USING LINEAR REGRESSION AND ANN TECHNIQUES IN DETERMINING VARIABLE IMPORTANCE

by

ADERIANA MUTHEU MBANDI

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Chemical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Mr Willie Coetzee

Cape Town
18/June/2009

Declaration

I, Andriannah Mbandi, declare that the contents of this thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed **Date**

Abstract

The use of Neural Networks in chemical engineering is well documented. There has also been an increase in research concerned with the explanatory capacity of Neural Networks although this has been hindered by the regard of Artificial Neural Networks (ANN's) as a black box technology.

Determining variable importance in complex systems that have many variables as found in the fields of ecology, water treatment, petrochemical production, and metallurgy, would reduce the variables to be used in optimisation exercises, easing complexity of the model and ultimately saving money. In the process engineering field, the use of data to optimise processes is limited if some degree of process understanding is not present.

The project objective is to develop a methodology that uses Artificial Neural Network (ANN) technology and Multiple Linear Regression (MLR) to identify explanatory variables in a dataset and their importance on process outputs. The methodology is tested by using data that exhibits defined and well known numeric relationships. The numeric relationships are presented using four equations.

The research project assesses the relative importance of the independent variables by using the "dropping method" on a regression model and ANN's. Regression used traditionally to determine variable contribution could be unsuccessful if a highly non-linear relationship exists. ANN's could be the answer for this shortcoming.

For differentiation, the explanatory variables that do not contribute significantly towards the output will be named "suspect variables". Ultimately the suspect variables identified in the regression model and ANN should be the same, assuming a good regression model and network.

The dummy variables introduced to the four equations are successfully identified as suspect variables. Furthermore, the degree of variable importance was determined using linear regression and ANN models. As the equations complexity increased, the linear regression models accuracy decreased, thus suspect variables are not correctly identified. The complexity of the equations does not affect the accuracy of the ANN model, and the suspect variables are correctly identified.

The use of R^2 and average error in establishing a criterion for identifying suspect variables is explored. It is established that the cumulative variable importance percentage (additive percentage), has to be below 5% for the explanatory variable to be considered a suspect variable.

Combining linear regression and ANN provides insight into the importance of explanatory variables and indeed suspect variables and their contribution can be determined. Suspect variables can be eliminated from the model once identified simplifying the model, and increasing accuracy of the model.

Acknowledgements

I wish to thank:

- Mr Willie Coetzee, my supervisor.

Dedication

I wish to thank Mrs Mary Mbandi, my mother, for all her support and encouragement.

Glossary

Backpropagation

General mathematical notation was used, in the form:

Where α β γ represent the following:

α can be one of the following

x, to denote an external input

h, to denote the internal output of a node being transformed nonlinearly

z, to denote the activation of a node (output of a node)

y, to denote the external output

d, to denote the desired network output

w, to denote the weight of a connection

, to denote the bias

can be one of the following

a number to stand for the identity of the layer; the number takes the values 1, 2 and 3 for input, hidden and output layer respectively. This is used for the internal outputs and bias before and after transformation.

P, to stand for the identity of a particular example. This is for external and internal outputs and for target output.

γ can be one of the following:

i, to denote the identity of the node that the connection comes from originally

ij, to denote i, the origin, and j, the termination, of the connection. This notation is only applicable to the weight notation.

Linear and Multiple Regression

X_i: input, predictor or explanatory variable.

Y_i: output or criterion variable.

a, bi: regression coefficients of regression.

ε: residual error for linear regression.

Y^l: model output.

R: Pearson correlation.

β_i: value of regression coefficient corresponding to the data value b.

t: the t-test for testing the null hypothesis.

F: corresponding test to the t-test.

P: test for assessing the consistency of the null hypothesis.

e_i: residual error for multiple linear regression.

S^2, σ: variance, standard deviation.

$\bar{X}_i$: average value of the input variable.

$\bar{Y}_i$: average value of the output variable.

$r_{yk,\,123}$: semi-partial correlations of X_i and Y in the multiple linear regression.

R_{yy}^l: multiple correlation coefficient.

SS_{reg}: sum of squares for regression.

SS_{tot}: total sum of squares.

SS_{res}: residual sum of squares.

H_o: null hypothesis test for multiple regression.

Table of contents

1. INTRODUCTION

The advent of computer use has brought about an avalanche of data, and the development of neural networks has afforded a promising technique to model data. Neural networks are seen as “universal approximators” as they can handle nonlinear multiple variable systems (Hornik, Stichcombe & White, 1989:360). The greatest advantage of artificial neural networks (ANN) is their user-friendliness, which hides from the user the complicated mathematical and computational network training procedure (Papadokonstantakis, Machefer, Schnitzlein & Lygeros, 2005:1647).

In the field of process engineering, advances in information technology brought about an increase in the availability of processed data. However, the use of this data to optimise processes is limited if some degree of process understanding is not present.

The use and application of neural networks in chemical engineering is well documented and the only limitation is the imagination of chemical engineers (Himmelblau et al., 2000:373). There has also been an increase in research on the explanatory capacity of neural networks, although this has been hindered by the fact that ANNs are regarded as a black box technology (Gevrey, 2003:259; Olden, 2004:389). ANN modelling has been used successfully as a pre-processing stage, which can be instrumental in the development of a successful model.

The data pre-processing stage is of great importance, especially in most process datasets with restricted quality and containing noise and faults (Hornik *et al*., 1989:360). Determining variable importance as a pre-processing stage in data modelling in complex systems that have many variables, such as ecological, water treatment, petrochemical plants and bio-processing, would reduce the variables to be used in the optimisation processes, easing the complexity of the model and ultimately saving money (Bruns, 2002:366; Grieu, 2006:2 ;Martinez, 1999:102).

The objective of this project is to develop a methodology that uses artificial neural network (ANN) technology and multiple linear regression to identify explanatory variables in a dataset and determine their importance for process outputs. The methodology is tested by using data that exhibit defined and well-known numeric relationships.

The research project assesses the relative importance of the independent variables by using the “dropping method” (Garson, 2007:6) on a regression model and ANNs. Regression used traditionally to determine variable contributions could be unsuccessful if a highly nonlinear relationship exists. ANNs have been shown to be highly competent approximators of many complex systems, and have been shown to have advantages over general linear models in predictive ability (Hastie, Tibishrani & Friedman, 2001:14), thus they could be the answer to this linear models shortcoming. However, the research does not advocate the use of ANN over linear regression, but rather a symbiotic relation. Linear regression is the well documented statistical methodology and ANNs are just beginning to be unravelled. The symbiosis already exists, as many statistical packages use ANN (Kemp, Zaradic & Hansen, 2007:326)

For differentiation, the explanatory variables that do not contribute significantly to the output will be named “suspect variables”.

The method used was as follows:

A linear regression was run that modelled the data against a fixed but variable target, measured with the absolute least squares method
The “dropping method” was used to identify suspect variables from the regression model

The data gained were run through a trained neural network in order to enhance the match of the modelled data against the original data.

The trained neural network was used to identify suspect variables.

It was determined whether the regression model and the ANN identify the same suspect variables.

Multiple regression attempts to model the relation between two or more independent variables (input) and the dependent variable (output) by fitting a linear equation to the observed data.

The observed data are:

(1.1)

And the model equation (also termed the regression line for k variables) is:

σ (1.2)

Using the least squares method (measured with the R^2 value), the best fitting line (plane) from the data is calculated by minimising the sum of the squared residuals (**ε**).

Multiple regression has three primary uses:

Understanding which input variables have the greatest effect on the output.
Knowing the direction of the effect of the input variable, e.g. increasing x_1 increases/decreases Y.

Using the model simulated to predict future values of the input variables when only the output variables are known.

To date it has been shown that multiple regression is able to establish that a set of independent variables contributes to the variance of a dependent variable to a significant extent. The significance of this contribution can be tested rigorously by the R^2 value, using the "dropping method". The importance can also be tested further by examining the beta weights attributed to each of the contributing and non-contributing variables.

ANNs are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. An ANN can be trained to perform a particular function by adjusting the value of the connections (weights) between elements, based on the complexity of a given problem (Demuth & Beale, 2004:8).

ANNs are trained so that a particular input leads to a specific target output. An input is presented to the network. The network compares the output to the target, and the network's weights are adjusted on the basis of this comparison until the network output matches the target. The target and output pairs are essential to the training of the network and, typically, many such pairs are used in what is termed as 'supervised learning' in training the network.

An ANN, once trained, can predict targets from inputs that have not been presented to the network before. It is on this premise that “suspect variables” should be identifiable, as the simulated output theoretically should not be affected when the “suspect variables” are altered.

