4 Problem statement

Many methods have been proposed to estimate the pressure drop in vertical wells that produce a mixture of oil and gas. The study conducted by Pucknell et al (1993) concludes that none of the traditional multiphase flow correlations works well across the full range of conditions encountered in oil and gas fields. Accurate prediction of pressure drop in vertical wells can be greatly helpful in cost management when it comes to well completions and production optimization.

Most of the vertical pressure drop calculation models were developed for average oilfield fluids and this is why special conditions such as; emulsions, non-Newtonian flow behaviour, excessive scale or wax deposition on the tubing wall, etc. can pose severe problems. Accordingly, predictions in such cases could be doubtful (Takacs, 2001).

The accuracy of estimating the pressure drop in vertical wells has been discussed frequently in the last decades. Although many solutions have been produced but it still can’t be raised to a level to be generally accepted. Early empirical models treated the multiphase flow problem as the flow of a homogeneous mixture of liquid and gas. This approach completely disregarded the well-known observation that the gas phase, due to its lower density, overtakes the liquid phase resulting in “slippage” between the phases. Slippage increases the flowing density of the mixture as compared to the homogeneous flow of the two phases at equal velocities. Because of the poor physical model adopted, calculation accuracy was low for those early correlations. Another reason behind that is the complexity in multiphase flow in the vertical pipes. Where water and oil may have nearly equal velocity, gas have much greater one. As a results, the difference in the velocity will definitely affect the pressure drop.

As measuring the pressure drop in vertical wells is not a practical options due to its high cost. Many methods have been proposed for pressure drop estimation. However, the variation of well conditions from one to another is an obstacle to have general correlation with acceptable range of error.

The parameters affecting the pressure drop are very important for the model generation. While not all fluid data or production data for example are critical, knowing the weight effect of each parameters can lead to a simple and direct estimation for the pressure drop.

1.5 Objectives

This project aims at generating a model which is capable of estimating the pressure drop in vertical wells using the minimum possible number of parameters and compare its performance with of the best current methods.

Other specific objectives of this project can be stated as follow:

a) Revising the best available correlations and defining the correlation parameters.

b) Understating the parameters influencing the pressure drop in vertical pipes.

c) Construct a new model using group method of data handling (GMDH) approach to estimate the pressure drop.

d) Evaluate the model performance by comparing the predicted results against the actual ones.

1.6 Feasibility of the study

This project requires a modelling software in order to conduct a successful study. By using Matlab Software -which is available in UTP-. And the field data required to carry on the project are collected from released actual field data. Hence, the project is clearly feasible to be implemented and results were obtained within the proposed time frame for the project.

The new model is helpful for designing the facilities needed in vertical wells. Besides that, this new model can be generally considered for estimating pressure drop in vertical wells in oil and gas industry because of its simplicity and high accuracy

Chapter 2

LITERATURE REVIEW

2.1 Overview

The existence of multiphase flow and their associated problems have been recognized since 1797.Numerous correlations and equations have been proposed for multiphase flow in vertical, inclined and horizontal wells in the literature. However, most of the significant contributions have been made since 1945 (Palisetti, 1998). Due to the importance of two-phase flow problems, researchers have developed many accurate pressure drop computation methods (Takacs 2001). However, it has not yet been proposed a universal model that can satisfy all well conditions.

The early approaches used the empirical correlation methods such as Hagedorn & Brown (1965) Duns & Ros, (1963), and Orkiszewski (1967). Then the trend shifted into mechanistic modelling methods such as Ansari (1994) and Aziz et al (1972) and lately the researchers has introduced the use of artificial intelligence into the oil and gas industry by using artificial neural networks such as Ayoub (2004) and Mohammadpoor (2010) and many others. The application of factorial design analysis for a well pressure drop modelling has been discussed by Gregory in 1974. The study discussed the proper usage of fractional factorial design analysis which can generate a direct simple linear approximation of the computer model that can be used in the prediction model. In that study, ranges of variables selected was aided by the well data tabulated by Aziz et al (1972). As the remaining input data for the computer program were arbitrarily specified. Gregory (1974) claims that his single linear equation with four independent variable can predict the pressure drop more accurate than the mechanistic model. However, this model proposed by Gregory is only applicable to be used under certain conditions. Gregory suggested to generate similar model by normal regression analysis techniques using the available well data. This may lead to find better and more accurate values for the pressure drop.

Takacs (2001) has collected and summarized the findings of many previous investigation on the accuracy of the different pressure drop calculation models. Statistical parameters of these investigations are shown to be widely scattered and to be of limited use to engineers seeking the most accurate model. The Early methods used very rude physical models and many simplifying assumptions, and were usually based on experimental data gathered from laboratory or field measurements. The era from the early 1980s to the present experienced the emergence of the so-called “mechanistic” models. These apply a modelling approach to the solution of the pressure drop calculation and are founded on a comprehensive description of the basic mechanisms occurring in multiphase flow. Takacs concluded his paper in the following points:

- None of the available vertical multiphase pressure drop calculation models is generally applicable because their prediction errors may considerably vary in the different ranges of the flow parameters.
- There is no “over-all best” calculation method, and all efforts to find one are deemed to fail.
- In spite of the claims found in the literature, the introduction of mechanistic models did not deliver a breakthrough yet because their accuracy not substantially exceed that of the empirical ones.
- Based on a sufficiently great number of experimental data from the oilfield considered, one can determine the optimum pressure drop prediction method for that field.

A different method using two phase fluid flow models is proposed to calculate the pressure drop in vertical and inclined oil wells (Griffith et al, 1975). The study has been implemented to oil and gas wells with an accuracy of about 10 %. Griffith methods is considered as very simple equation. However, some variables have been dropped out of the equation such as; pipe roughness, viscosity for liquid and gas and entrainment effect. The justification for these variables not be considered in his equation is simple as stated by the author. Pipe roughness effect is almost the same as in single phase flow. And the existence of liquid may increase or decrease the friction drop but with low effect and therefore it can be negligible. For the gas and liquid viscosity, they have little of consequence but the error under these conditions would be considerable.

Some correlations have been developed to calculate the pressure drop in special cases such as; deep gas wells (Moradi et al, 2011) and the presence of surfactant in the well (Soni et al, 2009).

2.2 Empirical Correlations

The empirical correlation was created by using mathematical equations based on experimental data. Most of the early pressure drop calculation was based on these correlations because of thier direct applicability and fair accuracy to the data range used in the model generation. In this study, the empirical correlation for pressure drop estimation in multiphase flow in vertical wells are reviewed and evaluated with consideration of its required dimensions, performance, limitation and range of applicability.

Duns & Ros Correlation (1963): This empirical correlation is resulted from laboratory experiments with some modification and adjustments in the correlation by using actual field data. Duns & Ros correlation are in terms of a dimensionless gas velocity number, diameter number, liquid velocity number and a dimensionless mathematical expression. The acceleration gradient is neglected in the methods. Although this method is developed to calculate the pressure drop with dry oil/gas mixtures, it can also be used with wet oil/gas mixtures in some cases.

Hagedorn & Brown Correlation (1965): Hagedorn & Brown correlation is one of the most common correlations used in the industry. Hagedorn & Brown correlation has developed using an experimental study of pressure gradients occurring during continuous two-phase flow in small diameter vertical conduits, a 1500 ft vertical wellbore and considering 5 different fluids types in the experiment which is water and four types of oil. This correlation involve only dimensionless groups of variables and it can be applied over a much wider range of conditions comparing to other correlations.

Orkiszewski Correlation (1967): This correlation developed an equation for two-phase pressure drops in flowing and gas-lift production wells over a wide range of well conditions with range of a precision about 10 percent. The method is an extension of the work done by Griffith and Wallis (1961). The correlation is valid for several flow regimes such as; bubble flow, slug flow, transition flow and annular-mist flow. Orkiszewski proved his assumption by comparing the measured pressure drop results of 184 wells to the calculated ones. The parameter considered in his equation for the pressure drop is the effect by the energy lost by friction, the change in potential energy and the change in kinetic energy. The results obtained by these methods still applicable for wide range of well conditions (e.g. heavy oil). But, there are some well conditions that have not been evaluated (e.g., flow in the casing annulus and in the mist flow regime).

Beggs & Brill Correlation (1973): The Beggs and Brill method was developed to predict the pressure drop for horizontal, inclined and vertical flow. It also took into account the several flow regimes in the multiphase flow. Therefore, Beggs & Brill (1973) correlation is most widely used. In their experiment, they used 90 ft long acrylic pipes to produce data. Fluids used were air and water and 584 tests were conducted. Gas rate, liquid rate and average system pressure was varied. Pipes of 1 and 1.5 inch diameter were used. The parameters used are gas flow rate, Liquid flow rate, pipe diameter, inclination angel, liquid holdup, pressure gradient and horizontal flow regime. This correlation has been developed so it can be used predict the liquid holdup and pressure drop.

Gray Correlation (1978): The Gray correlation was developed by H.E. Gray (Gray, 1978), specifically for wet gas wells. Although this correlation was developed for wet gas vertical flow, but it can also be used in multiphase vertical and inclined flow. In his correlation Flow is treated as single phase, and dropped out water or condensate is assumed to adhere to the pipe wall. The parameters considered in this method are the phase velocity, tube size gas condensate ratio and water ratio. The pressure difference due to friction is calculated using the Fanning friction pressure loss equation.

Mukherjee & Brill Correlation (1985): Mukherjee & Brill Proposed a correlation for Pressure loss, Holdup and flow map. Their correlation was developed following a study of pressure drop behaviour in two-phase inclined flow. However, it can also be applied to vertical flow. Prior knowledge of the liquid holdup is needed to compute the pressure drop using Mukherjee & Brill (1985) correlation. The results obtained from their experiments were verified with Prudhoe Bay and North Sea data.

2.3 Mechanistic Models

Mechanistic models or known also as semi-empirical correlations deal with the physical phenomena of the multiphase flow. These kinds of models are developed by using mathematical modelling approach. A fundamental hypothesise in this type of models is the existence of various flow configurations or flow patterns, including stratified flow, slug flow, annular flow, bubble flow, churn flow and dispersed bubble flow. The first objective of this approach is, thus, to predict the existing flow pattern for a given system. Although most of the current presented mechanistic models have been developed under certain condition which limits their ability to be used in different range of data, these models are expected to be more reliable and general because they incorporate the mechanisms and the important parameters of the flow (Gomez et al. 2000).

Aziz et al. Model (1972): Aziz, Govier and Fogarasi (1972) have proposed a simple mechanistically based scheme for pressure drop calculation in wells producing oil and gas. The scheme was based on the identification of the flow pattern map. The mechanical energy equation was presented in the relationship between the pressure gradient, the flow rate, the fluid properties and the geometry of the flow duct. While the model proposed new equation for bubble and slug flow patterns, it recommended the old Dun & Ros equations for annular mist pattern. The new prediction method incorporates an empirical estimate of the distribution of the liquid phase between that flowing as a film on the wall and that entrained in the gas core. It employs separate momentum equations for the gas-liquid mixture in the core and for the total contents of the pipe. The model has presented 44 value of predicted pressure drop with an absolute error almost equal to the Orkiszewski correlation. However, the uncertainties and lack of some filed data made it difficult to develop a fully mechanistically, reliable based computation method.

Ansari et al. Model (1994): This mechanistic model is developed for upward two-phase flow in wellbores. This model was developed as part of the Tulsa University Fluid Flow Projects (TUFFP) research program. The model predict the existence of four flow patterns which are; bubble flow, slug flow, churn flow and annular flow. The model was evaluated by using the TUFFP well databank that is composed of 1775 well cases, with 371 of them from Prudhoe Bay data. Ansari et al (1994) claim that the overall performance of the comprehensive model is superior to all other methods considered with an exception of Hagedorn & Brown empirical correlation due to its extensive data used in its development and modifications made to the correlation.

2.4 Artificial Neural Networks

An artificial neural networks is a structure (network) composed of a number of interconnected units (artificial neurons). Each unit has an input/output (I/O) characteristic and implements a local computation or function (Jahanandish & Jalalifar, 2011). It has been only a few years since neural networks first gained popularity. In the past two to three years, banks, credit card a companies, manufacturing companies, high tech companies and many more institutions have adopted neural nets to help them in their day-to-day operation. Within the past few years, several software companies have surfaced that work solely on neural net products. Most researchers believe that artificial neural networks may be able to produce what rule based artificial intelligence (expert systems) have promised for so long but failed to deliver.

The use of Artificial Neural Networks (ANNs) in petroleum industry can be tracked several years ago. Since the literature have many industry problems solved by several authors using ANNs models. ANNs have been used in several area of oil and gas industry such as; permeability prediction, well testing, enhanced oil recovery, PVT properties prediction, improvement of gas well production, prediction & optimization of well performance, integrated reservoir characterization and portfolio management. (Ayoub, 2004).

Experience showed that empirical correlations and mechanistic models failed to provide a satisfactory and reliable tool for estimating pressure drop in multiphase flowing wells. Large errors are usually associated with these models and correlations (Takacs, 2001). Artificial neural networks gained wide popularity in solving difficult and complex problems, especially in petroleum engineering (Mohaghegh and Ameri, 1995).

Ayoub Model (2004): Ayoub presented an Artificial Neural Networks (ANNs) model for prediction of the bottom-hole flowing pressure and consequently the pressure drop in vertical multiphase flow. The model was developed and tested using field data covering a wide range of variables. A total of 206 field data sets collected from Middle East fields; were used to develop the ANN model. These data sets were divided into training, cross validation and testing sets in the ratio of 3:1:1. The testing subset of data, which were not seen by the ANN model during the training phase, was used to test the prediction accuracy of the model. Trend analysis of the model showed that the model correctly predicted the expected effects of the independent variables on bottomhole flowing pressure. This indicated that the model simulates the actual physical process. Although, the results showed that his model significantly outperforms all existing methods and provides predictions with higher accuracy. The author warned that the new developed model can be used only within the range of used data. Caution should be taken beyond the range of used input variables. Ayoub (2004) model demonostrate the power of artificial neural networks model in solving complicated engineering problems.

Chapter 3

RESEARCH METHODOLOGY

3.1 Overview

There are many approaches that can be used in order to solve engineering problems. These approaches can be classified as:

1. Exact or rigorous approach.
2. Modelling approach.
3. Mechanistic approach.
4. Experimental approach.

In this project, GMDH approach is classified as “modelling approach”. To the best of my knowledge it has not been used before in this kind of estimation. GMDH polynomial neural networks are being used to construct a mathematical model that can estimate the pressure drop in vertical wells. This mathematical model is built and developed as an attempt to replace the previously developed rigorous correlations either empirical correlations or mechanistic models. This model consists of a very simple approach of predicting the pressure drop with high accuracy and minimum usage of parameters.

The outcomes of these models have been compared against the measured one. An optimization study also used the trend analysis that confirmed the physical possibility of the proposed model. Figure (3.1) is illustrates the sequence of research procedure.

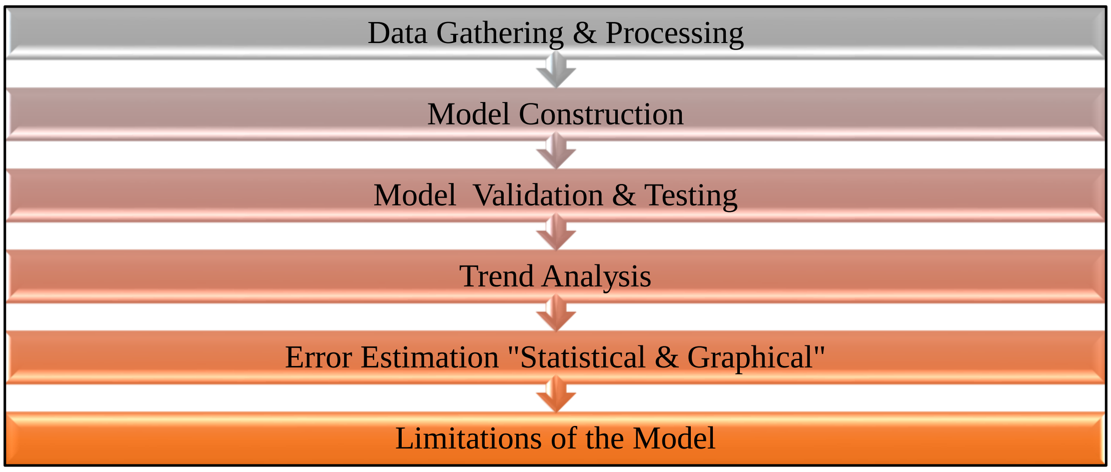
Data Gathering & Processing
Model Construction
Model Validation & Testing
Trend Analysis
Error Estimation "Statistical & Graphical"
Limitations of the Model

Figure 3.1 Modelling Construction Process

3.2 Data Gathering & Processing

The most important and critical step in the project is the data gathering which has the main impact on generating a successful model. During the data gathering and collection, it has been considered the quantity and quality of the collected data to ensure sufficient information that help to build the model.

When it comes to estimation the pressure drop in multiphase vertical wells, there are so many parameters known to be contribute to it. However, not all these parameters might be significantly contributed to the final output. Besides that, some of these parameters cannot be available in the data collection process due to some technical problems. Although this insufficiency in the data can reduce the accuracy of the model, it also might not have significant effects as it will be discussed later. Additionally, some of these input parameters were removed from the final data selection due to their low ranges.

A total number of 260 data sets had been used in this project in order to construct the mathematical model. The Input variables have been selected based on the most commonly used empirical correlations and mechanistic models used by the industry. These input variables are oil rate, water rate, gas rate, diameter of the pipe, length of pipe ”depth”, wellhead pressure, surface temperature and oil gravity “API”. Table (3.1) shows the statistical analysis of the used data in this project.

Table 3.1 Statistical Analysis of the Used Data

Flow Parameter	Min	Max	Average	STD
Bottomhole Pressure, (psi)	1019.79	3124	2234	476.971
Oil Rate, (bbl/D)	45.2	19618	5068.5	4838.1
Water Rate, (bbl/D)	0	7900	1757.3	2309
Gas Rate, (Mscf/D)	0	13562.2	2563.6	3047.57
Depth, (ft)	2726.4	8070.87	5830	1040
Tubing Diameter, (in)	2	4	3.75	0.33
Surface Temperature, (degreeF)	70	160	113.55	27.44
Wellhead Pressure, (Psi)	5	800	249.5	159.17
Oil Gravity, (API)	12.4	37	31.1	5.675

3.2.1 Partitioning

Partitioning the data is the process of dividing the data into three different sets: training sets, validation sets, and test sets. By definition, the training set is used to build and develop the model, the validation set is used to ensure the optimum generalization of the developed model and the test set, which is not be seen by the network during training, is used to examine the final performance of the model. Although different partitioning ratios were tested (2:1:1, 3:1:1, and 4:1:1), the author has choosen the 2:1:1 ratio because it’s more popular and frequently used by researchers (Ayoub, 2004).

According to the chosen partition ration 130 data set reserved for training the model while 65 data sets were utilized for validation purposes. The last 65 data set had been kept aside for testing the new model performance. Needless to mention that this testing set was never seen by the network during training and validation.

3.3 Building GMDH Model

The process of building the GMDH model started with selecting the input parameters which has been discussed earlier. Free software was being used for this purpose (Jekabsons, 2011). This source code was tested with MATLAB version 8.1 (R2013a). Despite the software allows great flexibility in selecting the model parameters, it also provides ample interference. Although, all of the input parameters had been used in generating the model, just a few are used in the final equation to estimate the pressure drop.

3.4 Software Used

In this Project, MATLAB software (version R2013a), [MATLAB], environment was utilized due to flexible programming and graphs visualization. This software provides a good way to monitor the performance of the three data sets (training, validation and testing data) simultaneously which ease the optimization process and the sensitivity analysis.

A MATLAB code was developed by Jekabsons (2011). His code represents a simple implementation of Group Method of Data Handling (GMDH) for building Polynomial Neural Networks. The algorithm uses the training data to build the network in a layer by layer arrangement. The connectivity and number of layers of the network is controlled by an evaluation criterion. The code algorithm gives the user either to use measuring performance in an additional validation data explicitly taking network's complexity into account such as Corrected Akaike's Information Criterion or Minimum Description Length. The code algorithm also includes other parameters such as, max number of inputs for individual neurons, degree of polynomials in the neurons, whether to allow the neurons to have inputs not only from the immediately preceding layer but also from the original input variables, number of neurons in a layer, whether to decrease the number of neurons in each subsequent layer.

3.5 Trend analysis

A trend analysis is carried out for the proposed GMDH model to check whether the model is physically possible. For this purpose, synthetic sets will be prepared where in each set only one input parameter will be changed while other parameters will be kept constant. To test the developed model, the well-known effects of different input parameters such as; oil rate, gas rate, water rate, oil gravity "API", pipe length (depth) will be studied. These created trends from the developed model will be expected to match with the inflow performance relationship (IPR) and typical pressure trends in well testing for multiphase flow in vertical pipes.

3.6 Statistical Error Analysis

This type of error analysis has been used to check the accuracy of the proposed models and also the other investigated models. The statistical parameters used in this project is average absolute percentage relative error, average percentage relative error, maximum absolute percentage error, minimum absolute percentage error, root mean square error, coefficient of determination and the standard deviation of error. Equations for those parameters are given in the appendices.

3.7 Graphical Error Analysis

Graphical tools aid in visualization the performance and the accuracy of the generated model. Three graphical analysis tools will be used; those are crossplots and error distribution.

3.7.1 Cross-plots

Cross plots were used to compare the performance of all the selected methods. A 45° straight line between the calculated pressure drop values versus measured pressure drop values is plotted which represent a perfect correlation line. When the values go closer to the line, it will indicate better results between the measured and the estimated values.

3.7.2 Error Distribution

Error distribution shows the error sharing histograms for the proposed GMDH model (both training, validation and testing data sets). Normal distribution curves had been fitted to each one of them. The errors are said to be normally distributed with a mean around 0% and the standard deviation equal to 1.0. The normal distribution is often used to describe, at least roughly, any variable that tends to cluster around the mean. In our case it was used to describe the error tendency around the mean, (which is alternatively known as a normal or Gaussian distribution).

3.8 Limitations of the Model

The proposed GMDH model may be limited due to two main reasons. The first one in the limitation of the collected data; as it has been discussed earlier and definitely this will have direct impact on the results accuracy. The second one is the range of each input variable and the availability of that input parameter. Each parameter has specific range that works well, however, the accuracy may be lightly or severely affected if the parameters are not in the proposed range. Therefore, care must be taken if obtained results applied for data range beyond that used in generating the model.

Chapter 4

RESULTS AND DISCUSSION

4.1 Development of the GMDH model

4.1.1 Introduction

Group Method of Data Handling approach is a set of several algorithms for different problems solution. This inductive approach is based on sorting-out of gradually complicated models and selection of the best solution by minimum of external criterion characteristic. Not only polynomials but also non-linear, probabilistic functions or cauterizations are used as basic models. Polynomial GMDH technique is offering a sound representation of input regime to output through the application of so called “regularity criterion”. Usually this one will be average absolute percentage error. It is implemented to reduce the error between the actual and estimated target in each layer. A threshold level is applied before each layer is added since addition of a new layer and neurons depends on this threshold level.

As it has been described earlier, free software has been used to construct the GMDH model. The constructed model consists of two layers. 84 neurons were tried in the first layer, while only three neurons were included at the end of the trial. Only one neuron had been included (by default) for the second layer, which was the pressure drop target. However four input parameters had shown pronounced effect on the final pressure drop estimate, which were; oil rate, length of the pipe “depth”, oil gravity and water rate. The selection of these three inputs had been conducted automatically without any interference from the user. They were selected based on their mapping influence inside the data set on the pressure drop values.

This topology was achieved after a series of optimization processes by monitoring the performance of the network until the best network structure was accomplished. Figure (4.1) shows the schematic diagram of the proposed GMDH topology. The final output layer “pressure drop” is being formed from five variables from the input layer which are, oil rate, depth “pipe length”, water rate, oil gravity and gas rate. Whereas, the first three variables combined in one variable in the hidden layer and then with the other two variables used to build the output layer.

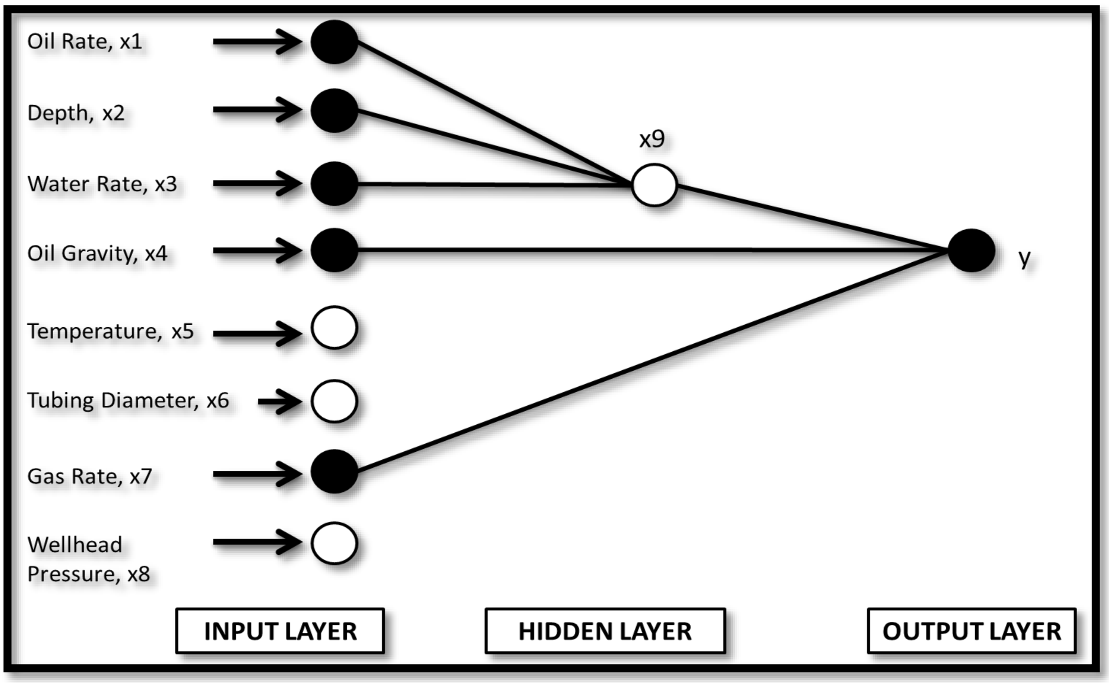

Figure 4.1 Schematic Diagram Of The Proposed GMDH Model Topology

4.1.2 Summary of the Model’s Equation

As described in the previous section the model consists of two layers as follows:

Number of layers: 2

Layer #1

Number of neurons: 1

```
x9 = -2301.44790006229 -2.60953702100053*x3 +1.84520074544553*x2 -
0.076677907649031*x1 +0.00077962057680922*x2*x3 -4.41529120121688e-
05*x1*x3 +9.86884816305755e-05*x1*x2 +3.67549182129376e-05*x3*x3 -
0.000316306612876803*x2*x2 -2.64450254505861e-05*x1*x1
+2.31597897389801e-09*x1*x2*x3 -7.00811416239125e-09*x2*x3*x3 -
5.4336357577612e-08*x2*x2*x3 +1.94571672043658e-09*x1*x3*x3 -
1.54237211114233e-08*x1*x2*x2 +1.73342685500666e-09*x1*x1*x3
+5.42470488360359e-09*x1*x1*x2 +1.35589606927285e-10*x3*x3*x3
+2.09789302765354e-08*x2*x2*x2 -1.83874781fig894772e-10*x1*x1*x1
```

Layer #2

Number of neurons: 1

```
y = 3415.95672012025 -0.605615045600824*x9 -1.04616288868911*x7 -
357.669124247228*x4 +0.000262141590712973*x7*x9
+0.166695464057425*x4*x9 -0.00880002548519891*x4*x7 -
0.000230414682054698*x9*x9 +0.000212921162434079*x7*x7
+10.9074140098337*x4*x4 -2.26040538469408e-06*x4*x7*x9 -
8.03293262244177e-08*x7*x9*x9 +1.26383450874532e-08*x7*x7*x9
+0.000159284034091951*x4*x9*x9 -7.62764844346145e-06*x4*x7*x7 -
0.0125022347939832*x4*x4*x9 +0.00120868087696399*x4*x4*x7 -
8.18463853556531e-07*x9*x9*x9 +1.67003375432709e-09*x7*x7*x7
+0.0651784907669141*x4*x4*x4
```

Where;

x1 = oil rate, bbl/d

x2 = length of the pipe, ft

x3 = water rate, bbl/d

x4 = oil grvity, API

x7= gas rate, scf/d

y = simulated pressure drop by GMDH Model

4.1.3 GMDH Model Optimization

In order to optimize the GMDH model, many factors have been taken into consideration. These factors are; whether to use measuring performance in an additional validation data explicitly taking network's complexity into account such as Corrected Akaike's Information Criterion or Minimum Description Length, max number of inputs for individual neurons, degree of polynomials in the neurons, whether to allow the neurons to have inputs not only from the immediately preceding layer but also from the original input variables, number of neurons in a layer, whether to decrease the number of neurons in each subsequent layer.

The effects of all the above mentioned factors have been studied and verified using the software. The selection of the best criteria to choose for the model was based on having the highest correlation coefficient for the testing and validation data sets.

4.2 Trend Analysis for the Proposed GMDH Mode

A trend analysis was carried out to check whether the developed model is physically correct or not. To test the developed model, the effects of gas rate, oil rate, water rate, and depth “pipe length” on pressure drop were determined and plotted on Figure (4.2) through Figure (4.6).

As expected, the developed model has achieved truthful trends that match the normal pressure trends. The pressure drop increases as the gas, water and oil increases as justified by the general energy equation. Same goes to the increase in pressure drop with depth. The increase in pressure drop when oil gravity in increased in simply justified by the specific gravity equation, where specific gravity is directionally proportional to pressure.