The observed data to be used for modelling comprises of randomly generated data. The data is set up using the following equations:

Equation 1: — dummy variables $\mathbf{x_3}$, $\mathbf{x_4}$

Equation 2: $\sqrt{\quad}$ dummy variables $\mathbf{x_2}$, $\mathbf{x_4}$

Equation 3: — $\sqrt{\quad}$

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_7}$, $\mathbf{x_9}$

The third equation is a combination of the first two equations.

Equation 4:

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$

The general equations are used for generating random data of a range between 1 and 10. This can be done without loss of generality, since the data are produced at random. A number of matrices are constructed for all variables and then exported to Matlab. In general, the number of matrices created for each given equation is directly proportional to the number of input variables.

No. of matrices = no of input variables + 2 (1.3)

At first, all input variables are utilised to calculate the regression values, after which variables are successfully removed from the input while creating new matrices.

The output variable remains constant for each equation, as it is determined by a fixed relationship.

A neural network is created using the backpropagation algorithm. The matrices of the ANN are derived from the random data that are generated from the four equations in the range of 1 to 10.

X_{all} comprises all the input variables x_1, x_2,……., x_k.

Y_{all} comprises the output variable y

The default training algorithm that was used is trainlm, a network training function that updates weight and bias values according to the Levenberg-Marquardt optimisation algorithm. The Levenberg-Marquardt optimisation is an algorithm for least squares estimation of nonlinear parameters that outperforms the gradient descent and conjugate algorithm. It does so by approaching the second order training speed by approximating the Hessian matrix as opposed to computing it.

The trained network provides a platform for evaluating the effect of altering the values of input variables on the simulated output of the trained network. The trained network is represented with periodic alterations of all the variables that were used in the training network. The simulated output within each variable is compared to establish significant changes. The deficiency of change for the simulated output for a particular variable indicates that a specific variable does not contribute significantly to the output, and thus can be identified as a suspect variable.

2. LITERATURE REVIEW

2.1 MULTIPLE LINEAR REGRESSION

To define multiple linear regression it is best to first describe simpler linear regression. Linear regression attempts to find a relationship between the observed data by finding the best linear correlation (Edwards, 1976:3). The independent (also named input, predictor, explanatory) variable is X_i and the dependent variable (also named output, criterion) is Y_i. Linear regression models the data in the equation:

(2.1)

thereby estimating the coefficients a and b, which are determined on condition that the ε residual error is minimised.

(2.2)

$$\frac{\sum \quad \sum \quad \sum}{\sum \quad \sum}$$

$$\sum \quad \sum$$

The correlation coefficient of Y and X is defined by the Pearson-R correlation:

$$\frac{\sum \quad \frac{\sum \quad \sum}{}}{\sqrt{\left[\sum \quad \frac{\sum}{}\right]\left[\sum \quad \frac{\sum}{}\right]}} \qquad (2.3)$$

The value of the correlation coefficient R falls between the values 1 and -1. If the absolute value R^2 is calculated, then the maximum value is 1. The absolute value of R^2 provides an index of the degree to which a set of data points cluster around the proposed regression line.

If the points from the data fall along the regression line, the values of R^2 are higher, and when R^2 is 1 the points are exactly on the regression line. When the value of R^2 is small, the data points will show considerable scatter around the regression line, and the data points that are represented by the regression line determined will be very few.

The value of the regression coefficient corresponding to the data value b is represented by β. The null hypothesis is used to determine if the Y values are linearly independent of the X values in the dataset, and thus if β = 0. The test normally used is the t test, which assumes that the null hypothesis is true.

To apply the t test, let

$$\frac{}{\sqrt{\frac{}{\sum[\quad - \quad -]}}} \qquad (2.4)$$

A table of the t distribution confidence interval can be used to determine the extent to which the two distributions differ from each other. For any t test there is a corresponding F test, such that

(2.5)

$$\frac{}{\frac{}{\sum[\quad - \quad -]}}$$

The P test assesses the consistency of the null hypothesis. The smaller the P value, the more evidence there is that the null hypothesis is false. The higher the P value, the more evidence of the null hypothesis being true. A P value of 0.05 means that there is a 5% chance of observing a difference as large as observed, even if the sets of data are identical.

Multiple linear regression is an extension of the above linear equation (2.1-2.5) to include multiple variables as the independent variables (input). Multiple regression attempts to model the relation between two or more independent variables (input)

and the dependent variable (output), by fitting the linear equation to the observed data.

The observed data are:

(2.6)

And the model equation is (also termed as the regression line for k variables)

(2.7)

σ

This line describes how Y_i changes in response to the input X_i variables.

Using the least squares method, the best fitting line (plane) from the data is calculated by minimising the sum of the squared residuals (**ε**).

The fitted values take the equation

(2.8)

The residual is then the difference between the observed values and the fitted value.

(2.9)

The least squares method chooses b_0, b_1, ……., b_k to minimise the sum of the residuals.

The next step is to minimise $\sum$, which is also known as the residual sum of squares.

The variance can be estimated by the equation

$$\text{———} \sum \quad = \sigma \tag{2.10}$$

This is also known as the mean square error (MSE).

The observed values are:

(2.11)

The observed values may vary about their mean Y_i and are assumed to have the same standard deviation σ.

The fitted values b_0, b_1, ….., b_k estimate the parameters β_0, β_1, ….., β_k of the regression line.

Multiple regression has three primary uses:

Understanding which input variables make the greatest contribution towards the output.
Knowing the direction of the effect, e.g. increasing X_1 increases/decreases Y.

Using the simulated model to predict future values of the input variables when only the output variables are known.

Multiple regression is a technique going back to Professor Karl Pearson in 1908. Professor Pearson is considered to be one of the founders of modern statistics.

To date it has been shown that multiple regression can establish that a set of independent variables contributes to the variance of a dependent variable to a significant degree (Edwards, 1979:39). The significance of this contribution can be tested rigorously using the R^2 value. The importance of the contribution can be tested further by examining the beta weights attributed to each of the contributing and non-contributing variables. The regression coefficient is then calculated as:

$$\_ \quad \_ \quad \_ \quad \_ \tag{2.12}$$

From the multiple regression equation it is possible to ascertain b_0 and b_1, b_2,…, b_k, which will result in the highest possible positive correlation between the observed value Y' and the predicted value Y. When this is done, the resulting correlation coefficient will be $R_{y.123....k}$.

As the number of X variables increases, the calculations become increasingly complex.

(2.13)

The values b_1 and b_2 can then be calculated as follows:

$$\frac{\sum \quad \sum \quad \sum \quad \sum}{\sum \quad \sum \quad \sum} \tag{2.14}$$

Following a similar procedure for b_2:

$$\frac{\sum \quad \sum \quad \sum \quad \sum}{\sum \quad \sum \quad \sum} \tag{2.15}$$

The general term b_k must satisfy the following equation to minimise the mean square error:

$$\sum \quad \sum \quad \sum \quad \sum \tag{2.16}$$

The square of the multiple correlation coefficient will be given by

(2.17)

SS_{reg}: sum of squares for linear regression

SS_{tot}: total sum of squares

SS_{res}: residual sum of squares

SS_{tot} is then defined by (2.18)

$$\sum \qquad \sum \qquad \sum \tag{2.19}$$

$$\sum \tag{2.20}$$

Thus, the $\mathbf{R_{YY'}}^2$ then is a combination of the above equations and becomes:

$$\frac{\sum \qquad \sum \qquad \sum}{\left[\sum \qquad \sum \qquad \sum \right] \sum} \tag{2.21}$$

$\mathbf{R_{YY'}}^2$ can also be calculated from the partial correlation coefficients of say X_1 and Y, X_2 and Y generally X_i and Y. In the equation below, r_{y1}, r_{y2},…, r_{yk} stand for the semi-partial correlations for X_1 and Y , X_2 and Y,…., X_k and Y.

(2.22)

The argument is that if any of the variables X_1, X_2,…….., X_k are not contributing significantly to the output, then dropping these variables will not alter the correlation coefficient R much. Notice that r_{y1}, r_{y2},…, r_{yk} are obtained as if there were single variable models consisting of one input and one output variable.

The significance for $\mathbf{R_{YY'}}^2$ is tested using the null hypothesis (H_0). The t-test checks if $\beta_1 = \beta_2 =$ …….. $\beta_k = 0$. To test if the null hypothesis is true and $\beta_1 = \beta_2 =$ …….. $\beta_k = 0$, it is necessary to use the F-test:

_____ (2.23)

If the null hypothesis is rejected, then $\beta_1 = \beta_2 =$ …….. $\beta_k \neq 0$

A large value of F indicates that the null hypothesis is not true, while small values are consistent with the null hypothesis. The F value can be any value between zero and infinity. Under the null hypothesis, F is expected to be small positive numbers.

The P value in multiple regression represents a decreasing index of the reliability of a result (Brownlee, 1967: 570). The higher the P value, the less representative the sample data is of the entire set of data, and therefore it is not possible to take the observed relation between variables in the dataset to be reliable indicators of the relationship of the entire dataset.

The P value is calculated in numerous ways, depending on the sampling distribution. It is determined by standardising, with the calculation of two or more sample test statistic.

——————————————————————— (2.24)

Once the test results are obtained, t is compared to the appropriate sampling distribution.

2.2 NEURAL NETWORKS

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the value of the connections (weights) between the elements (Demuth & Beale, 2004).

Neural networks are trained so that a particular input leads to a specific target output. The diagram below illustrates the mechanics of neural networks. An input is presented to the network, there is a comparison between the output and the target, and the network weights are adjusted on the basis of this comparison until the network output matches the target. The target and output pairs are crucial to the training of the network and, typically, many such pairs are used in what is termed as ‘supervised learning’ in training the network.

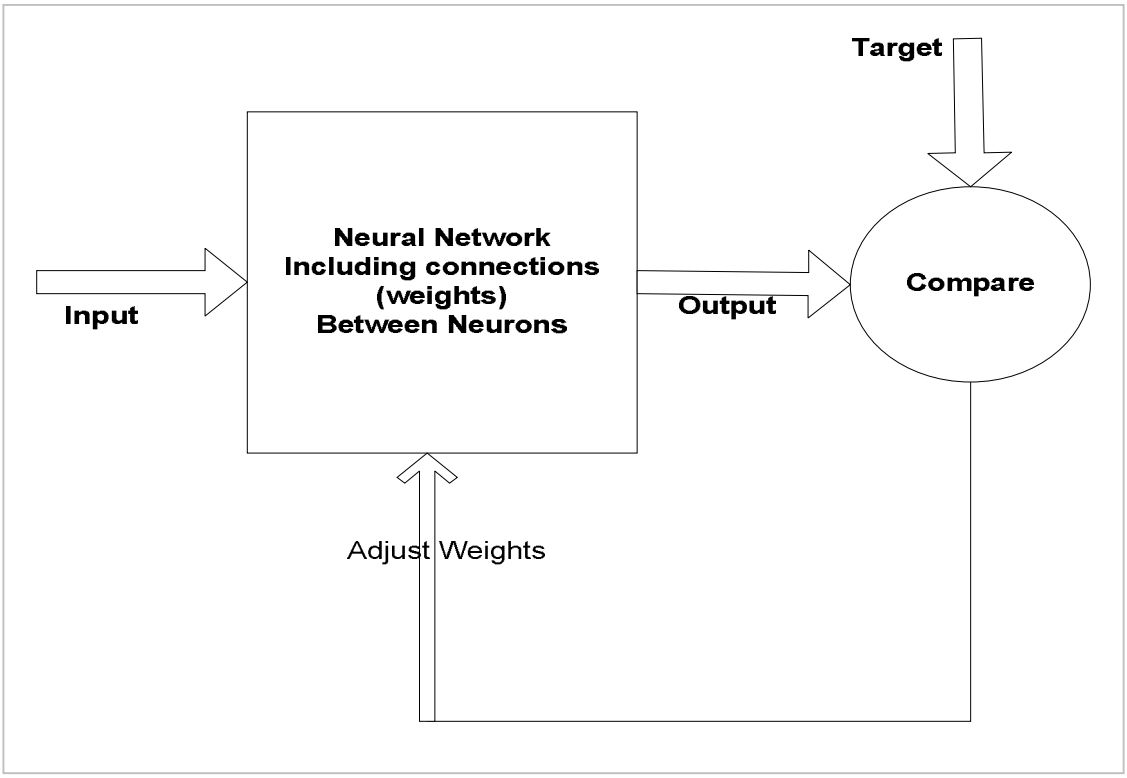

Figure 2.1: Basic neural network (Demuth & Beale, 2004)

2.2.1 Types of neural networks

Neural networks comprise interconnected neurons. There are two general categorisations of neural networks: structural categorisation and learning algorithm categorisation (Pham, 1995:1)

The structure of the neural networks is determined by how the inter-neuron connections are arranged and by the nature of the connections. The structural categorisation is further subdivided into feedforward networks and recurrent networks.

The feedforward network neurons are grouped into layers. Signals flow from the input layer to the output layer in one direction through connections. Thus neurons are connected from one layer to the next, but not within the same layer. Examples of feedforward network are: multilayer perceptron (MLP) networks, learning vector quantisation (LVQ) networks, and group method of data handling (GMDH) networks. The multilayer perceptron network is probably the best known type of feedforward network. The MLP generally has three layers: an input layer, an output layer and an intermediate or hidden layer. With regard to the output of a feedforward network, it is

important to note that, at a given instant, the output is a function only of the input at that instant.

The recurrent networks, however, have a dynamic memory, so that the outputs at a given instant reproduce the current input as well as previous inputs and outputs.

The learning algorithm is how the strengths of the connections are adjusted and trained to achieve a desired result. This category is further subdivided into the supervised learning algorithm, the unsupervised learning algorithm, and the reinforcement learning algorithm. The supervised learning algorithm adjusts the strengths (weights) of the inter-neuron connections according to the difference between the target output and the given input. The supervised learning requires a “teacher” to provide desired target output signals. Examples of supervised learning algorithms are the delta rule, the backpropagation algorithm (generalised delta rule), and the learning vector quantisation algorithm (LVQ).

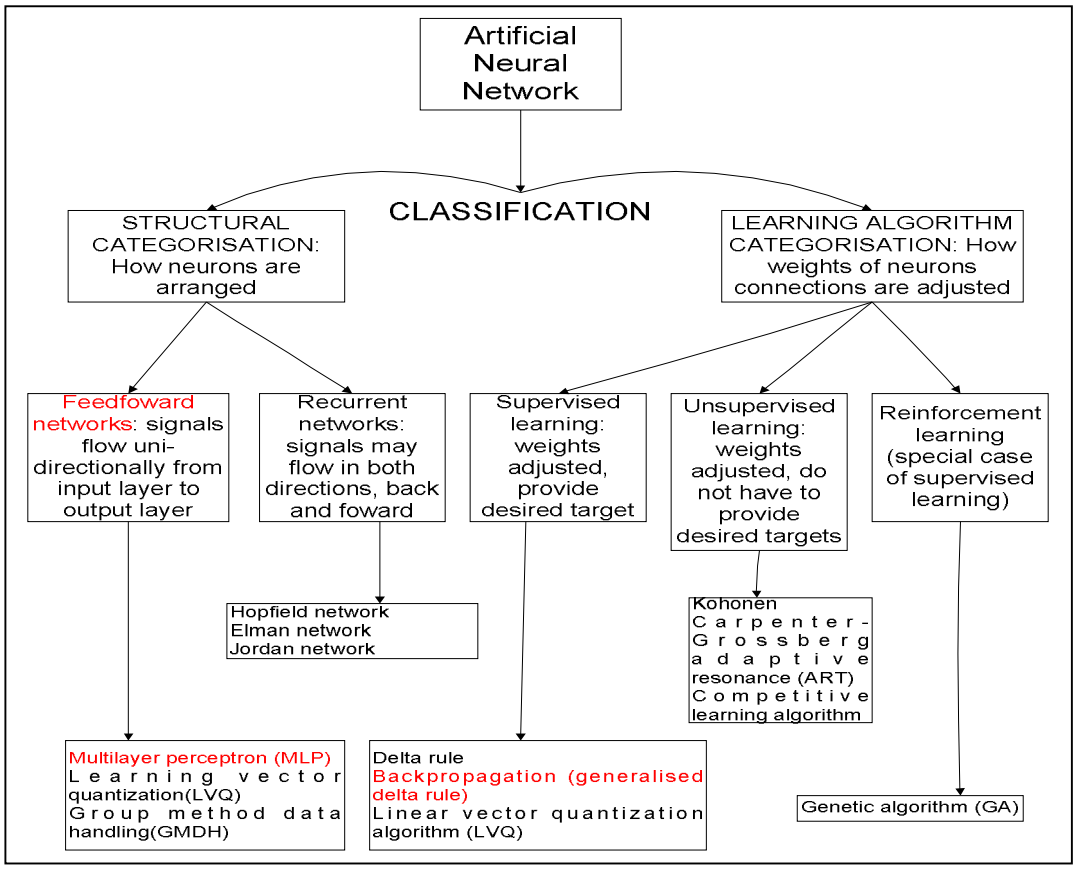

Figure 2.2: ANN classification based on structure and learning algorithm

The multi-layer perceptron (MLP) Networks is the best known of the feedforward networks. The MLP has three layers: an input layer, a hidden layer and an output layer. The neurons in the input layer act as a holding area in order to distribute the input signals to neurons in the hidden layer; thus the hidden layer conventionally is not counted as a layer.