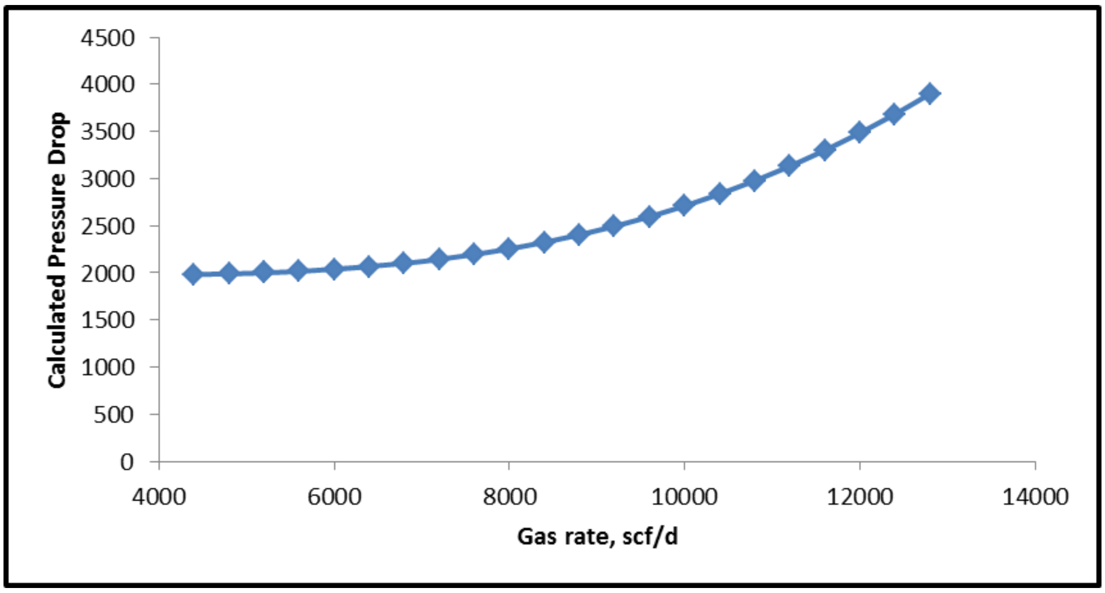

Figure 4.2 Effect of Gas Rate on Pressure Drop

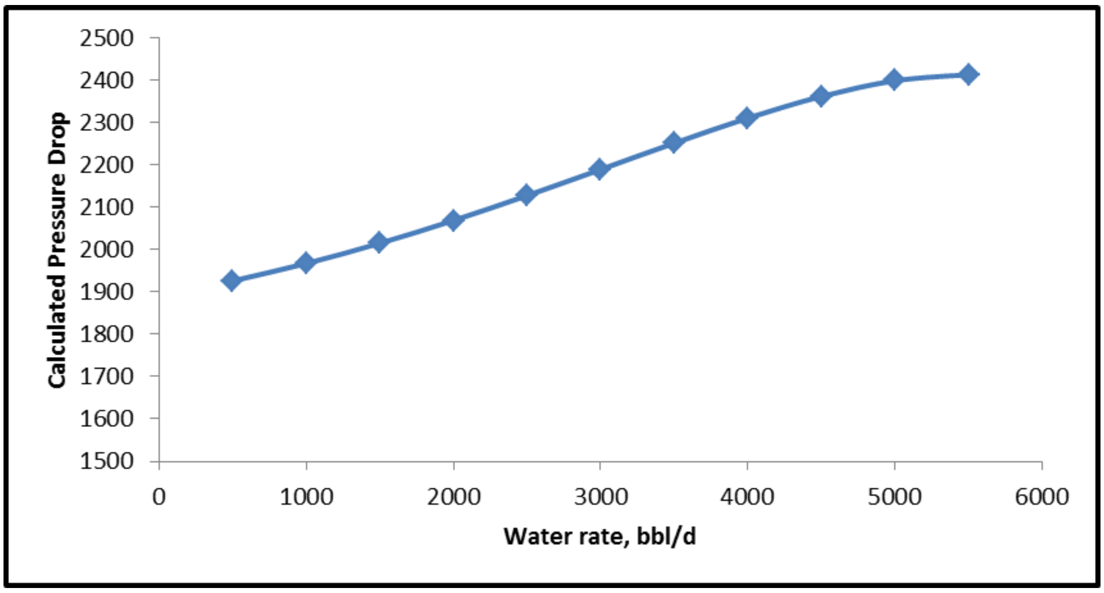

Figure 4.3 Effect of Water Rate on Pressure Drop

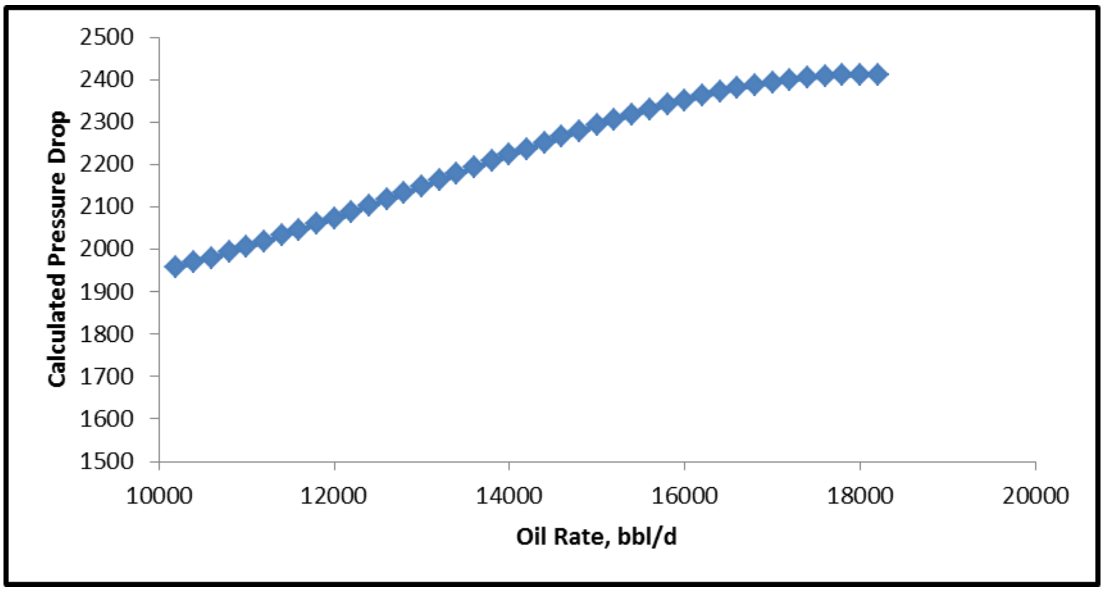

Figure 4.4 Effect of Oil Rate on Pressure Drop

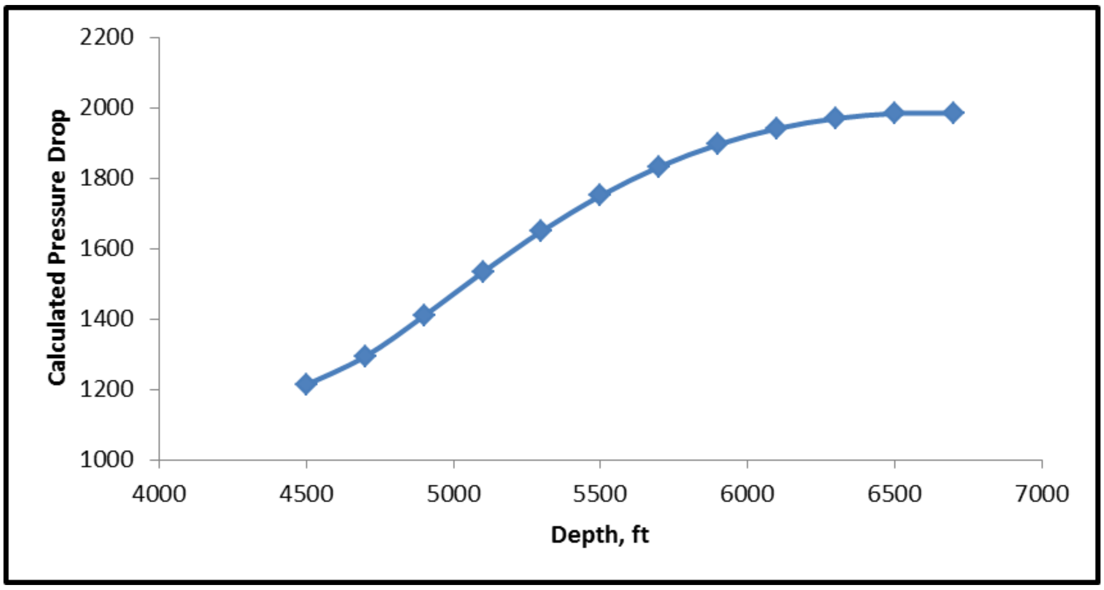

Figure 4.5 Effect of Depth "Pipe Length" on Pressure Drop

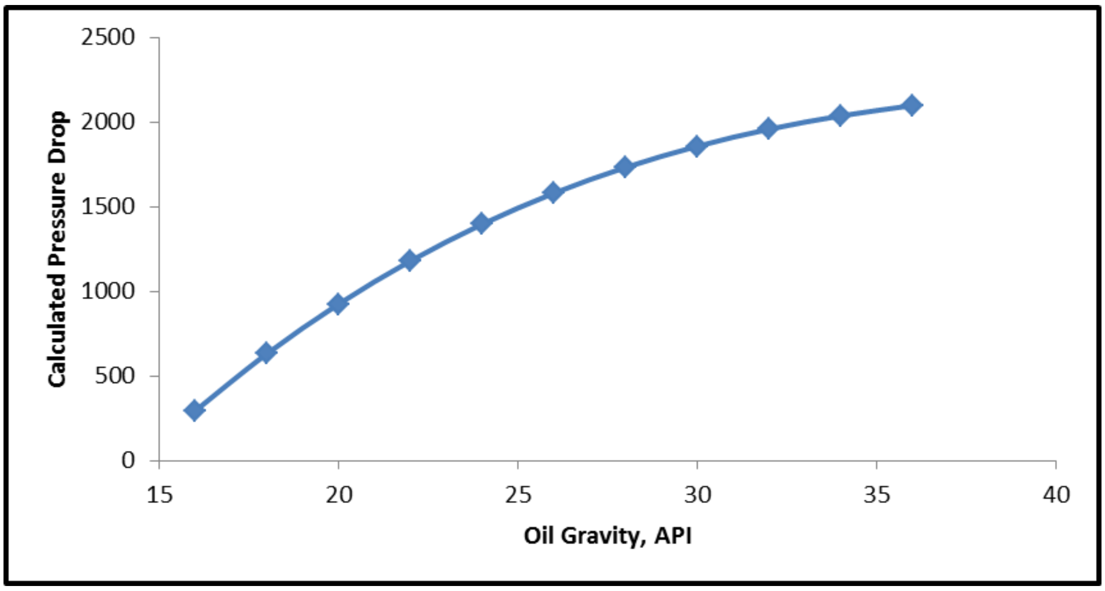

Figure 4.6 Effect of Oil Gravity on Pressure Drop

4.3 Statistical Error Analysis for the Proposed GMDH Model against Other Investigated Models

As mentioned in methodology chapter, this error analysis was used to check the accuracy of all investigated models. The statistical parameters used in this project are average absolute percentage relative error (AAPE), average percentage relative error (APE), maximum absolute percentage error (MaxAE), minimum absolute percentage error (MinAE), root mean square error (RMSE), coefficient of determination (R^2) and the standard deviation of error (STD).

Summary of statistical comparisons between all model's sets (training, validation, and testing) is presented in Table 4.1. And Table 4.2 summarizes these statistical parameters of the proposed GMDG model and the investigated models.

4.4 Graphical Error Analysis for the Proposed GMDH Model against Other Investigated Models

Graphical tools aid in visualization the performance and the accuracy of the generated model. Two graphical analysis tools used to check to model accuracy, heterogeneity and limitation. Those graphical error analysis are crossplots and error distribution.

Table 4.1 Statistical Analysis Results of the Proposed GMDH Model

Statistical Parameter	Training	Validation	Testing
AAPE	4.36119	7.156847	4.462313
APE	0.128042	1.450954	-0.38283
MaxAE	18.96302	29.13439	22.31491
MinAE	0.017754	0.164541	0.215909
RMSE	5.632366	9.624206	5.861574
R^2	0.933	0.7234	0.9233
STD	3.564207	6.434662	3.800766

Table 4.2 Statistical Analysis Results of the Proposed GMDH Model and Investigate Models

	AAPE	APE	MaxAE	MinAE	RMSE	R^2	STD
Aziz et al	18.1616	15.3085	60.5106	0.1688	25.7129	0.2044	18.2019
Hagedorn & Brown	11.9679	10.8906	25.6410	0.2806	13.7067	0.7888	6.6815
Gray	11.7019	10.0595	25.1312	0.0611	13.7572	0.7346	7.2337
Orkiszewski	11.5102	10.1379	28.1472	0.4975	13.4310	0.7758	6.9215
Mukhrejee & Brill	9.6868	5.5606	39.3635	0.4499	11.6977	0.8061	6.5575
Ansari et al	7.8973	4.9361	30.0916	0.0918	9.5092	0.8614	5.2970
Duns & Ros	7.6357	5.5159	24.2722	0.0475	9.4680	0.8362	5.5981
Beggs & Brill	6.5778	3.2094	24.9539	0.3135	8.1210	0.8710	4.7626
Ayoub	4.5295	-0.3216	18.2205	0.0150	6.1522	0.9052	4.1633
GMDH Model	4.4623	-0.3828	22.3149	0.2159	5.8616	0.9233	3.8008

4.4.1 Cross Plots of GMDH Model against Investigated Models

Figure (4.7) through Figure (4.9) present cross-plots of estimated pressure drop versus measured pressure drop for the proposed GMDH model data sets; Training, Validation and Testing. In these figures, the coefficient of determination obtained by the training set was (0.933), while the validation set obtained a value of (0.7234) and the testing set obtained a value of (0.9233). Moreover, Figure (4.10) through Figure (4.18) present cross-plots of estimated pressure drop versus measured pressure drop for other investigated models including the coefficient of determination for each model.

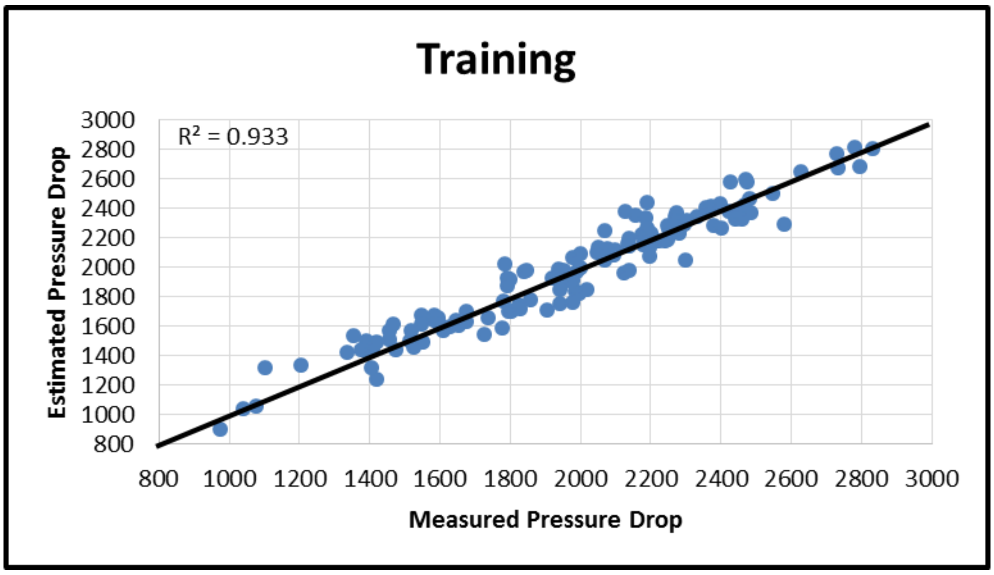

Figure 4.7 Cross plot of pressure drop for Training set (GMDH Model)

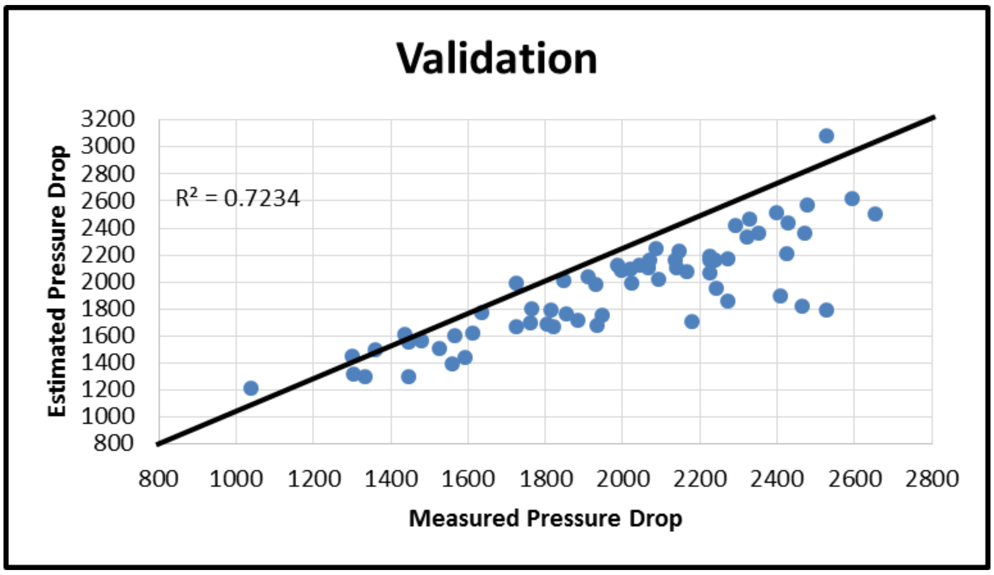

Figure 4.8 Cross plot of pressure drop for Validation set (GMDH Model)

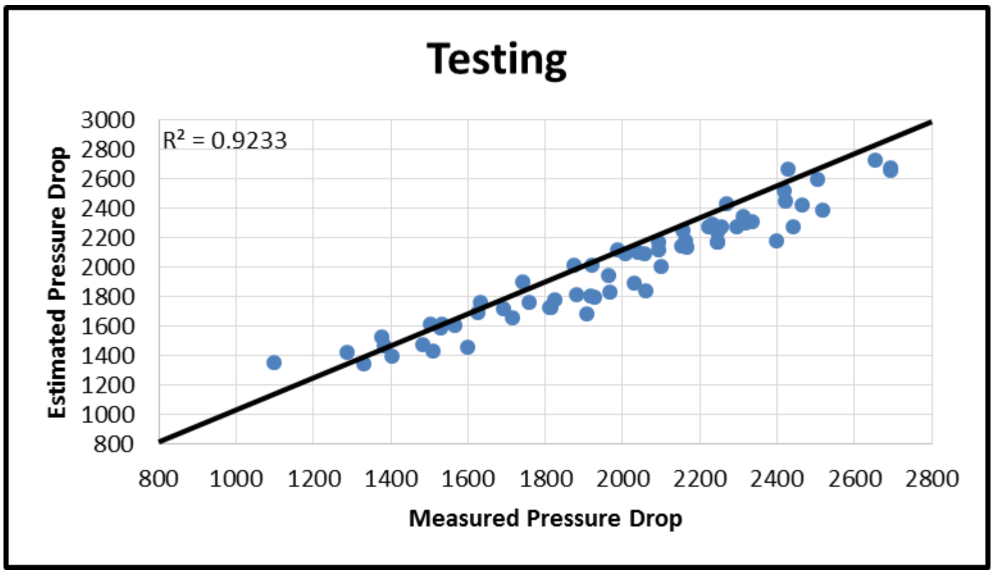

Figure 4.9 Cross plot of pressure drop for Testing set (GMDH Model)