Each neuron, j, in the **hidden layer** acts as a sum up individual input signals xi after weighting the strengths of their connections w_{ji} from the input layer, while computing individual output y_j as a function f.

$$\sum \qquad (2.25)$$

f is a differentiable function that can be a sigmoidal or hyperbolic tangent or radial function.

The output of the neurons in the **output layer** is computed similarly.

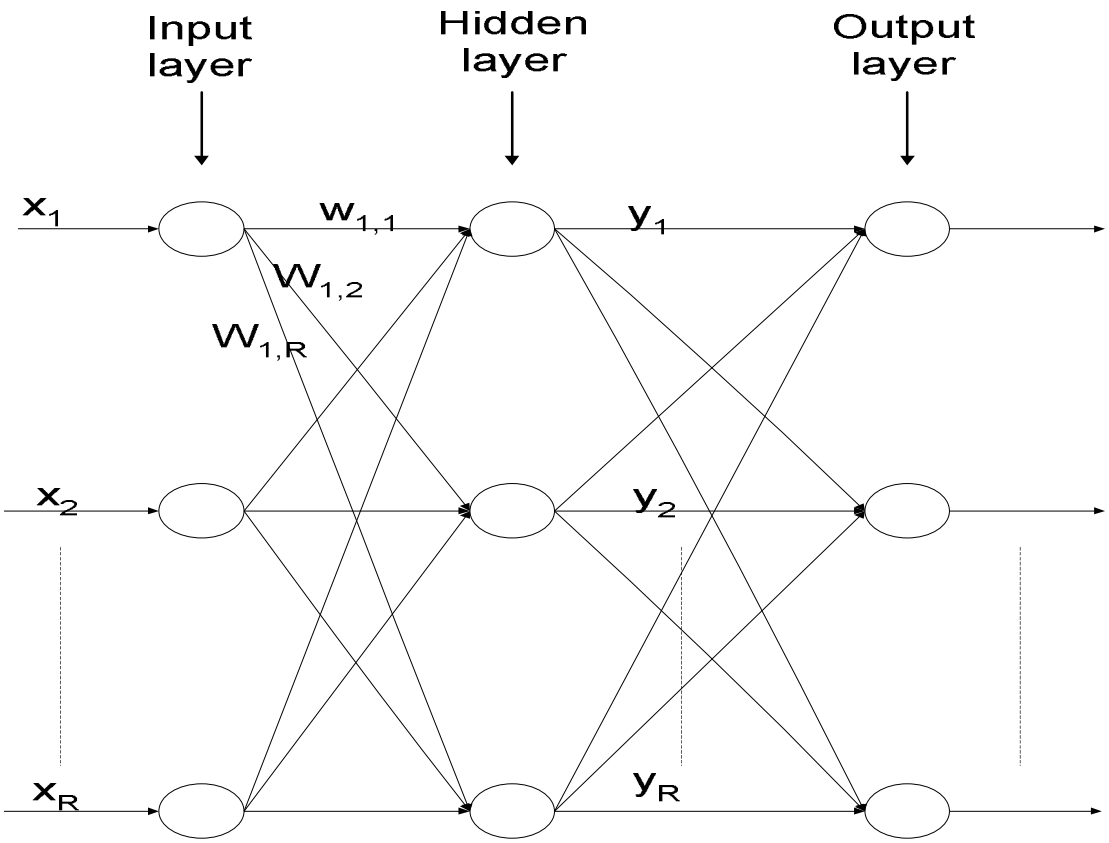

Figure 2.3: ANN general architecture with hidden layer (Pham, 1995).

2.2.2 Historical background

There are at least two ingredients that are essential for the successful advancement of a technology: conceptualisation and implementation (Hagan et al., 1995:89). The development of neural networks has progressed through conceptualisation and implementation, but this happened in fits and false starts rather than as a continuous evolution. The table below offers a summary of the progress in artificial neural networks over the years.

Table 2.1: Historical development of ANN

RESEARCHERS	YEAR	FIELD CONTRIBUTED
Warren McCulloch Walter Pitts	1943	Showed that ANN could, in principle, be used to compute any arithmetical or logical function.
Donald Hebb	1949	Proposed that the classical condition (discovered by Pavlov) is present because of the properties of individual neurons. He then proposed a mechanism for learning in biological neurons.

Frank Rosenblatt	1958	First practical application of ANN. The invention of the perceptron network and associated learning rule. Demonstration that the perceptrons networks had ability to perform pattern recognition. Limitation: can only solve a limited class of problems.
Benard Widrow Tedd Hoff	1960	Introduced the Widrow-Hoff learning rule, a new learning algorithm. It was used to train adaptive linear neural networks and was quite similar in structure to Rosenblatt's perceptron. Limitations: can only solve a limited class of problems.
Marvin Minsky Seymore Papert	1969	Widely publicised the shortcomings of the proposed networks. This report, combined with the lack of powerful digital computers, led to a lull in ANN research, as many people believed there was no future in ANN.
Teuvo Kohonen James Anderson	1972	They independently and separately developed ANN that could act as memory.
Paul Werbos	1974	Started work on a supervised learning technique used for ANN, groundwork for backpropagation.
Stephen Grossberg	1976	Self-organising networks.
John Hopfield	1982	The impediments presented by the lack of powerful computers were removed, thus great developments were possible in the field of ANN. He proposed the use of statistical mechanics to explain the operation of certain classes of recurrent networks; associative memory.
David Rumelhart James McClelland	1986	Most influential publication on backpropagation algorithm for training multilayer perceptron networks. This answered Minsky and Papert criticism in the 1960s.

The advances in neural networks have been propelled largely by the availability of powerful new computers and new concepts, such as innovative architecture and training rules. Neural networks seem to have taken a permanent place as an engineering tool, not just as a solution to any problem, but as a tool to be used in appropriate situations.

There are a large number of applications of neural networks and the list seems to grow every day.

Aerospace

Automative

Banking

Defence

Electronics

Entertainment

Financial

Insurance

Manufacturing

Medical

Oil and gas exploration

Robotics

Speech

Securities

Telecommunication

Transportation

The application of ANN to chemical engineering is a wide and varied field. Here are examples of implementation and research:

Bioprocess engineering: application of feedforward neural networks for system identification of a biochemical process
Use of ANN in modelling of chemical processes

Polymers, pulp and paper industry: modelling and optimisation of pulp and paper processing using neural networks

Separation process: prediction of product quality parameters of a crude fractionation section of an oil refinery using neural networks; ANN has also been used extensively in the study of the crystallisation process and its dynamics

Heat and mass transfer: a neural network approach to nonlinear Identification and control of a heat exchanger

Thermodynamics: evaluation of the thermodynamic models *Uniquac* and *Unifac* using artificial neural networks

Fluid dynamics: flow regime identification in air-water two-phase flow using neural networks – process control and design: closed loop nonlinear process identification using internally recurrent neural nets

Catalysis: the application of neural networks in the development of an online model for a semi-regenerative catalytic reformer

Chemical reactor design: use of neural networks for LPCVD reactor modelling

Process control and design: use of artificial neural networks to monitor faults and for troubleshooting in the process industries

Unit operations of chemical engineering: HETP and pressure drop prediction for structured packing distillation columns using a neural network model

Environmental engineering: modelling of activated sludge waste water treatment processes using integrated neural networks and a first principle model

Transport phenomena: modelling of unsteady heat conduction fields by using composite recurrent neural networks

These are among many examples of the applications of neural networks in chemical engineering. The range of interests and applications continues to grow every day in chemical engineering, as well as in all other fields.

There are a number of advantages and disadvantages related to the use of neural networks.

Advantages

They have the ability to represent complex linear and nonlinear relationships; they learn this relationships from modelled data.
There is no need to know the data relationships before building a neural network.

Neural nets are general, thus they can handle problems with diverse parameters and are able to classify a very complex distribution.

Simple elements operating in parallel permit solutions to problems where multiple constraints can be satisfied simultaneously.

Graceful degradation and final presentation of results.

Rules are implicit rather than explicit, thus eliminating the need for the user to formulate the rules.

Disadvantages

Neural networks take training data and generate opaque, complex models, thus it is very difficult to determine how the net is making its decision.