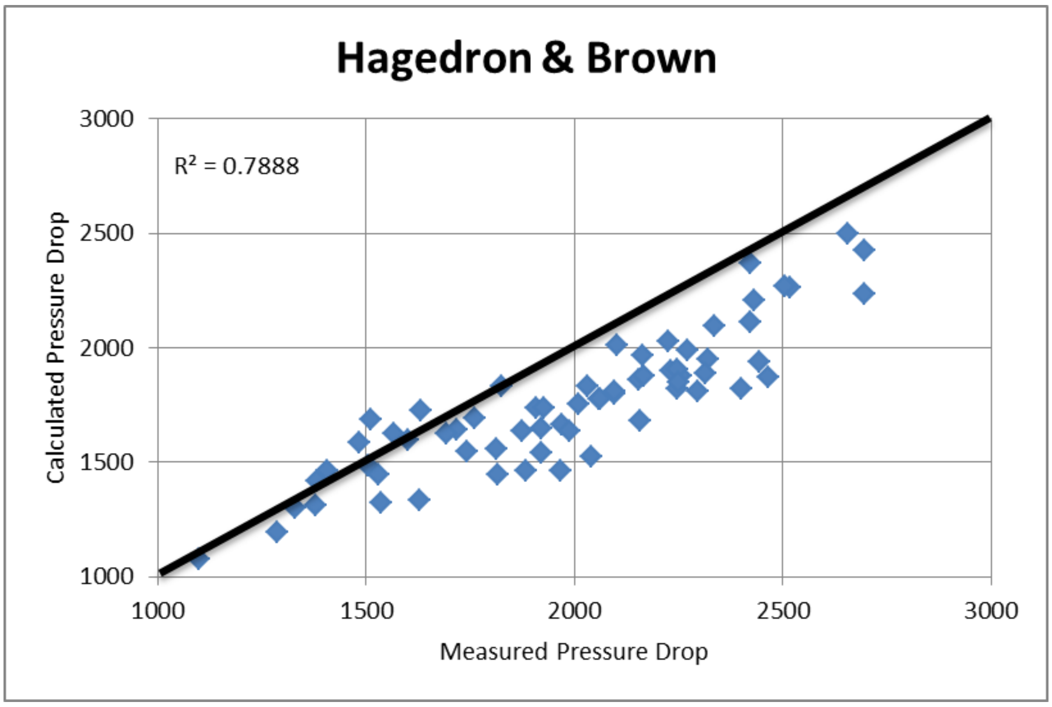

Figure 4.10 Cross plot of pressure drop for Hagedorn & Brown Correlation

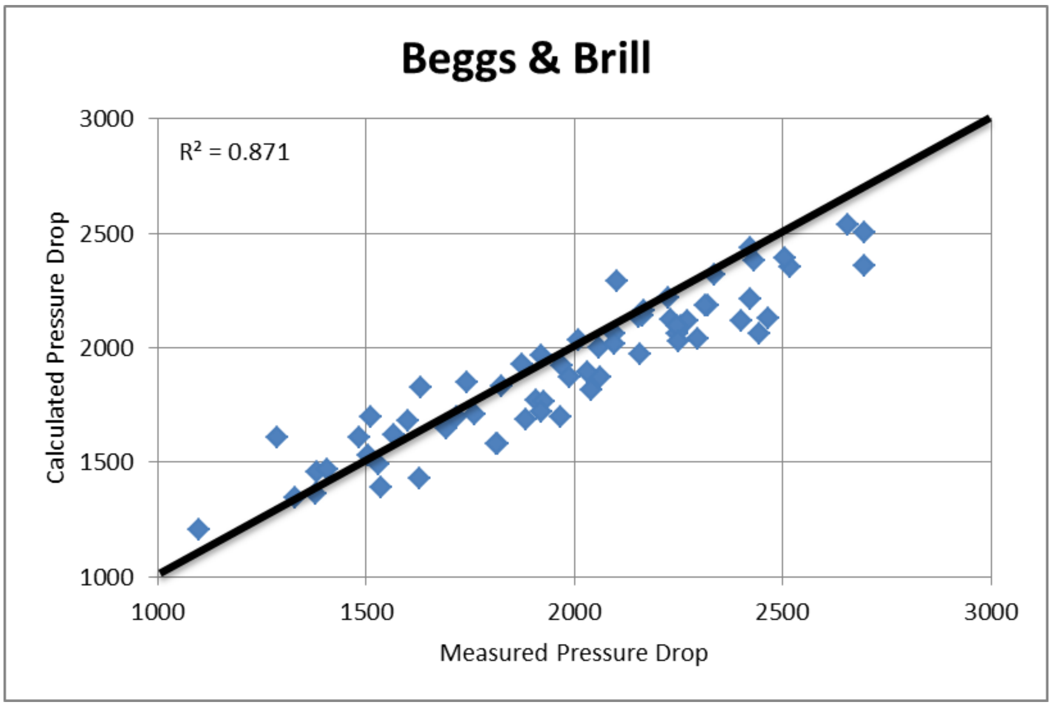

Figure 4.11 Cross plot of pressure drop for Beggs Brill Correlation

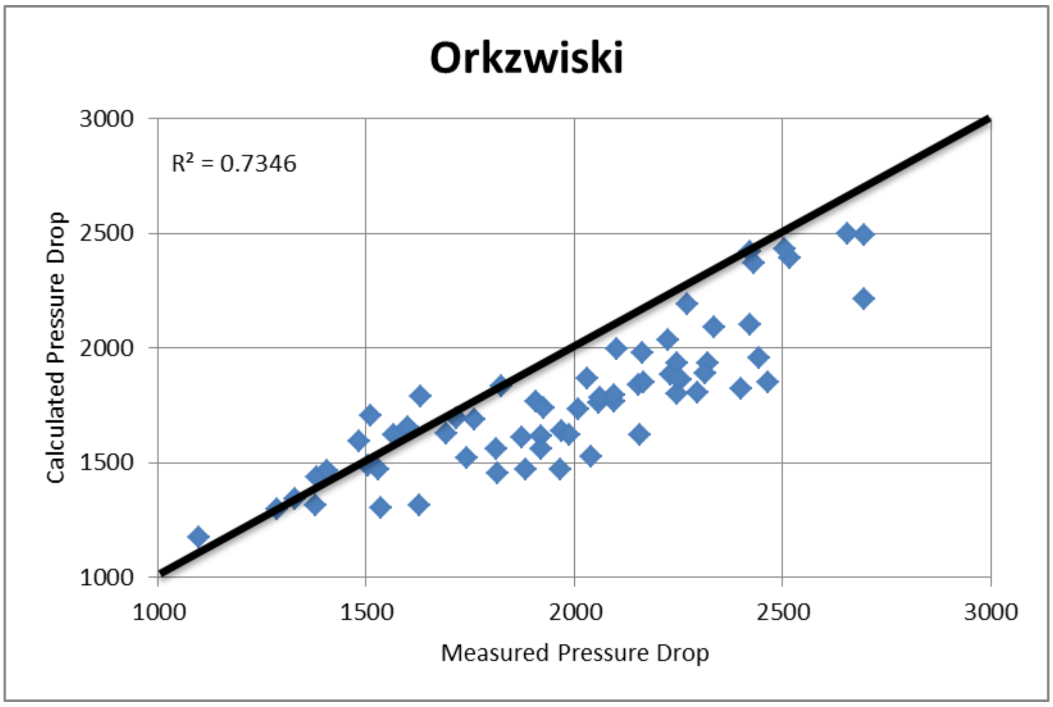

Figure 4.12 Cross Plot of Pressure Drop for Orkiszewski Correlation

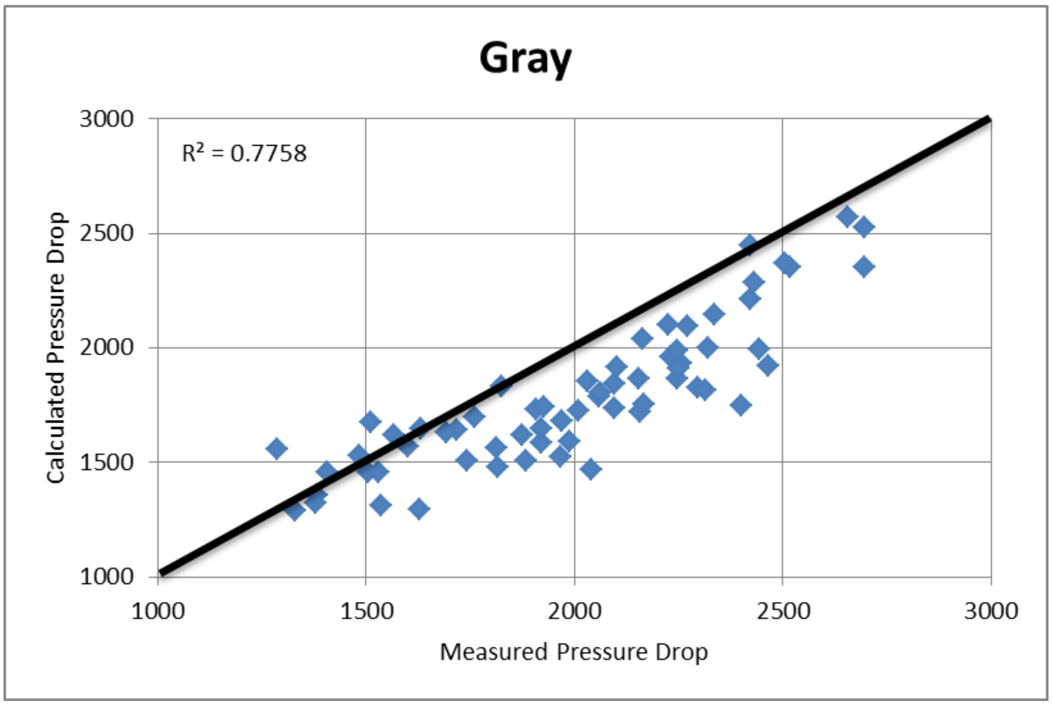

Figure 4.13 Cross plot of pressure drop for Gray Correlation

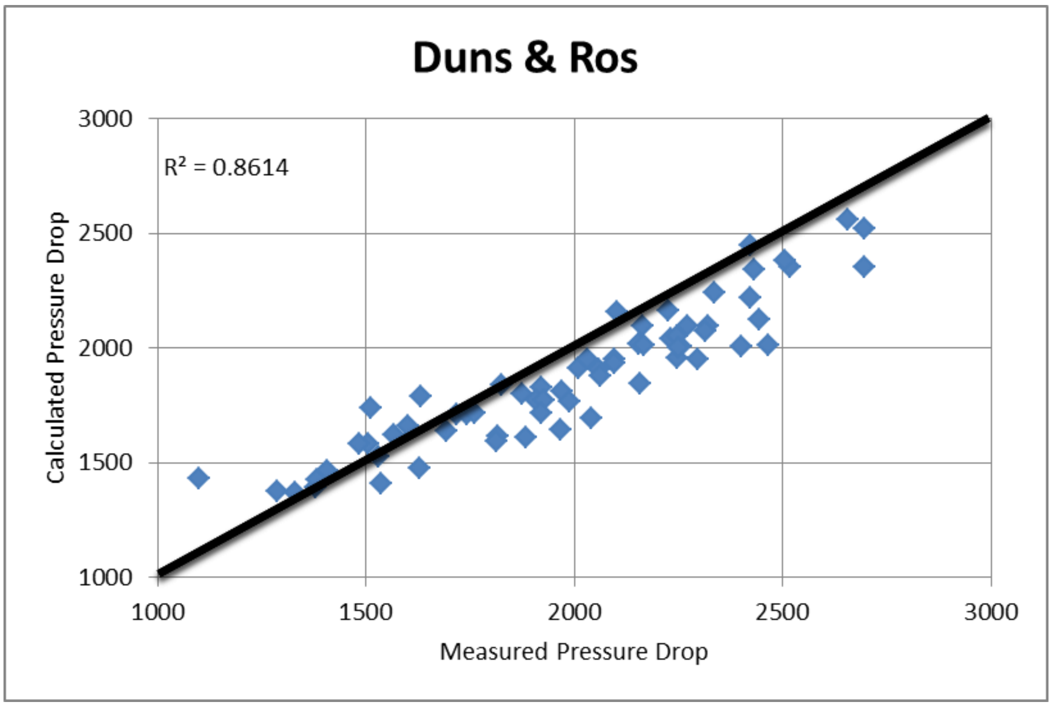

Figure 4.14 Cross plot of pressure drop for Duns & Ros Correlation

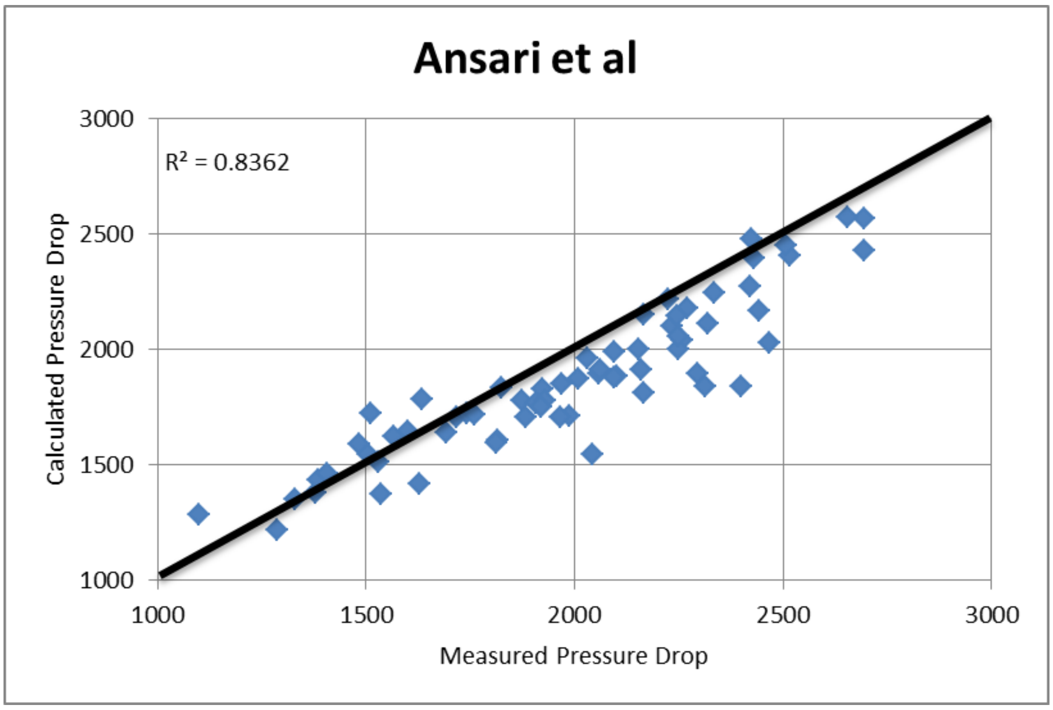

Figure 4.15 Cross plot of pressure drop for Ansari et al Model

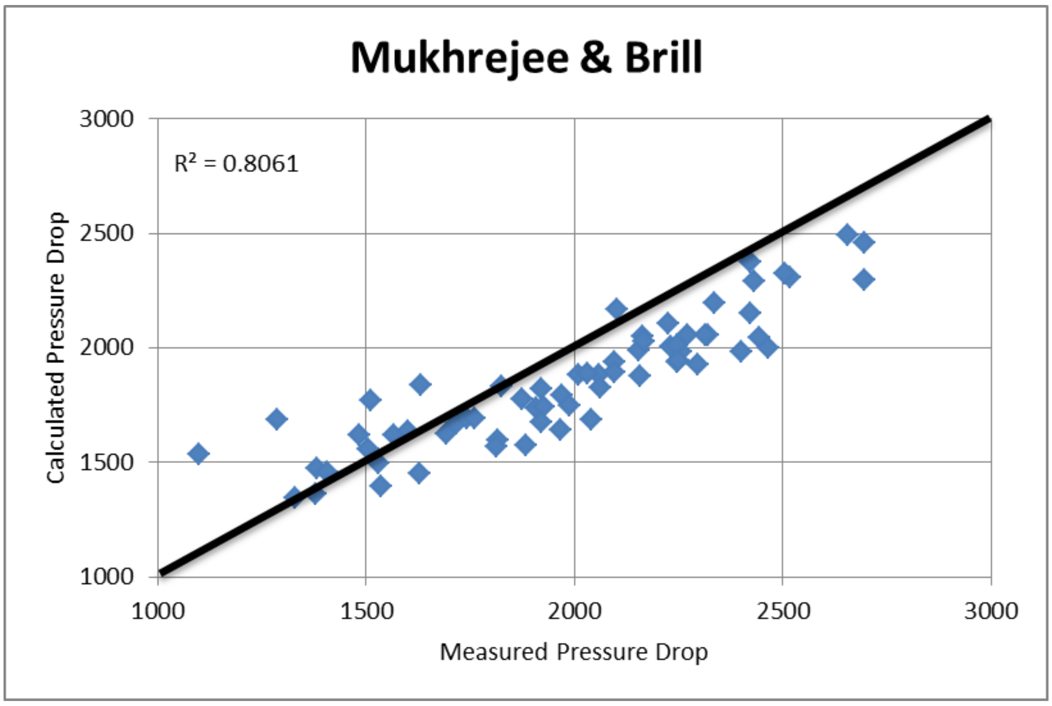

Figure 4.16 Cross plot of pressure drop for Mukherjee & Brill Correlation

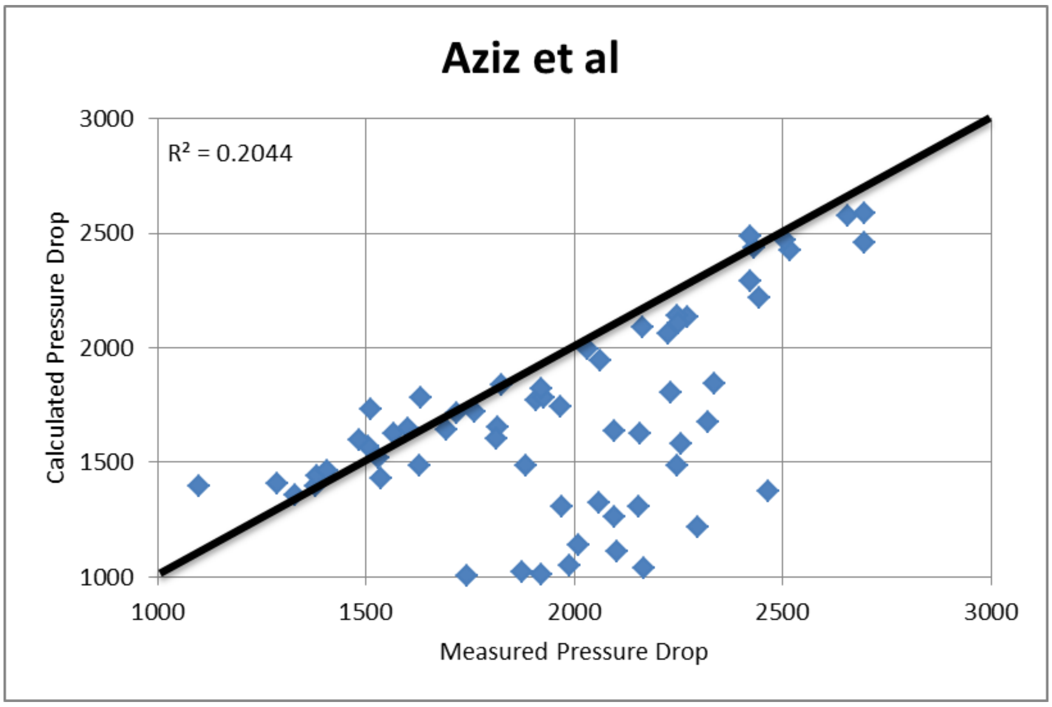

Figure 4.17 Cross plot of pressure drop for Aziz et al Model

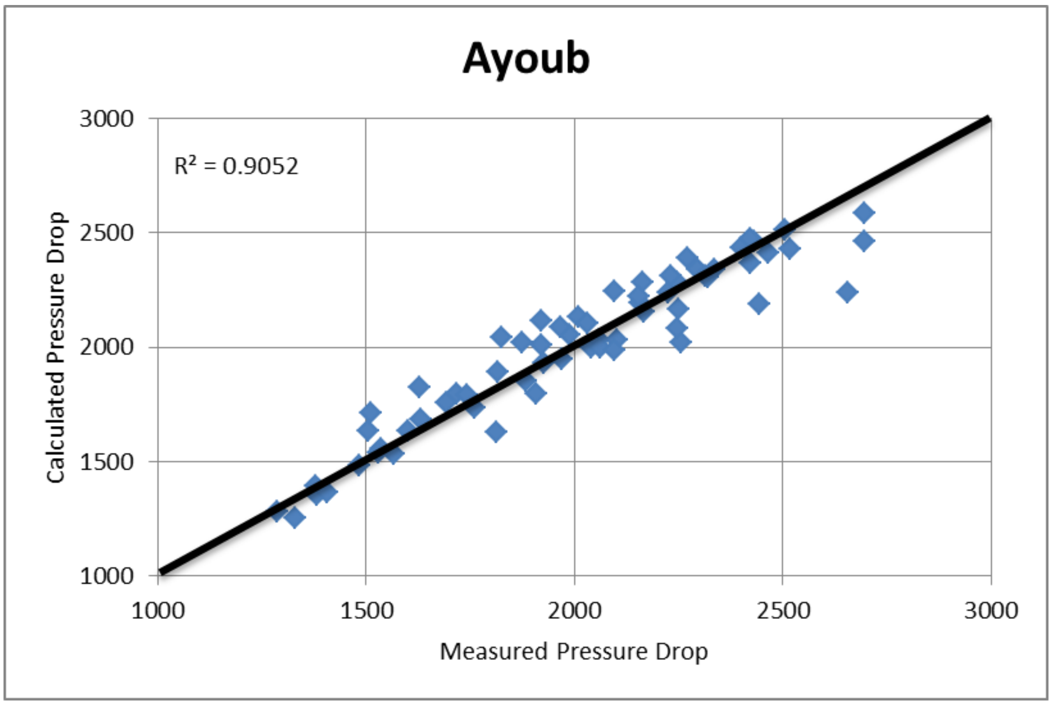

Figure 4.18 Cross plot of pressure drop for Ayoub Model

4.4.2 Error Distribution of GMDH Model against Investigated Models

Figure (4.19), Figure (4.20) and Figure (4.21) show the error distribution histograms for the GMDH model data sets, training, validation and testing sets. And, Figure (4.22) shows the error distribution histograms for the GMDH model and other investigated models.

Analyzing the GMDH model's error distribution histogram is quite important. It can give a clear idea about the model's performance for all data sets. According to the obtained results, the training set has normal distribution without any noticeable shifting towards the negative or positive side of the plot which indicates a good estimation, validation set has a slight shift towards the positive side of the plot which means that the pressure drop was slightly underestimated, and the testing set has also a slight shift towards the positive side of the plot which means that the pressure drop was slightly underestimated.

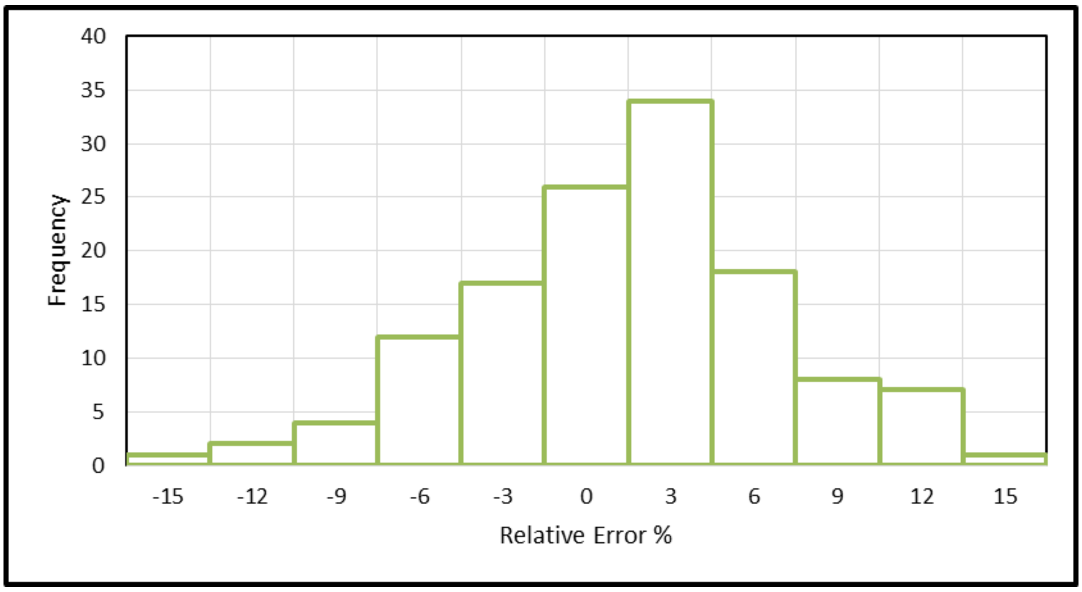

Figure 4.19 Error Distribution for Training Set

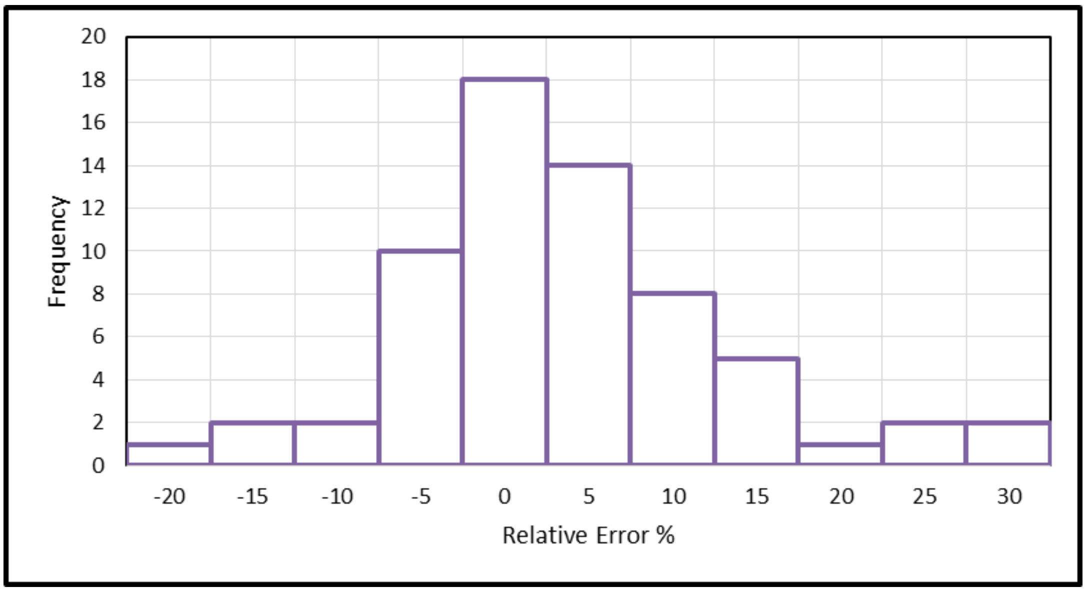