Because the opaque complex models are very diverse, no one network of the same problem is the same.

The trained data may sometimes contain what are termed ‘chance effects’, but this is intrinsically built into the model where else this ‘chance’ may indeed never occur again e.g. Zuma announces his presidential candidacy, the stock market shifts.

Neural networks are unable to manage imprecise or vague information.

They are unable to handle linguistic information and thus unable to combine numeric data with linguistic data.

There is heavy reliance on trial and error to determine the number of layers, number of nodes, transfer functions and training functions.

2.2.3 Neural network feedforward backpropagation

Backpropagation is also known as the backpropagation of errors, and was created by generalising the Widrow-Hoff learning rule, which is a gradient descent algorithm (Rumelhart & McClelland, 1986). The gradient descent algorithm is an optimisation algorithm to find a local minimum. The algorithm takes steps proportional to the negative of the gradient and thus the network weights are adjusted along the negative of the gradient. Backpropagation is used for feedforward networks that have no feedback or networks that do not have connections that loop.

Backpropagation requires that transfer functions used in the hidden layer are differentiable, and backpropagation networks usually have multiple layers. The errors are said to propagate backwards from the outer nodes to the inner nodes, thus calculating the gradient of the error of the network with respect to the adjusted weights of the network. Although the standard backpropagation is a gradient descent algorithm, there are a number of variations on the basic algorithm, based on different methods of optimisation, such as the conjugate gradient and the Newton method.

The input vectors and corresponding output vectors are presented to a chosen multi-layered nonlinear network and are trained until the network can approximate a function that associates a specific input with a specific output. Properly trained backpropagation networks are capable of approximating any function and, once trained, give reasonable answers if they are presented with inputs they have never seen.

Neurons use any differentiable function to generate their output, but the commonly used transfer function is logsigmoid and tansigmoid. The transfer function calculates a layer's output from the net input.

$$\sum \qquad (2.26)$$

Feedforward networks have one or more layers of sigmoid neurons, followed by an output layer of linear neurons. The multiple layers of nonlinear transfer functions allow the network to train nonlinear and linear relations between the input and output. The linear output layer produces outputs of values between the range of 1 and -1.

The logsigmoid, tansigmoid and purelin transfer functions are shown below:

Purelin

Purelin is a transfer function that calculates the output of a layer from its net input. Since it is a linear transfer function, it simply returns the value passed to it according to the expression:

f(n) = n (2.27)

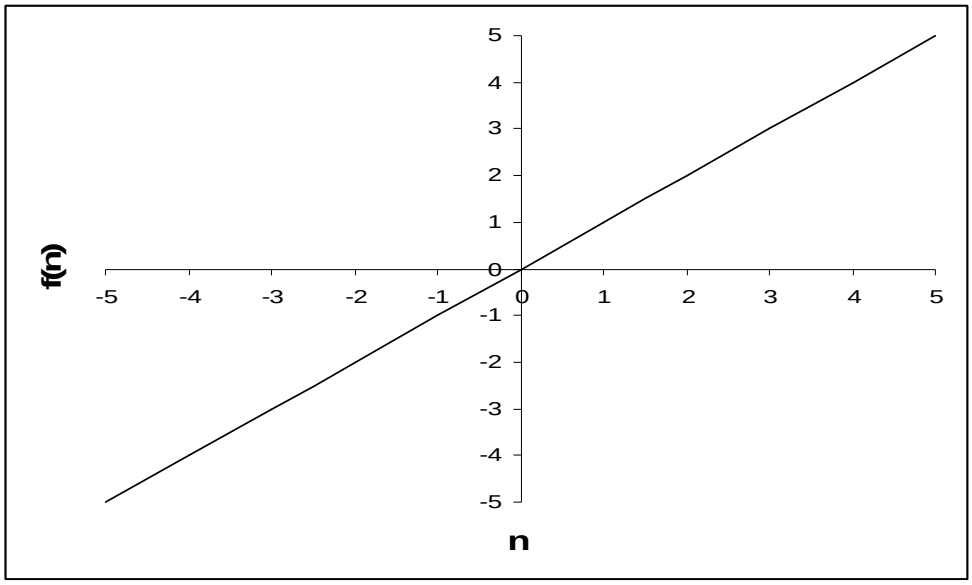

Figure 2.4: Purelin transfer function with f(n) = purelin (n) = purelin(Wp+b)

This transfer function is important, as it finds linear approximation to nonlinear functions. However, this does not mean that a linear network can be made to perform a nonlinear computation.

Nonlinear transfer function

Tansigmoid

Tansigmoid is a transfer function that calculates the output of a layer from its net input, which can range from plus to minus infinity. It takes input and returns outputs squashed between -1 and 1, where f is an exponential function in the format:

(2.28)

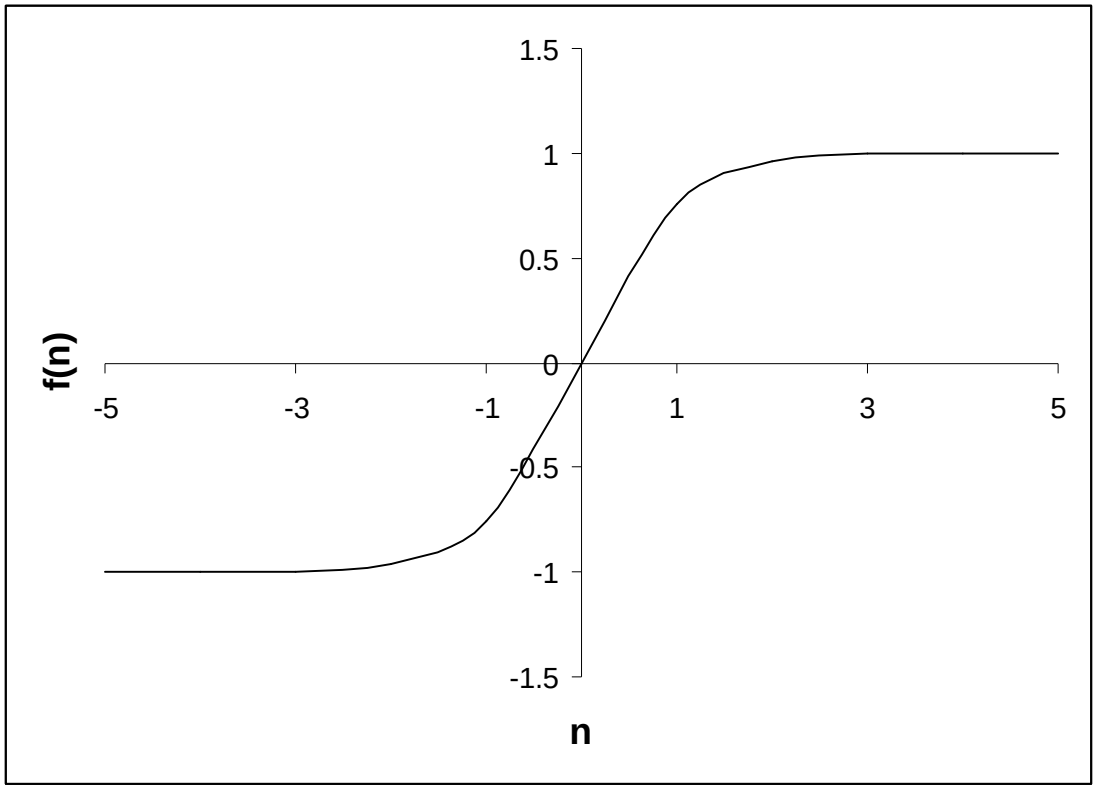

Figure 2.5: Tansigmoid transfer function with f(n) = tansig(n) = tansig (Wp+b)

Logsigmoid

Logsigmoid is a transfer function that calculates the output of a layer from its net input, which can range between plus and minus infinity. It takes input and returns each element of N squashed between 0 and 1 according to the expression:

(2.29)

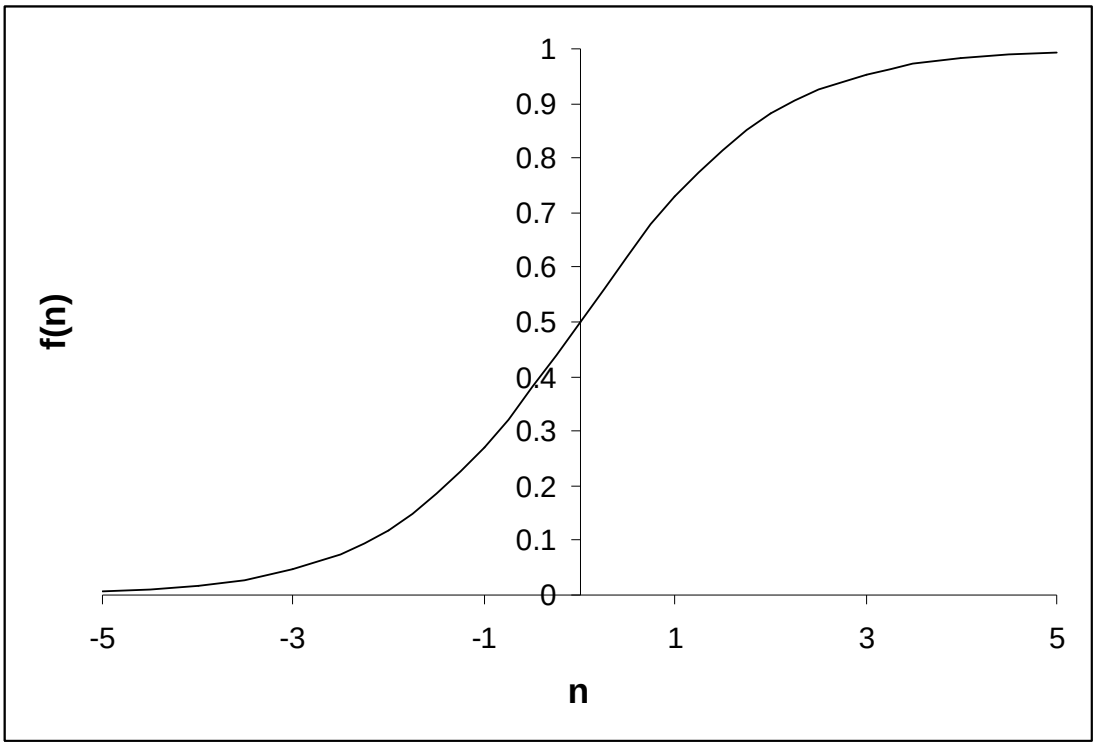

Figure 2.6: Logsigmoid transfer function with f(n)= logsig(n) = logsig(Wp+b)

2.2.4 Summary of the backpropagation algorithm

The backpropagation algorithm learns to recognise and reproduce patterns iteratively, where weights are adjusted in order to minimise error with a pre-defined selected criterion. The four steps of the algorithm are summarised below:

2.1.1.1 Initialise weights or start from random weight state

2.1.1.2 The output of the network is evaluated (forward pass)

The purpose of the nodes in the **input layer** is to transmit the received single external input to all nodes of the next layer.

The output of the k^{th} input of the p^{th} example is shown in Figure 2.7.

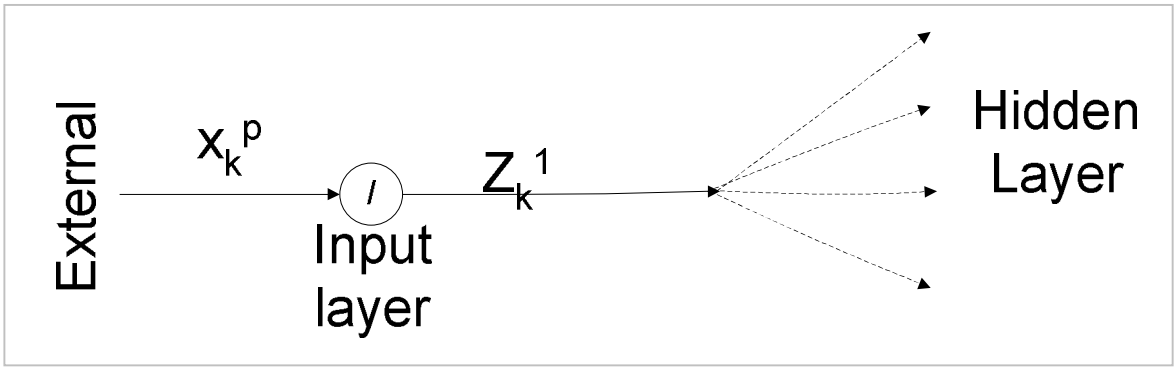

Figure 2.7: Input layer in the forward pass (Tsaptsinos, 1995)

This can be written as:

(2.30)

The function of any other node is to receive inputs, sum the weighted inputs plus the bias, nonlinearly/linearly convert the sums and transmit the outputs to the nodes in the next layer or to the external environment.

O - represents an input, hidden and output node

W - represents the bias, default value = 1

- represents the weights

$\rightarrow$ - represents the unidirectional flow placed on the connections

The output of the l^{th} **hidden layer** is shown graphically in Figure 2.8.

This is written as

$$l \qquad l \qquad l \quad K \qquad l \quad l \tag{2.31}$$

$$l = \sum_k \qquad l + \quad 1 \; l \tag{2.32}$$

The nonlinear transformation using the transfer function is represented as follows:

$$l \tag{2.33}$$

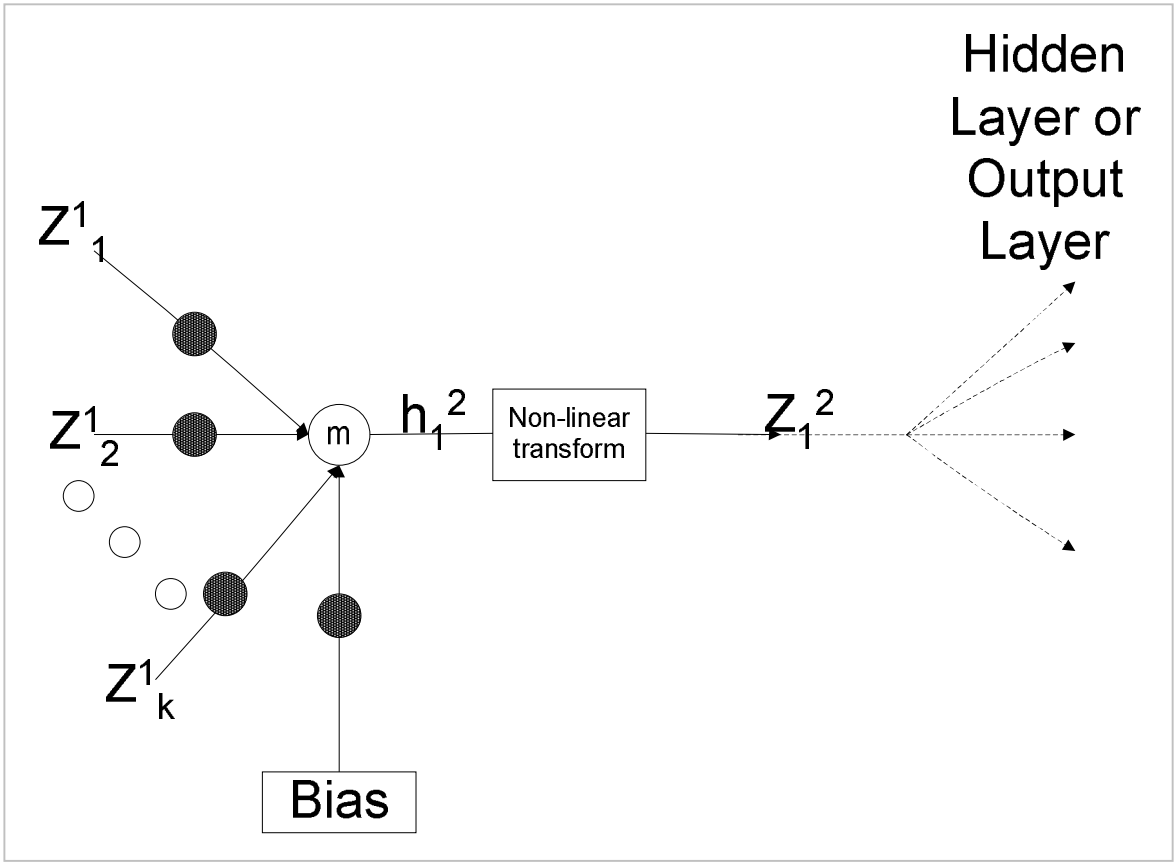

Figure 2.8: Hidden layer in the forward pass (Tsaptsinos, 1995)

The equations 2 to 4 represent one hidden layer, but for more than one hidden layer the equations are similarly derived, although for different numbers of layers.

The **output layer** of the m^{th} output node is shown graphically in Figure 2.8.