Figure 4.20 Error Distribution for Validation Set

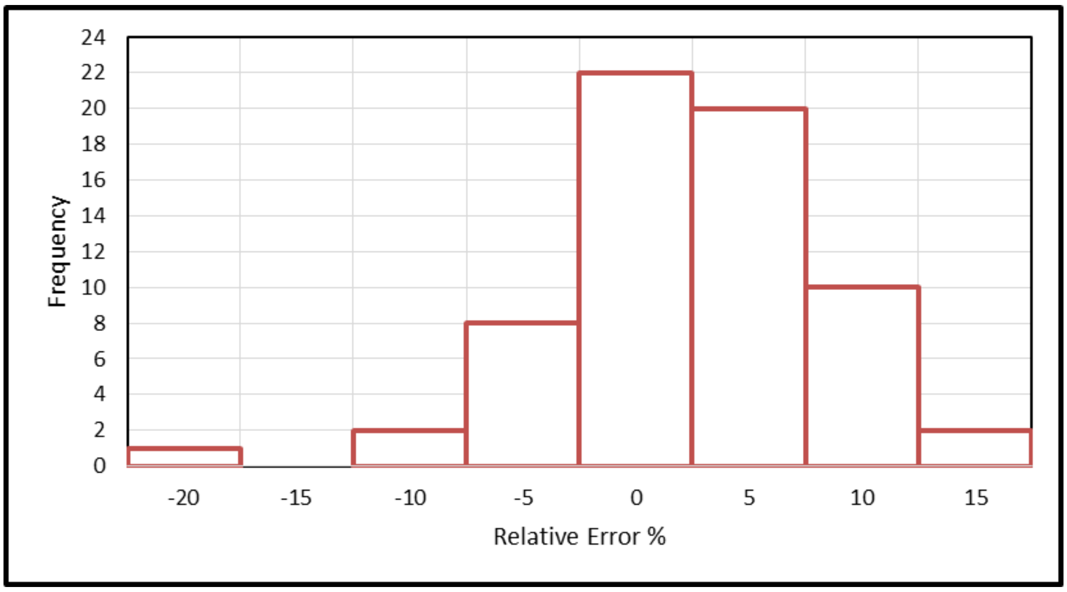

Figure 4.21 Error Distribution for Testing Set

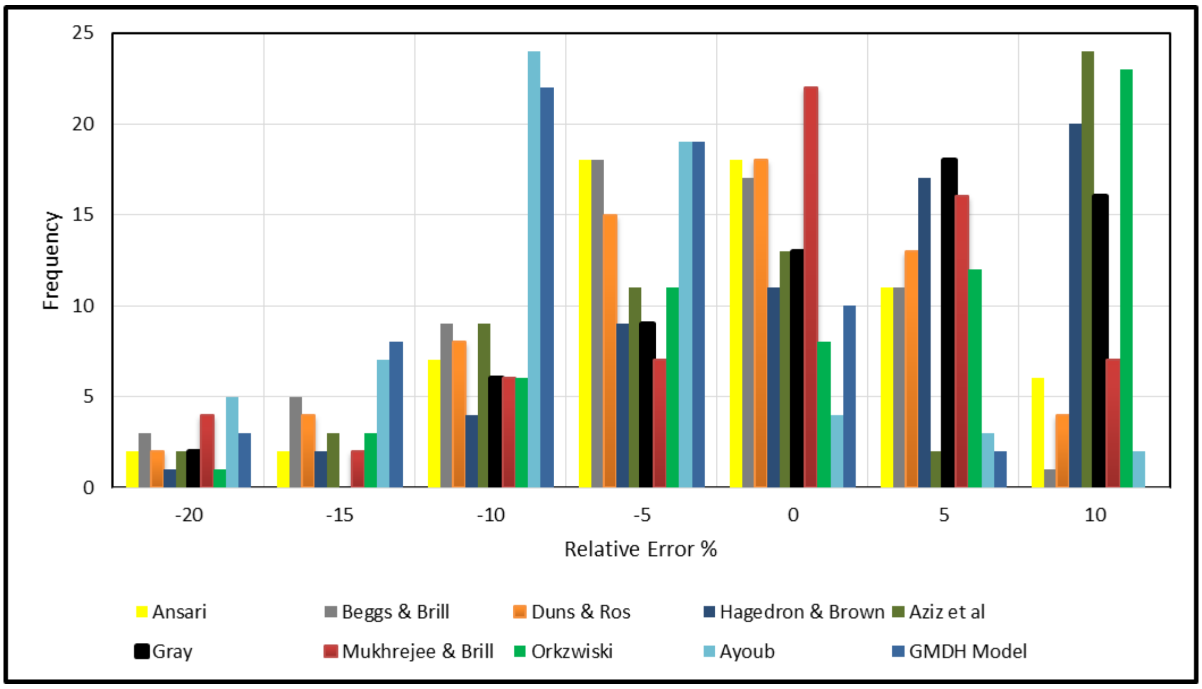

Figure 4.22 Error Distribution for All models including GMDH model

4.5 Discussion of the Results

Comparison between the performance of all investigated models plus the new proposed GMDH model has been showed earlier in Table (4.2). Figure (4.23) through Figure (4.25) indicate the performance of all investigated models. Aziz et al mechanistic model achieved the worst AAPE, RMSE and coefficient of determination among all investigated models.

The training set has achieved the best results statistically among the three sets of data. This is due to the fact that the training set has been used as the primary set for the model development. On the other hand, the validation set has achieved lower than the testing although the validation also used during the model development. This low performance of the validation maybe attributed to the nature of the validation set. The latter is introduced during the training process to control the performance of the training set whereas several failure cases may be occurred in order to obtain high accuracy and allow thee training set to learn effectively.

A close result can be extracted when root mean square errors (RMSE) of each model have been plotted against the standard deviation (STD) of errors, as presented in Figure (4.25). However, this time the best model will be located at the left lower corner, which indicated by the intersection of both lower values of RMSE and STD. Also, average absolute relative errors (AAPE) of each model has been plotted against the confident of determination (R^2), as presented in Figure (4.26). However, this time the best model will be located at the left upper corner, which indicated by the intersection of both low AAPE value with High R^2.

GMDH Model has always fallen in the best corner of the graph as compared against the other models. This indicates better-quality performance of GMDH model when compared to other tested models.

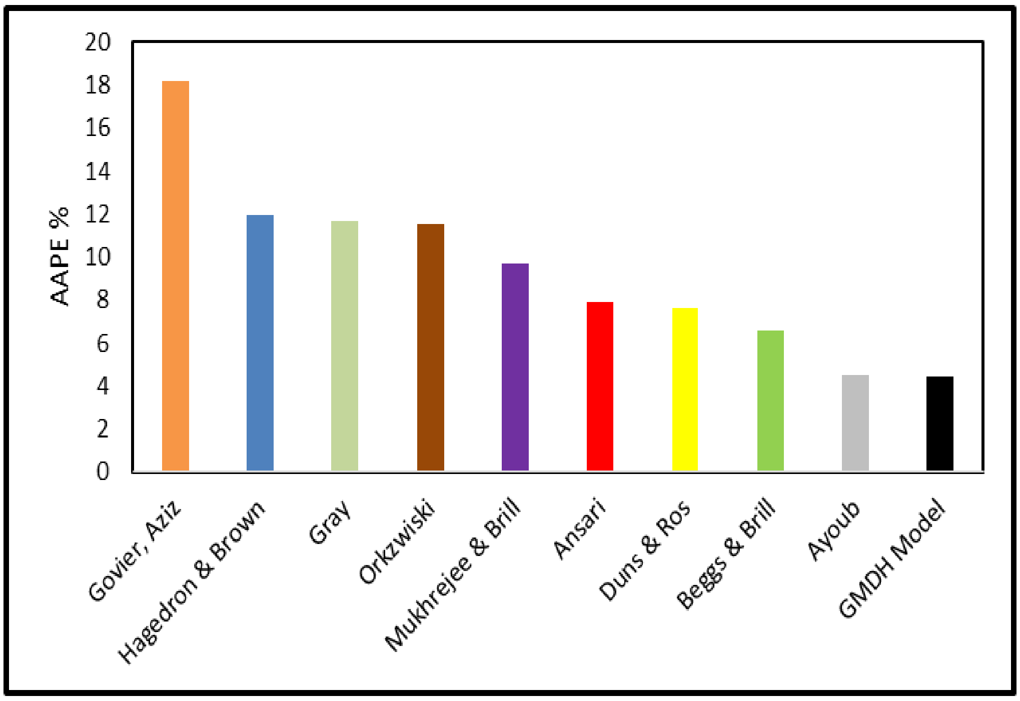

Figure 4.23 Average Absolute Relative Error for All Models

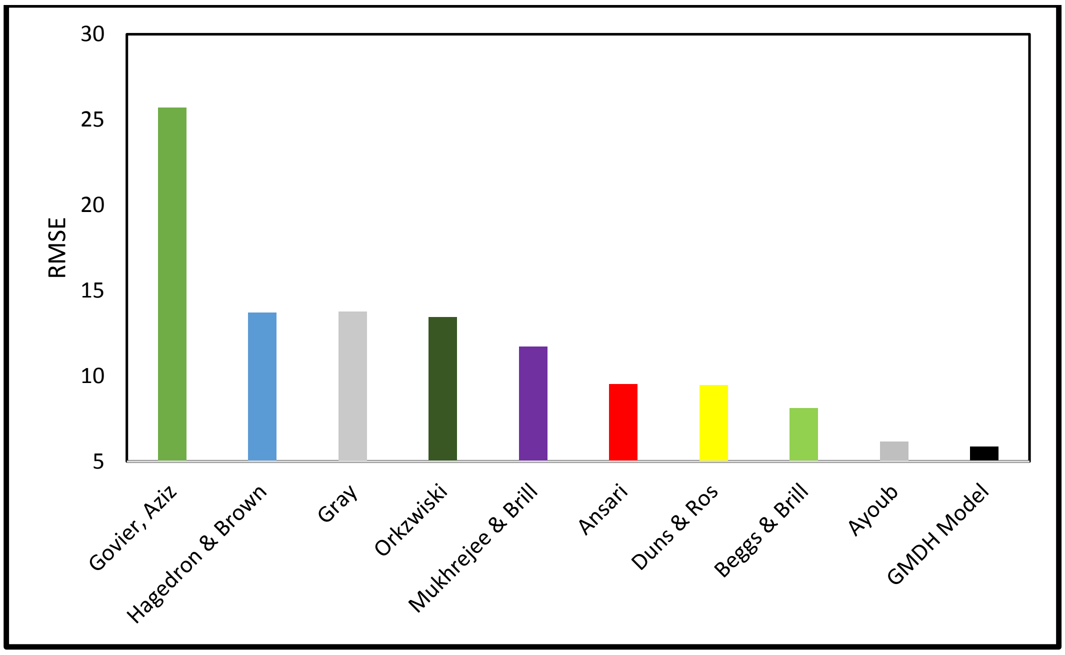

Figure 4.24 Root Mean Square Error for All Models

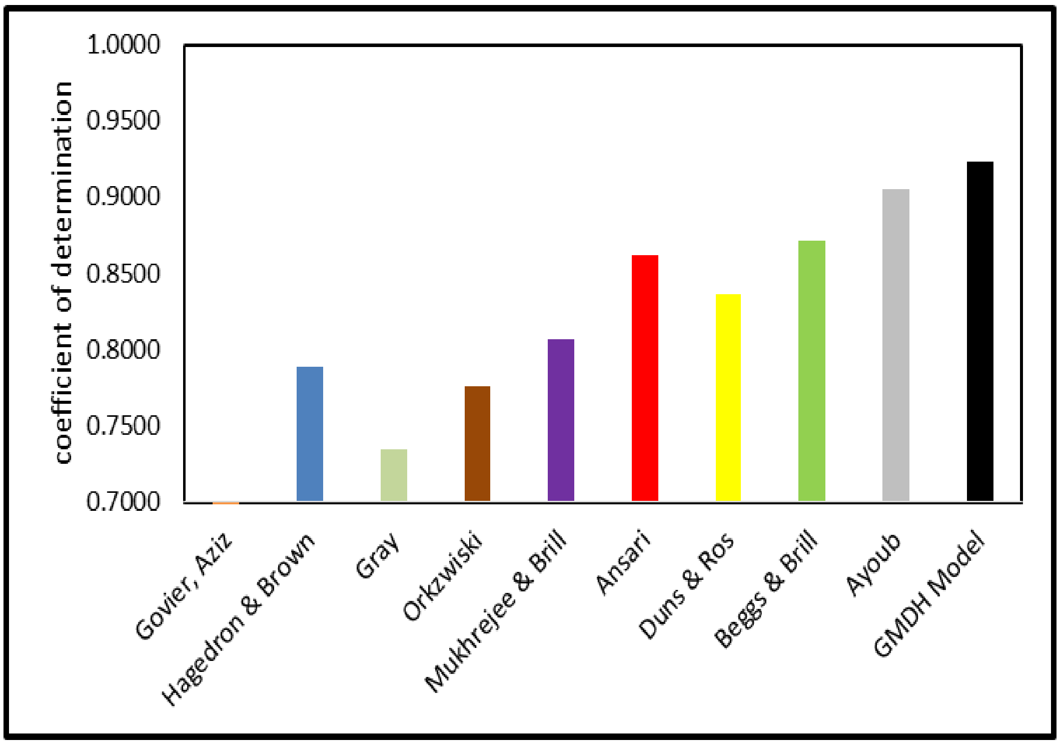

Figure 4.25 Coefficient of Determination for All Models

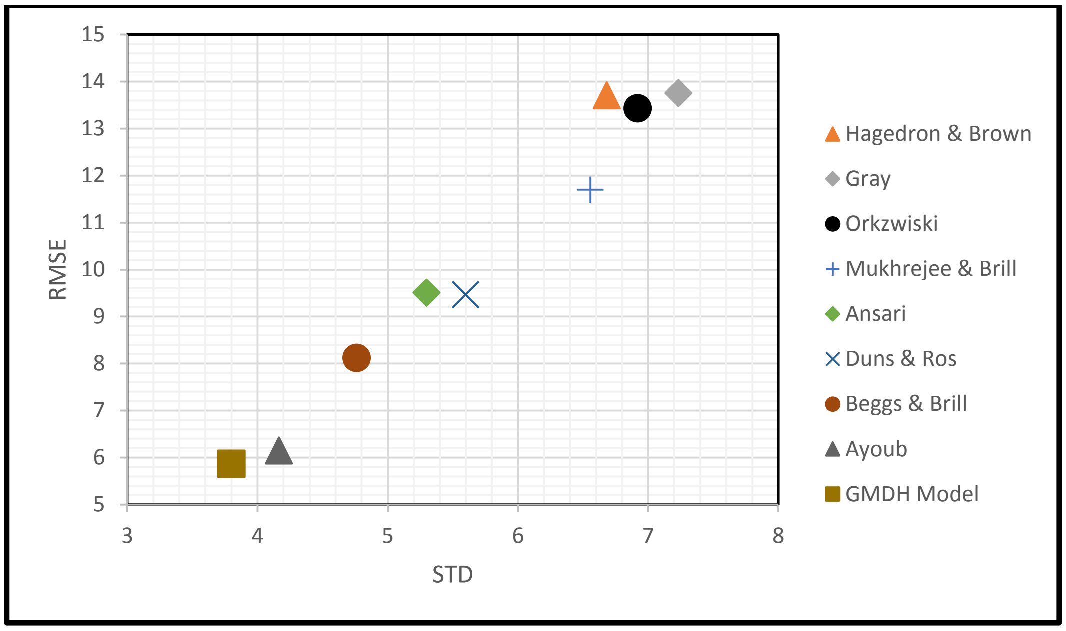

Figure 4.26 Root Mean Square Error against Standard Deviation for All Models

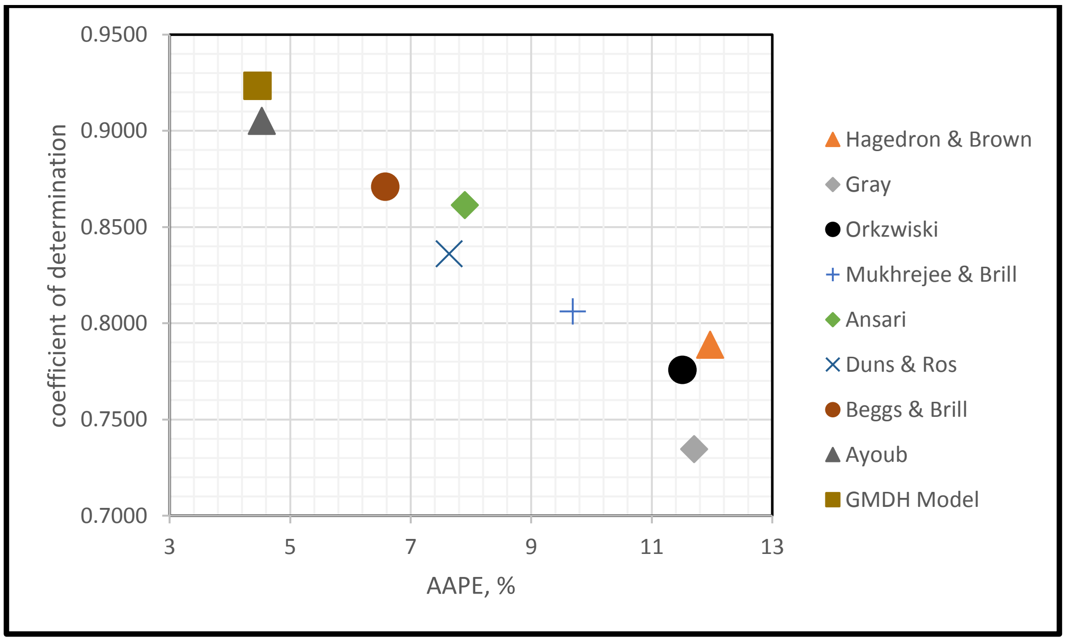

Figure 4.27 Average Absolute Relative Errors against the Confident of Determination for All Mode

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusions

This project aims at developing a model based on Group method of data handling (GMDH) approach. The literature study has shown that none of the current methods used to predict the pressure drop is applicable for general usage.

GMDH approach has been used successfully in developing a model that can estimate the pressure drop in vertical wells. This developed model showed better results when it has been compared against the common used models in the industry.

Comparison of the statistical error proves the GMDH model superiority over the existing correlations and models. The developed model achieved the lowest average absolute percent relative error (4.4623%), the lowest root mean square error (5.8616), the highest coefficient of determination (0.9233) and the lowest standard deviation of error (3.8008%). The trend analysis of the model showed that the model is correctly predicting predicted the expected effects of the independent variables on pressure drop.

Not to be forgotten, the new developed model is highly recommended to be used within the same range of the used data. Otherwise, unexpected results may come up when different ranges of inputs variables is used.

The GMDH model is successfully developed and the objectives of this project are successfully achieved.

5.2 Recommendations

Based on the above conclusions, the author suggests the following recommendation:

- A wide range of data that can be collected from different fields with additional input variables can be used to construct more accurate model using GMDH approach.
- More improvements and developments in the group method of data handling code and process for predicting pressure drop in the multiphase vertical flowing wells will definitely lead to better and accurate prediction in the future. Hence, all focuses and researches are highly recommended to go through that direction.
- Smart simulator like PIPESIM and PROSPER can be utilized to double check the presented models results.

Not to be forgotten, there are still many empirical correlations, mechanistic models and artificial neural networks in the literature which have not be evaluated in this project and may have more or less accuracy compared to GMDH model for predicting pressure drop in vertical wells. However, the methods were selected based on the author's perspective. And therefore, all the conclusions and recommendations were based on the selected me

REFERENCES

Abdul-Majeed, G. H., & Al-Mashat, A. M. (2000). A mechanistic model for vertical and inclined two phase slug flow. *Journal of Petroleum Science and Engineeringn*, 59-67.

Al-Attar, H. H., Mohamed, M. Y., & AMin, M. E. (2012). A Modified Version of the Aziz et al. Multiphase Flow correlation Imporoves Ressure drop Calculation in High-Rate Oil wells. *EAGE annula conference and exhibition, 4-7 June 2012.* Copenhagen, Denemark.

Ansari, A. M., Sylvester, N. D., Sarica, C., Shoham, O., & Brill, J. P. (1994). A Comperhensive Mechanistic Model for Upward Two-Phase Flow in Wellbore. *SPE Production and Facilities, May 1994*, 143-152.

Ayoub , M. A. (2011). *Development and Testing of Universal Pressure Drop Model in Pipelines Using Abductive and Artificial Neural Networks.* Bandar Seri Iskander, Perak: Phd Thesis, Universiti Teknologi Petronas.

Ayoub, M. A. (March 2004). *Development and Testing of an Artificial Neural Network Model for Predicting Bottomhole Pressure in Vertical Multiphase Flow.* Dahran, Saudi Arabia: M S Thesis, King Fahd University of Petroleum and Minerals.

Aziz, K., Govier, G. W., & Fogarasi, M. (1972). Pressure Drop in Wells producing Oil and Gas. *The Journal of Canadian Petroleum*, 38-48.

Beggs, H. D., & Brill, J. P. (May 1973). A Study in Two-Phase Flow in Inclined Pipes. *Journal of Petroleum Technology*, 607-17., AIME 255.

Chierici, G. L., Ciucci, G. M., & Sclocchi, G. (1974). Two-Phase Vertical Flow in Oil Wells- Prediction of Pressure drop. *Journal of Petroleum Technology, August 1974*, 927-938.

Dun, R., & Ros, N. C. (1963). VERTICAL FLOW OF GAS AND LIQUID MIXTURES IN WELLS. *Proc., Sixth World Pet. Con- gress, Frankfort (J nne 19-26, 1963) Section II, Paper 22-PD6.*, (pp. 451-465).

Farlow, S. J. (1981). The GMDH algorith of ivkhnenko. *The American Statistian, Vol.35, No.4*, 210-215.

Farlow, S. J. (1984). *The GMDH algorithm," in Self-Organizing Methods in Modeling: GMDH Type Algorithms.* New York: Marcel-Dekker.

Gomez, L. E., Shoham, O., Schmidt, Z., Chokshi, R. N., & Northug, T. (2000). Unified Mechanistic Model Foe Steady State Two-Phase Flow: Horizontal to Vertical Upward Flow. *SPE Journal, Vol. 5, No. 3, September* , 393-350.

Gould, T. L., Tek, M. R., & Katz, D. L. (1974). Two-Phase Flow Through Vertical, Inclined or Curved Pipe. *Journal of Petroleum Technology*, 915-926.

Govier, G., & Aziz, k. (1972). The Flow of Complex Mixtures in Pipes. *Van Nostrand-Reinhold, New York.*

Gregory, G. A. (1974). Application of Factorial Design Analysis to Producing Well Pressure-Drop Modelling. *The Journal of Canadian Petroelum Technology*, 21-27.