It can be written as

(2.34)

Σ

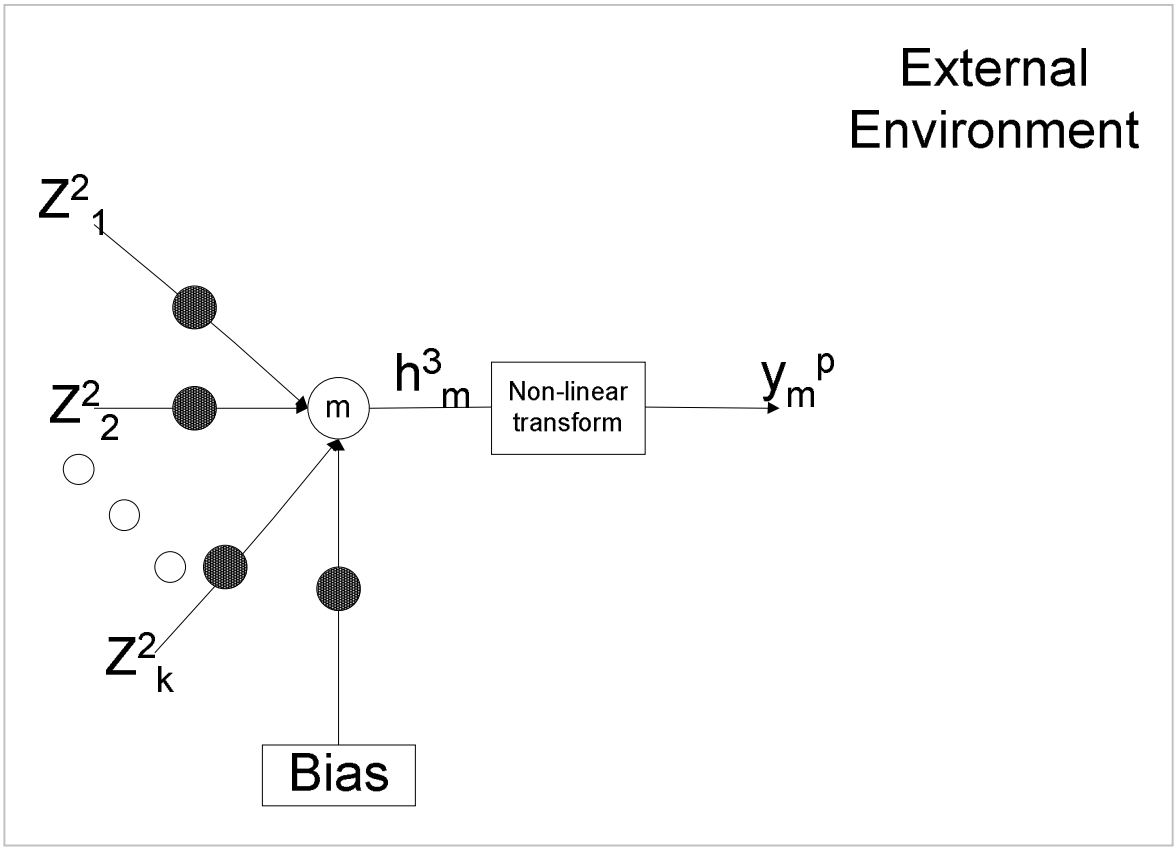

Figure 2.9: Output layer in the forward pass (Tsaptsinos, 1995)

Then

(2.35)

2.1.1.3 The error is calculated (error criterion)

The error criterion E measures the difference between the actual outputs of the output node of the network ($\mathbf{y^p_m}$) and the desired outputs ( ).

This is given by:

$\sum$ (2.36)

The total error for all can be summed up by the following equation:

$\sum$ (2.37)

A tolerance level t can be determined for the outputs:

If the required output is zero, any network output below t is considered correct. If the required output is one, any network output greater than 1-t is considered correct. If there is a continuous output, the generated output is considered correct if it lies within $\pm t$ of the desired value.

2.1.1.4 The weights are adjusted (backward pass)

In order to minimise E, which depends on the weights and biases, the backpropagation uses gradient steepest descent in order to locate appropriate weights and biases. The gradient descent algorithm works in such a way that it changes each weight by an amount proportional to the gradient of the error criterion at the present weight location.

The backward pass is symbolised by the following equation:

(2.38)

The gradient can be found using the following equation:

——— (2.39)

Where n determines the step size the descent takes and is negative, as the aim is to decrease the error.

The generalised delta rule can be summarised after much derivation and is given by:

(2.40)

n: learning rate

: “delta term” representing the error

x: represents the incoming inputs to a particular node

Weights on the connection between the hidden layer and the output layer

( )

These weights are adjusted with the following summarised equations. The derivations are left out with generic node and weight one at a time, thus summation is dropped:

——— (2.41)

y^p_m is the output of the output layer node; then

———

(2.42)

———

The gradient is then given as:

(2.43)

Weights on the connection between the input layer and the hidden layer

The weights are adjusted for the backpass with the following summarised equations. There is one generic node and weight at a time, and the summation is therefore dropped for the purposes of simplification:

——— (2.44)

is effectively the input of the first hidden layer.

—— ∑ ∑ ∑

(2.45)

Simplifying the equation above,

—— ∑ (2.46)

The gradient is then:

∑ (2.47)

Where the delta (∂) term of this layer is represented by:

∑ (2.48)

Notice the delta term for the input hidden layer uses the delta term for the hidden output layer, as they are interconnected.

The new weights for the input hidden layer and the hidden output layer are calculated by means of the following two equations:

∑ (2.49)

∑∑ (2.50)

The summary of these steps, namely forward pass, error criterion and backward pass, is the following:

1. Initialise all weights

Present the input variables to the nodes in the input layer.
Calculate the output of every input variable with the following equation

(1a)

Calculate the output of every hidden node using the transfer function chosen; in this case a logsigmoid function was used. The output of a hidden layer is a nonlinear transformation of the sum of the product of incoming inputs from the previous layer and the associated weights, plus the bias.

$$\frac{\qquad\qquad}{\sum} \tag{1b}$$

Calculate the output of every output node using the following equation, which also utilises the logsigmoid function as the transfer function. The output of a node from the output layer is a nonlinear transformation of the sum of the product of incoming inputs from the previous layer and the associated weights, plus the bias.

$$\frac{\qquad\qquad}{\sum} \tag{1c}$$

Calculate the error, which is the difference between the target values and the values generated by the network, multiplied by 0.5.

$$\sum \tag{1d}$$

2. Accumulate the total error.

$$\sum \tag{2a}$$

3. If the total error is satisfactory, stop; otherwise continue to the next step.

4. For each layer and for the input/output pair, calculate the following:

Delta values, using the following equation:

(4a)

$$\sum \tag{4b}$$

Calculate the change required for each weight using the following equations; represents the steepness parameter, and is usually set to 0.5 or to 1:

(4c)

(4d)

(4e)

(4f)

For each weight, amass the total change required.

5. Adjust the weights using the following equations:

$$\sum \quad (5a)$$

$$\sum \quad (5b)$$

$$\sum \quad (5c)$$

$$\sum \quad (5d)$$

6. Continue from step 1 until step 3 is satisfactory.

3. RESEARCH METHODOLOGY

3.1 DATA

The data used were generated randomly in Microsoft Excel. The range was set so that every term in the set equation contributed within the set range. After a review of the literature and a comparative study, the range of 1 to 10 was chosen. For a proper comparative study to be done, four equations were set to generate the random values. When the random values were generated, the missing variables were added as dummy variables.

Equation 1 with 5 input variables:

$\frac{}{}$ dummy variables: $\mathbf{X_3, X_4}$

Equation 2 with 5 input variables:

$\sqrt{}$ dummy variables: $\mathbf{X_2, X_4}$

Equation 3 is a combination of equation 1 and equation 2, with 10 input variables:

$\frac{}{}$ $\sqrt{}$

dummy variables: $\mathbf{X_3, X_4, X_7, X_9}$

Equation 4 has seven input variables:

dummy variables: x3, x4, x5

This was done for 1 000 observations. The purpose was to check if these dummy variables inserted in the equation contributed any significant difference to the

regression value and to check if they were detected by the ANN as suspect variables.

3.2 MATLAB REGRESSION

The general equations were used for generating random data of a range between 1 and 10. A number of matrices were constructed for all variables and exported to Matlab.

In general, the number of matrices created for each given equation is directly proportional to the number of input variables.

Number of matrices = number of input variables + 2

The output variable remains constant for each equation, as it is determined by the equation. At first, all input variables are utilised in calculating the regression values, and then one variable is successfully removed from the input while creating new matrices.

All matrices in Matlab for the command regression have to be formatted with a column of ones. The column of ones is for estimating the y intercept of the linear model. The y intercept in the equation below is represented by and it is the only standalone constant.