Griffith, P., & Wallis, G. B. (1961). Two-Phase Slug Flow. *Journal of heat Transefer, A.S.M.E transaction (AUG 1961) Vol.83*, 307-320.

Griffith, P., Lau, C. W., Hon, P. C., & Pearson, J. F. (1975). Two Phase Pressure Drop in Inclined and Vertical Wells.

Hagedorn, A., & Brown, K. (1965). Experimental study of pressure gradients occurring during continuous two-phase flow in small diameter vertical conduits. *Journal of Petroleum Technology (April 1965) 475; Tran., AIME*.

Hasan, R., & Kabir, S. (2005). A Simple Model for Annular Two-Phase Flow in Wellbores. *SPE Annual Technical Conference and Exhibition.* Dallas, Texas, U.S.A: Paper SPE 95523.

Ivakhenko, A. G. (1968). The Group Method of Data Handling A Rival of the Method of Stochastic Approximation.

Ivakhnenko, A. G. (1966). Group Method of Data Handling a Rival of the Method of Stochastic Approximation. *Soviet Automatic Control, 13,* 43-71.

Ivakhnenko, A. G. (1971). Polynomial Theory of Complex Systemq. *IEEE Transections on System, Man and Cybernetics*, 364-378.

Jahanandish, I., Sakimifard, B., & Jalalifar, H. (2011). Predicting bottomhole pressure in vertical multiphase flowing wells using artificial neural networks. *Journal of Petroleum science and engineering*, 336-342.

Mohaghegh, S., & Ameri, S. (1995). Artificial Neural Network As A Valuable Tool For Petroleum Engineers. *West Virginia University*. U.S.A. Telex: SPE 29220.

Mohammadpoor, M., Shahbazi, K., Torabi, F., & Qazfini, A. (2010). A new methodology for prediction of bottomhole flowing pressure in vertical multiphase flow in Iranian oil fields using artificial neural networks (ANNs). *SPE latin american and caribbean petroleum engineering conference, 1-3, December 2010.* Lima, Peru.

Moradi, B., Awang, M., & Shoushtari, M. A. (2011). Pressure Drop Prediction in Deep Gas Wells. *SPE Asia pacific oil & gas conference and exhibition, September 20-22, 2011.* Jakerta, Inonesia.

Mukherjee, H., & Brill, J. P. (December 1985). Pressure DRop Correlations for Inclined Two-Phase Flow. *Journal of Energy Resources Technology*, 549-554.

Orkiszewski, J. (1967). Predicting Two-.Phase Pressure Drops in Vertical Pipe. *Journal of Petroleum Technology,* 829-838.

Osman, E. A., & Abdel-Aal, R. E. (2002). Abductive Networks: A New Modeling Tool for the Oil and Gas Industry. *SPE Asia Pacific Oil and Gas Conference and Exhibition, 8-10 October 2002.* Melbourne, Australia: Research Institute, KFUPM, Dhahran, Saudi Arabia.

Palisetti, R. V., & Heinze, L. (1998). Simplified Approach for Predicting Pressure Profiles in a Flowing Well. *SPE Permain Basin Oil and Gas REcovery Conference , 23-26 March, 1998.* Midland, Texas.

Persad, S. (2005). Evaluation of Multiphae-Flow Correlations for Gas Wells Located Off the Trinidad Southeast Coast. *Latin American and Carribean Petroluem Engineering Comference.* Rio de Janeiro, Brazil, 20-23 June.

Poettmann, F. H., & Carpenter, P. G. (n.d.). The Multiphase Flow of Gas, Oil, and Water Through Vertical Flow Strings with Application to the Design of Gas-lift Installations. 257-317.

Pucknell, J., Mason, J., & Vervest, E. (1993). An Evaluation of Recent Mechanistic Models of Multiphase Flow for Predicting Pressure Drops in Oil and Gas Wells. *Paper SPE26682 , the 1993 offshore European Conference , 7-10 September.* Aberdeen.

Soni, S., Kelkar, m., & Perez, C. (2009). Pressure Drop Predictions in Tubing in The Presence of Surfactants. *SPE production and operations symposium.* Oklahoma, USA.

Subsurface Controlled Subsurface Safety Valve Sizing Computer Program. (1978). In *Institute American Petroleum.* API Manual 14 BM, Second Ed., p. 38, API.

Sukubo, I. A., & Igboanugo, A. C. (2011). Improved analytical model for predicting field production performance in vertical multiphase flow in pipes using MATLAB; A case study. *the Nigeria annual international conference and exibition,30 July - 3 August 2011.* Abuja, Nigeria.

Takacs, G. (2001). Consideration on the Selection of an Optimum Vertical Multiphase Pressure drop Prediction Model for oil Wells. *SPE Production and Operation Symposium , 24-27 March 2001.* Oklahoma.

Trick, M. D. (2003). Comparision of Correlations For Predicting Wellbore Pressure Losses in Gas-Condensate and Gas-Water Wells. *The Petroleum Society's Canadian International Petroleum Conference.* Calgari, Alberta: Neotechnology Consultants Ltd.

Wei Ma , S., Chen, L., Huan Kou, C., & Pei Wang, A. (2009). Application of Group Method of Data Handling to Stream-Way Transition. *International Joint Conference on Artificial Intelligence* (pp. 301-304). IEEE.

Xiao, J. J., Shoham, O., & Brill, J. P. (1990). A Comperhensive Mechanistic Model for Two-Phase Flow in Pipelines. *65th Anuual technical Conference and Exibition of the Society of Petroleum Engineering, September 23-26* (pp. 167-180). New Orleans, LA: University of Tulsa.

Zafareh, F., Hill, A. D., & Podio, A. L. (1988). Flow regimes in vertical and inclined Oil/Water flow in pipes. *63rd Annual technical conference and exibition of the society of petroleum engineers, October 2-5*. Huoston, TX: U. of Texas.

APPENDIX A - Statistical Error Equation

a) Average Absolut Percent Relative Error:

$$E_a = \frac{1}{n}\sum_{i=1}^{n}|E_i|$$

b) Average Percent Relative Error:

$$E_r = \frac{1}{n}\sum_{i=1}^{n}E_i$$

c) Maximum Absolute Relative Error:

$$E_{max} = {}_{i+1}^{n}\max|E_i|$$

d) Minimum Absolute Relative Error:

$$E_{max} = {}_{i+1}^{n}\min|E_i|$$

e) Root Mean Square Error:

$$\mathrm{RMSE} = \left[\frac{1}{n}\sum_{i=1}^{n}{E_i}^2\right]^{0.5}$$

f) Coefficient of determination:

$$R^2 = \sqrt{1 - \frac{\sum_{i=1}^{n}[(\Delta P)_m - (\Delta P)_c]}{\sum_{i=1}^{n}[(\Delta P)_m - \overline{\Delta\Delta P}]}}$$

g) Standard Deviation:

$$STD = \sqrt{\left[\left(\frac{1}{m-n-1}\right)\right]\left[\sum_{i=1}^{n}\left\{\frac{[(\Delta P)_m - (\Delta P)_c]}{(\Delta P)_m}\right\} * 100\right]^2}$$

Where, E_i is the relative deviation of a calculated value from the measured value;

$$\boldsymbol{E_i} = \left[\frac{(\Delta \boldsymbol{P})_{\boldsymbol{m}} - (\Delta \boldsymbol{P})_{\boldsymbol{c}}}{\Delta \boldsymbol{P}_{\boldsymbol{m}}}\right] * \mathbf{100}\%, \qquad \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \dots\dots\dots, \boldsymbol{n}$$

Where:

$(\Delta P)_m = Measured\ value\ pressure\ drop$

$(\Delta P)_c = calculated\ valuepressure\ drop$

$$\overline{\Delta\Delta P} = \frac{1}{n}\sum_{i=1}^{n}[(\Delta P_m)]_i$$

APPENDIX B- GMDH CODE

Function gmdhbuild

```
function [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, 
inputsMore, ...
maxNumNeurons, decNumNeurons, p, critNum, delta, Xv, Yv, verbose)
% GMDHBUILD
% Builds a GMDH-type polynomial neural network using a simple
% layer-by-layer approach
%
% Call
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons, p, critNum, delta, Xv, Yv, 
verbose)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons, p, critNum, delta, Xv, Yv)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons, p, critNum, delta)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons, p, critNum)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons, p)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons,
%                      decNumNeurons)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore, 
maxNumNeurons)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs, inputsMore)
%   [model, time] = gmdhbuild(Xtr, Ytr, maxNumInputs)
%   [model, time] = gmdhbuild(Xtr, Ytr)
%
% Input
% Xtr, Ytr      : Training data points (Xtr(i,:), Ytr(i)), i = 
1,...,n
% maxNumInputs : Maximum number of inputs for individual neurons - 
if set
%                to 3, both 2 and 3 inputs will be tried (default = 
2)
% inputsMore    : Set to 0 for the neurons to take inputs only from 
the
%                preceding layer, set to 1 to take inputs also from 
the
%                original input variables (default = 1)
% maxNumNeurons: Maximal number of neurons in a layer (default = 
equal to
%                the number of the original input variables)
% decNumNeurons: In each following layer decrease the number of 
allowed
%                neurons by decNumNeurons until the number is equal 
to 1
%                 (default = 0)
% p             : Degree of polynomials in neurons (allowed values 
are 2 and
%                3) (default = 2)
% critNum      : Criterion for evaluation of neurons and for
stopping.
%                In each layer only the best neurons (according to
the
%                criterion) are retained, and the rest are
discarded.
%                (default = 2)
%                0 = use validation data (Xv, Yv)
%                1 = use validation data (Xv, Yv) as well as
training data
%                2 = use Corrected Akaike's Information Criterion
(AICC)
%                3 = use Minimum Description Length (MDL)
%                Note that both choices 0 and 1 correspond to the so
called
%                "regularity criterion".
% delta        : How much lower the criterion value of the network's
new
%                layer must be comparing the the network's preceding
layer
%                (default = 0, which means that new layers will be
added as
%                long as the value gets better (smaller))
% Xv, Yv       : Validation data points (Xv(i,:), Yv(i)), i =
1,...,nv
%                (used when critNum is equal to either 0 or 1)
% verbose      : Set to 0 for no verbose (default = 1)
%
% Output
% model        : GMDH model - a struct with the following elements:
%    numLayers     : Number of layers in the network
%    d             : Number of input variables in the training data
set
%    maxNumInputs  : Maximal number of inputs for neurons
%    inputsMore    : See argument "inputsMore"
%    maxNumNeurons : Maximal number of neurons in a layer
%    p             : See argument "p"
%    critNum       : See argument "critNum"
%    layer         : Full information about each layer (number of
neurons,
%                    indexes of inputs for neurons, matrix of
exponents for
%                    polynomial, polynomial coefficients)
%                    Note that the indexes of inputs are in range
[1..d] if
%                    an input is one of the original input
variables, and
%                    in range [d+1..d+maxNumNeurons] if an input is
taken
%                    from a neuron in the preceding layer.
% time         : Execution time (in seconds)
%
% Please give a reference to the software web page in any
publication
% describing research performed using the software e.g., like this:
% Jekabsons G. GMDH-type Polynomial Neural Networks for Matlab,
2010,
% available at http://www.cs.rtu.lv/jekabsons/

% This source code is tested with Matlab version 7.1 (R14SP3).
```

```
%
======================================================================
=====
% GMDH-type polynomial neural network
% Version: 1.5
% Date: June 2, 2011
% Author: Gints Jekabsons (gints.jekabsons@rtu.lv)
% URL: http://www.cs.rtu.lv/jekabsons/
%
% Copyright (C) 2009-2011  Gints Jekabsons
%
% This program is free software: you can redistribute it and/or
modify
% it under the terms of the GNU General Public License as published
by
% the Free Software Foundation, either version 2 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program. If not, see
<http://www.gnu.org/licenses/>.
%
======================================================================
=====

if nargin < 2
    error('Too few input arguments.');
end
[n, d] = size(Xtr);
[ny, dy] = size(Ytr);
if (n < 2) || (d < 2) || (ny ~= n) || (dy ~= 1)
    error('Wrong training data sizes.');
end
if nargin < 3
    maxNumInputs = 2;
elseif (maxNumInputs ~= 2) && (maxNumInputs ~= 3)
    error('Number of inputs for neurons should be 2 or 3.');
end
if (d < maxNumInputs)
    error('Numbet of input variables in the data is lower than the
number of inputs for individual neurons.');
end
if nargin < 4
    inputsMore = 1;
end
if (nargin < 5) || (maxNumNeurons <= 0)
    maxNumNeurons = d;
end
if maxNumNeurons > d * 2
    error('Too many neurons in a layer. Maximum is two times the
number of input variables.');
end
if maxNumNeurons < 1
    error('Too few neurons in a layer. Minimum is 1.');
end
if (nargin < 6) || (decNumNeurons < 0)
```

```
    decNumNeurons = 0;
end
if nargin < 7
    p = 2;
elseif (p ~= 2) && (p ~= 3)
    error('Degree of individual neurons should be 2 or 3.');
end
if nargin < 8
    critNum = 2;
end
if any(critNum == [0,1,2,3]) == 0
    error('Only four values for critNum are available (0,1 - use
validation data; 2 - AICC; 3 - MDL).');
end
if nargin < 9
    delta = 0;
end
if (nargin < 11) && (critNum <= 1)
    error('Evaluating the models in validation data requires
validation data set.');
end
if (nargin >= 11) && (critNum <= 1)
    [nv, dv] = size(Xv);
    [nvy, dvy] = size(Yv);
    if (nv < 1) || (dv ~= d) || (nvy ~= nv) || (dvy ~= 1)
        error('Wrong validation data sizes.');
    end
end
if nargin < 12
    verbose = 1;
end
ws = warning('off');
if verbose ~= 0
    fprintf('Building GMDH-type neural network...\n');
end
tic;
if p == 2
    numTermsReal = 6 + 4 * (maxNumInputs == 3); %6 or 10 terms
else
    numTermsReal = 10 + 10 * (maxNumInputs == 3); %10 or 20 terms
end

Xtr(:, d+1:d+maxNumNeurons) = zeros(n, maxNumNeurons);
if critNum <= 1
    Xv(:, d+1:d+maxNumNeurons) = zeros(nv, maxNumNeurons);
end
%start the main loop and create layers
model.numLayers = 0;
while 1
    if verbose ~= 0
        fprintf('Building layer #%d...\n', model.numLayers + 1);
    end
    layer(model.numLayers + 1).numNeurons = 0;
    modelsTried = 0;
    layer(model.numLayers + 1).coefs = zeros(maxNumNeurons,
numTermsReal);
    for numInputsTry = maxNumInputs:-1:2

        %create matrix of exponents for polynomials
        if p == 2
            numTerms = 6 + 4 * (numInputsTry == 3); %6 or 10 terms
            if numInputsTry == 2
                r = [0,0;0,1;1,0;1,1;0,2;2,0];
            else
                r =
[0,0,0;0,0,1;0,1,0;1,0,0;0,1,1;1,0,1;1,1,0;0,0,2;0,2,0;2,0,0];
            end
        else
            numTerms = 10 + 10 * (numInputsTry == 3); %10 or 20
terms
            if numInputsTry == 2
                r = [0,0;0,1;1,0;1,1;0,2;2,0;1,2;2,1;0,3;3,0];
            else
                r =
[0,0,0;0,0,1;0,1,0;1,0,0;0,1,1;1,0,1;1,1,0;0,0,2;0,2,0;2,0,0; ...

1,1,1;0,1,2;0,2,1;1,0,2;1,2,0;2,0,1;2,1,0;0,0,3;0,3,0;3,0,0];
            end
        end
        %create matrix of all combinations of inputs for neurons
        if model.numLayers == 0
            combs = nchoosek(1:1:d, numInputsTry);
        else
            if inputsMore == 1
                combs = nchoosek([1:1:d
d+1:1:d+layer(model.numLayers).numNeurons], numInputsTry);
            else
                combs =
nchoosek(d+1:1:d+layer(model.numLayers).numNeurons, numInputsTry);
            end
        end
        %delete all combinations in which none of the inputs are
from the preceding layer
        if model.numLayers > 0
            i = 1;
            while i <= size(combs,1)
                if all(combs(i,:) <= d)
                    combs(i,:) = [];
                else
                    i = i + 1;
                end
            end
        end
        makeEmpty = 1;

        %try all the combinations of inputs for neurons
        for i = 1 : size(combs,1)

            %create matrix for all polynomial terms
            Vals = ones(n, numTerms);
            if critNum <= 1
                Valsv = ones(nv, numTerms);
            end
            for idx = 2 : numTerms
                bf = r(idx, :);
                t = bf > 0;
                tmp = Xtr(:, combs(i,t)) .^ bf(ones(n, 1), t);
                if critNum <= 1
                    tmpv = Xv(:, combs(i,t)) .^ bf(ones(nv, 1), t);
                end
                if size(tmp, 2) == 1
                    Vals(:, idx) = tmp;
```

```
                        if critNum <= 1
                            Valsv(:, idx) = tmpv;
                        end
                    else
                        Vals(:, idx) = prod(tmp, 2);
                        if critNum <= 1
                            Valsv(:, idx) = prod(tmpv, 2);
                        end
                    end
                end
                %calculate coefficients and evaluate the network
                coefs = (Vals' * Vals) \ (Vals' * Ytr);
                modelsTried = modelsTried + 1;
                if ~isnan(coefs(1))
                    predY = Vals * coefs;
                    if critNum <= 1
                        predYv = Valsv * coefs;
                        if critNum == 0
                            crit = sqrt(mean((predYv - Yv).^2));
                        else
                            crit = sqrt(mean([(predYv - Yv).^2; (predY -
Ytr).^2]));
                        end
                    else
                        comp = complexity(layer, model.numLayers,
maxNumNeurons, d, combs(i,:)) + size(coefs, 2);
                        if critNum == 2 %AICC
                            if (n-comp-1 > 0)
                                crit = n*log(mean((predY - Ytr).^2)) +
2*comp + 2*comp*(comp+1)/(n-comp-1);
                            else
                                coefs = NaN;
                            end
                        else %MDL
                            crit = n*log(mean((predY - Ytr).^2)) +
comp*log(n);
                        end
                    end
                end
                if ~isnan(coefs(1))
                    %add the neuron to the layer if
                    %1) the layer is not full;
                    %2) the new neuron is better than an existing worst
one.
                    maxN = maxNumNeurons - model.numLayers *
decNumNeurons;
                    if maxN < 1, maxN = 1; end;
                    if layer(model.numLayers + 1).numNeurons < maxN
                        %when the layer is not yet full
                        if (maxNumInputs == 3) && (numInputsTry == 2)
                            layer(model.numLayers +
1).coefs(layer(model.numLayers + 1).numNeurons+1, :) = [coefs'
zeros(1,4+6*(p == 3))];
                            layer(model.numLayers +
1).inputs(layer(model.numLayers + 1).numNeurons+1, :) = [combs(i, :)
0];
                        else
                            layer(model.numLayers +
1).coefs(layer(model.numLayers + 1).numNeurons+1, :) = coefs;
                            layer(model.numLayers +
1).inputs(layer(model.numLayers + 1).numNeurons+1, :) = combs(i, :);
```

```
                    end
                    layer(model.numLayers +
1).comp(layer(model.numLayers + 1).numNeurons+1) = length(coefs);
                    layer(model.numLayers +
1).crit(layer(model.numLayers + 1).numNeurons+1) = crit;
                    layer(model.numLayers +
1).terms(layer(model.numLayers + 1).numNeurons+1).r = r;
                    if makeEmpty == 1
                        Xtr2 = [];
                        if critNum <= 1
                            Xv2 = [];
                        end
                        makeEmpty = 0;
                    end
                    Xtr2(:, layer(model.numLayers + 1).numNeurons+1)
= predY;
                    if critNum <= 1
                        Xv2(:, layer(model.numLayers +
1).numNeurons+1) = predYv;
                    end
                    if (layer(model.numLayers + 1).numNeurons == 0)
|| ...
                       (layer(model.numLayers + 1).crit(worstOne) <
crit)
                        worstOne = layer(model.numLayers +
1).numNeurons + 1;
                    end
                    layer(model.numLayers + 1).numNeurons =
layer(model.numLayers + 1).numNeurons + 1;
                else
                    %when the layer is already full
                    if (layer(model.numLayers + 1).crit(worstOne) >
crit)
                        if (maxNumInputs == 3) && (numInputsTry ==
2)
                            layer(model.numLayers +
1).coefs(worstOne, :) = [coefs' zeros(1,4+6*(p == 3))];
                            layer(model.numLayers +
1).inputs(worstOne, :) = [combs(i, :) 0];
                        else
                            layer(model.numLayers +
1).coefs(worstOne, :) = coefs;
                            layer(model.numLayers +
1).inputs(worstOne, :) = combs(i, :);
                        end
                        layer(model.numLayers + 1).comp(worstOne) =
length(coefs);
                        layer(model.numLayers + 1).crit(worstOne) =
crit;
                        layer(model.numLayers + 1).terms(worstOne).r
= r;
                        Xtr2(:, worstOne) = predY;
                        if critNum <= 1
                            Xv2(:, worstOne) = predYv;
                        end
                        [dummy, worstOne] =
max(layer(model.numLayers + 1).crit);
                    end
                end
            end
        end
```

```
    if verbose ~= 0
        fprintf('Neurons tried in this layer: %d\n', modelsTried);
        fprintf('Neurons included in this layer: %d\n',
layer(model.numLayers + 1).numNeurons);
        if critNum <= 1
            fprintf('RMSE in the validation data of the best neuron:
%f\n', min(layer(model.numLayers + 1).crit));
        else
            fprintf('Criterion value of the best neuron: %f\n',
min(layer(model.numLayers + 1).crit));
        end
    end

    %stop the process if there are too few neurons in the new layer
    if ((inputsMore == 0) && (layer(model.numLayers + 1).numNeurons
< 2)) || ...
       ((inputsMore == 1) && (layer(model.numLayers + 1).numNeurons
< 1))
        if (layer(model.numLayers + 1).numNeurons > 0)
            model.numLayers = model.numLayers + 1;
        end
        break
    end
    %if the network got "better", continue the process
    if (layer(model.numLayers + 1).numNeurons > 0) && ...
       ((model.numLayers == 0) || ...
        (min(layer(model.numLayers).crit) -
min(layer(model.numLayers + 1).crit) > delta) )
%(min(layer(model.numLayers + 1).crit) <
min(layer(model.numLayers).crit)) )
        model.numLayers = model.numLayers + 1;
    else
        if model.numLayers == 0
            warning(ws);
            error('Failed.');
        end
        break
    end

    %copy the output values of this layer's neurons to the training
    %data matrix
    Xtr(:, d+1:d+layer(model.numLayers).numNeurons) = Xtr2;
    if critNum <= 1
        Xv(:, d+1:d+layer(model.numLayers).numNeurons) = Xv2;
    end
end

model.d = d;
model.maxNumInputs = maxNumInputs;
model.inputsMore = inputsMore;
model.maxNumNeurons = maxNumNeurons;
model.p = p;
model.critNum = critNum;
%only the neurons which are actually used (directly or indirectly)
to
%compute the output value may stay in the network
[dummy best] = min(layer(model.numLayers).crit);
model.layer(model.numLayers).coefs(1,:) =
layer(model.numLayers).coefs(best,:);
model.layer(model.numLayers).inputs(1,:) =
layer(model.numLayers).inputs(best,:);
```

```
model.layer(model.numLayers).terms(1).r =
layer(model.numLayers).terms(best).r;
model.layer(model.numLayers).numNeurons = 1;
if model.numLayers > 1
    for i = model.numLayers-1:-1:1 %loop through all the layers
        model.layer(i).numNeurons = 0;
        for k = 1 : layer(i).numNeurons %loop through all the
neurons in this layer
            newNum = 0;
            for j = 1 : model.layer(i+1).numNeurons %loop through
all the neurons which will stay in the next layer
                for jj = 1 : maxNumInputs %loop through all the
inputs
                    if k == model.layer(i+1).inputs(j,jj) - d
                        if newNum == 0
                            model.layer(i).numNeurons =
model.layer(i).numNeurons + 1;
model.layer(i).coefs(model.layer(i).numNeurons,:) =
layer(i).coefs(k,:);
model.layer(i).inputs(model.layer(i).numNeurons,:) =
layer(i).inputs(k,:);
model.layer(i).terms(model.layer(i).numNeurons).r =
layer(i).terms(k).r;
                            newNum = model.layer(i).numNeurons + d;
                            model.layer(i+1).inputs(j,jj) = newNum;
                        else
                            model.layer(i+1).inputs(j,jj) = newNum;
                        end
                        break
                    end
                end
            end
        end
    end
end

time = toc;
warning(ws);
if verbose ~= 0
    fprintf('Done.\n');
    used = zeros(d,1);
    for i = 1 : model.numLayers
        for j = 1 : d
            if any(any(model.layer(i).inputs == j))
                used(j) = 1;
            end
        end
    end
    fprintf('Number of layers: %d\n', model.numLayers);
    fprintf('Number of used input variables: %d\n', sum(used));
    fprintf('Execution time: %0.2f seconds\n', time);
end
return
%====================  Auxiliary functions  ====================
function [comp] = complexity(layer, numLayers, maxNumNeurons, d,
connections)
%calculates the complexity of the network given output neuron's
connections
%(it is assumed that the complexity of a network is equal to the
number of
%all polynomial terms in all it's neurons which are actually
connected
%(directly or indirectly) to network's output)
comp = 0;
if numLayers == 0
    return
end
c = zeros(numLayers, maxNumNeurons);
for i = 1 : numLayers
    c(i, :) = layer(i).comp(:)';
end
%{
%unvectorized version:
for j = 1 : length(connections)
    if connections(j) > d
        comp = comp + c(numLayers, connections(j) - d);
        c(numLayers, connections(j) - d) = -1;
    end
end
%}
ind = connections > d;
if any(ind)
    comp = comp + sum(c(numLayers, connections(ind) - d));
    c(numLayers, connections(ind) - d) = -1;
end
%{
%unvectorized version:
for i = numLayers-1:-1:1
    for j = 1 : layer(i).numNeurons
        for k = 1 : layer(i+1).numNeurons
            if (c(i+1, k) == -1) && (c(i, j) > -1) && ...
               any(layer(i+1).inputs(k,:) == j + d)
                comp = comp + c(i, j);
                c(i, j) = -1;
            end
        end
    end
end
%}
for i = numLayers-1:-1:1
        for k = 1 : layer(i+1).numNeurons
            if c(i+1, k) == -1
                inp = layer(i+1).inputs(k,:);
                used = inp > d;
                if any(used)
                    ind = inp(used) - d;
                    ind = ind(c(i, ind) > -1);
                    if ~isempty(ind)
                        comp = comp + sum(c(i, ind));
                        c(i, ind) = -1;
                    end
                end
            end
        end
end
return
```

function gmdhpredict

```
function Yq = gmdhpredict(model, Xq)
% GMDHPREDICT
% Predicts output values for the given query points Xq using a GMDH
model
%
% Call
%   [Yq] = gmdhpredict(model, Xq)
%
% Input
% model      : GMDH model
% Xq         : Inputs of query data points (Xq(i,:)), i = 1,...,nq
%
% Output
% Yq         : Predicted outputs of query data points (Yq(i)), i =
1,...,nq

% This source code is tested with Matlab version 7.1 (R14SP3).

%
=====================================================================
=====
% GMDH-type polynomial neural network
% Version: 1.5
% Date: June 2, 2011
% Author: Gints Jekabsons (gints.jekabsons@rtu.lv)
% URL: http://www.cs.rtu.lv/jekabsons/
%
% Copyright (C) 2009-2011  Gints Jekabsons
%
% This program is free software: you can redistribute it and/or
modify
% it under the terms of the GNU General Public License as published
by
% the Free Software Foundation, either version 2 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program. If not, see
<http://www.gnu.org/licenses/>.
%
=====================================================================
=====

if nargin < 2
    error('Too few input arguments.');
end
if model.d ~= size(Xq, 2)
    error('The matrix should have the same number of columns as the
matrix with which the network was built.');
end

[n, d] = size(Xq);
Yq = zeros(n, 1);

for q = 1 : n
    for i = 1 : model.numLayers
        if i ~= model.numLayers
            Xq_tmp = zeros(1, model.layer(i).numNeurons);
        end
        for j = 1 : model.layer(i).numNeurons

            %create matrix for all polynomial terms
            numTerms =  size(model.layer(i).terms(j).r,1);
            Vals = ones(numTerms,1);
            for idx = 2 : numTerms
                bf = model.layer(i).terms(j).r(idx, :);
                t = bf > 0;
                tmp = Xq(q, model.layer(i).inputs(j,t)) .^ bf(1, t);
                if size(tmp, 2) == 1
                    Vals(idx,1) = tmp;
                else
                    Vals(idx,1) = prod(tmp, 2);
                end
            end

            %predict output value
            predY = model.layer(i).coefs(j,1:numTerms) * Vals;
            if i ~= model.numLayers
                %Xq(q, d+j) = predY;
                Xq_tmp(j) = predY;
            else
                Yq(q) = predY;
            end

        end
        if i ~= model.numLayers
            Xq(q, d+1:d+model.layer(i).numNeurons) = Xq_tmp;
        end
    end
end

return
```

function gmdhtest

```
function [MSE, RMSE, RRMSE, R2] = gmdhtest(model, Xtst, Ytst)
% GMDHTEST
% Tests a GMDH-type network model on a test data set (Xtst, Ytst)
%
% Call
%   [MSE, RMSE, RRMSE, R2] = gmdhtest(model, Xtst, Ytst)
%
% Input
% model      : GMDH model
% Xtst, Ytst: Test data points (Xtst(i,:), Ytst(i)), i = 1,...,ntst
%
% Output
% MSE        : Mean Squared Error
% RMSE       : Root Mean Squared Error
% RRMSE      : Relative Root Mean Squared Error
% R2         : Coefficient of Determination

% Copyright (C) 2009-2011  Gints Jekabsons

if nargin < 3
    error('Too few input arguments.');
end
if (size(Xtst, 1) ~= size(Ytst, 1))
```

```
    error('The number of rows in the matrix and the vector should be
equal.');
end
if model.d ~= size(Xtst, 2)
    error('The matrix should have the same number of columns as the
matrix with which the model was built.');
end
MSE = mean((gmdhpredict(model, Xtst) - Ytst) .^ 2);
RMSE = sqrt(MSE);
if size(Ytst, 1) > 1
    RRMSE = RMSE / std(Ytst, 1);
    R2 = 1 - MSE / var(Ytst, 1);
else
    RRMSE = Inf;
    R2 = Inf;
end
return
```

function gmdheq

```
function gmdheq(model, precision)
% gmdheq
% Outputs the equations of GMDH model.
%
% Call
%   gmdheq(model, precision)
%   gmdheq(model)
%
% Input
%   model           : GMDH-type model
%   precision       : Number of digits in the model coefficients
%                     (default = 15)

% This source code is tested with Matlab version 7.1 (R14SP3).

%
===================================================================
=====
% GMDH-type polynomial neural network
% Version: 1.5
% Date: June 2, 2011
% Author: Gints Jekabsons (gints.jekabsons@rtu.lv)
% URL: http://www.cs.rtu.lv/jekabsons/
%
% Copyright (C) 2009-2011  Gints Jekabsons
%
% This program is free software: you can redistribute it and/or
modify
% it under the terms of the GNU General Public License as published
by
% the Free Software Foundation, either version 2 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program. If not, see
<http://www.gnu.org/licenses/>.
```

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

if nargin < 1
    error('Too few input arguments.');
end
if (nargin < 2) || (isempty(precision))
    precision = 15;
end

if model.numLayers > 0
    p = ['%.' num2str(precision) 'g'];
    fprintf('Number of layers: %d\n', model.numLayers);
    for i = 1 : model.numLayers %loop through all the layers
        fprintf('Layer #%d\n', i);
        fprintf('Number of neurons: %d\n',
model.layer(i).numNeurons);
        for j = 1 : model.layer(i).numNeurons %loop through all the
neurons in the ith layer
            [terms inputs] = size(model.layer(i).terms(j).r);
%number of terms and inputs
            if (i == model.numLayers)
                str = ['y = ' num2str(model.layer(i).coefs(j,1),p)];
            else
                str = ['x' num2str(j + i*model.d) ' = '
num2str(model.layer(i).coefs(j,1),p)];
            end
            for k = 2 : terms %loop through all the terms
                if model.layer(i).coefs(j,k) >= 0
                    str = [str ' +'];
                else
                    str = [str ' '];
                end
                str = [str num2str(model.layer(i).coefs(j,k),p)];
                for kk = 1 : inputs %loop through all the inputs
                    if (model.layer(i).terms(j).r(k,kk) > 0)
                        for kkk = 1 :
model.layer(i).terms(j).r(k,kk)
                            if (model.layer(i).inputs(j,kk) <=
model.d)
                                str = [str '*x'
num2str(model.layer(i).inputs(j,kk))];
                            else
                                str = [str '*x'
num2str(model.layer(i).inputs(j,kk) + (i-2)*model.d)];
                            end
                        end
                    end
                end
            end
            disp(str);
        end
    end
else
    disp('The network has zero layers.');
end

return
```