The command in Matlab:

(3.1)

returns the least squares fit of Yall on Xall by solving the linear model where

(3.2)

(3.3)

for where:

is an n-by-1 vector of output variables

is an n-by-p matrix of input variables

is a p-by-1 vector of parameters

is an n-by-1 vector of random disturbances

[ ] (3.4)

The command above provides an estimate of in b. Since a column of one has been added to the Xall matrix, there is a p-by-1 vector of estimates from the least squares fit. **b** constants, which are estimates of the β_i of the equation

is a p-by-2 vector gives for all datasets (Xall, Yall) with a range in which 95% of the b_i values lie. In other words, 95% of the **β** estimates lie within a classified multiple of the respective standard deviation.

The residuals are defined as

(3.5)

This is returned in **r** and which is a n-by-1 vector

is an n-by-2 vector is a p-by-2 vector gives for all datasets (Xall, Yall) with a range in which 95% of the r_i values lie. In other words, 95% of the residuals lie within a classified multiple of the respective standard deviation.

The vector **stats** contains the R^2 statistic along with the F and p values for the regression.

Equation 1: —— dummy variables x_3, x_4

The first matrix, $\mathbf{X_{all}}$, will contain ones and the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5}$, a 1 000 by 6 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows.

The second matrix, $\mathbf{X_{no1}}$, will contain ones and the input variables $\mathbf{X_2, X_3, X_4, X_5}$, a 1 000 by 5 matrix, while $\mathbf{Y_{no1}}$ will have the output variable, a vector with 1 000 rows.

The third matrix, $\mathbf{X_{no2}}$, will contain ones and the input variables $\mathbf{X_1, X_3, X_4, X_5}$, a 1 000 by 5 matrix, while $\mathbf{Y_{no2}}$ will have the output variable, a vector with 1 000 rows.

The fourth matrix, $\mathbf{X_{no3}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_4}$, $\mathbf{X_5}$, a 1 000 by 5 matrix while $\mathbf{Y_{no3}}$ will have the output variable, a vector with 1 000 rows.

The fifth matrix, $\mathbf{X_{no4}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_5}$, a 1000 by 5 matrix, while $\mathbf{Y_{no4}}$ will have the output variable, a vector with 1 000 rows.

The sixth matrix, $\mathbf{X_{no5}}$, will contain ones with input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, a 1000 by 5 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1 000 rows.

The seventh matrix created removes the dummy variables $\mathbf{X_3}$ and $\mathbf{X_4}$. It is called $\mathbf{X_{no34}}$ and has ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_5}$, a 1000 by 4 matrix. The output variable is $\mathbf{Y_{no34}}$ and will have a vector with 1 000 rows.

Most importantly, the output vector remains constant for comparison, because the dummy variables $\mathbf{X_3}$, and $\mathbf{X_4}$ do not contribute to the output and thus the regression value should not be altered by their presence.

Thus, this set of data has seven matrices in total. These are the matrices that were created in Excel spreadsheets and exported to Matlab for the regression for the first dataset.

Actual commands in Matlab

```
[b,bint,r,rint,stats] = regress(Yall,Xall);
[bno1,bintno1,rno1,rintno1,statsno1] = regress(Yno1,Xno1);
```

```
[bno2,bintno2,rno2,rintno2,statsno2] = regress(Yno2,Xno2);
[bno3,bintno3,rno3,rintno3,statsno3] = regress(Yno3,Xno3);
[bno4,bintno4,rno4,rintno4,statsno4] = regress(Yno4,Xno4);
[bno5,bintno5,rno5,rintno5,statsno5] = regress(Yno5,Xno5);
 [bno34,bintno34,rno34,rint34,stats34] = regress(Yno34,Xno34);
```

STATS = [stats; statsno1; statsno2; statsno3; statsno4; statsno5; statsno34]

B = [bno1 bno2 bno3 bno4 bno5]

Equation 2: $\sqrt{}$ dummy variables x_2, x_4

The first matrix, $\mathbf{X_{all}}$, will contain ones and the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5}$, a 1 000 by 6 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows.

The second matrix, $\mathbf{X_{no1}}$, will contain ones and the input variables $\mathbf{X_2, X_3, X_4, X_5}$, a 1000 by 5 matrix, while $\mathbf{Y_{no1}}$ will have the output variable, a vector with 1000 rows.

The third matrix, $\mathbf{X_{no2}}$, will contain ones and the input variables $\mathbf{X_1, X_3, X_4, X_5}$, a 1000 by 5 matrix, while $\mathbf{Y_{no2}}$ will have the output variable, a vector with 1000 rows.

The fourth matrix, $\mathbf{X_{no3}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_4}$, $\mathbf{X_5}$, a 1000 by 5 matrix, while $\mathbf{Y_{no3}}$ will have the output variable, a vector with 1000 rows.

The fifth matrix, $\mathbf{X_{no4}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_5}$, a 1000 by 5 matrix, while $\mathbf{Y_{no4}}$ will have the output variable, a vector with 1000 rows.

The sixth matrix, $\mathbf{X_{no5}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, with 1000 by 5 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The seventh matrix removes the dummy variables $\mathbf{X_2}$ and $\mathbf{X_4}$, is called $\mathbf{X_{no24}}$ and has ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_5}$, a 1000 by 4 matrix. The output variable is $\mathbf{Y_{no24}}$ and will have a vector with 1000 rows.

Most importantly, the output vector remains constant for comparison, because the dummy variables $\mathbf{X_2}$, and $\mathbf{X_4}$ do not contribute to the output and thus the regression value should not be altered by their presence.

Thus, this set of data has seven matrices in total. These are the matrices that were created in Excel spreadsheets and exported to Matlab for the regression for the second dataset.

Actual commands in Matlab

```
[b,bint,r,rint,stats] = regress(Yall,Xall);
[bno1,bintno1,rno1,rintno1,statsno1] = regress(Yno1,Xno1);
[bno2,bintno2,rno2,rintno2,statsno2] = regress(Yno2,Xno2);
[bno3,bintno3,rno3,rintno3,statsno3] = regress(Yno3,Xno3);
[bno4,bintno4,rno4,rintno4,statsno4] = regress(Yno4,Xno4);
[bno5,bintno5,rno5,rintno5,statsno5] = regress(Yno5,Xno5);
 [bno24,bintno24,rno24,rint24,stats24] = regress(Yno24,Xno24);
```

STATS = [stats; statsno1; statsno2; statsno3; statsno4; statsno5; statsno24]

B = [bno1 bno2 bno3 bno4 bno5]

Equation3: $\overline{\qquad}$ $\overline{\qquad\sqrt{\qquad}}$

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_7}$, $\mathbf{x_9}$

The third equation is a combination of the first two equations.

The first matrix, $\mathbf{X_{all}}$, will contain ones and the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 11 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1000 rows.

The second matrix, $\mathbf{X_{no1}}$, will contains ones and the input variables $\mathbf{X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no1}}$ will have the output variable, a vector with 1000 rows.

The third matrix, $\mathbf{X_{no2}}$, will contain ones and with input variables $\mathbf{X_1, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no2}}$ will have the output variable, a vector with 1000 rows.

The fourth matrix, $\mathbf{X_{no3}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_4}$, $\mathbf{X_5}$, $\mathbf{X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no3}}$ will have the output variable, a vector with 1000 rows.

The fifth matrix, $\mathbf{X_{no4}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_5}$, $\mathbf{X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no4}}$ will have the output variable, a vector with 1000 rows.

The sixth matrix, $\mathbf{X_{no5}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_6, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The seventh matrix, $\mathbf{X_{no6}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_7, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The eighth matrix, $\mathbf{X_{no7}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_6, X_8, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The ninth matrix, $\mathbf{X_{no8}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_6, X_7, X_9, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The tenth matrix , $\mathbf{X_{no9}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_6, X_7, X_8, X_{10}}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The eleventh matrix, $\mathbf{X_{no10}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_6, X_7, X_8, X_9}$, a 1000 by 10 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The twelfth matrix, $\mathbf{X_{no3479}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_4}$, $\mathbf{X_5}$, $\mathbf{X_6}$, $\mathbf{X_8}$, $\mathbf{X_{10}}$, a 1000 by 8 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

Thus, this set of data has 12 matrices in total. These are the matrices that were created in Excel spreadsheets and exported to Matlab for the regression for the third dataset.

Actual commands in Matlab

```
[b,bint,r,rint,stats] = regress(Yall,Xall);
[bno1,bintno1,rno1,rintno1,statsno1] = regress(Yno1,Xno1);
[bno2,bintno2,rno2,rintno2,statsno2] = regress(Yno2,Xno2);
[bno3,bintno3,rno3,rintno3,statsno3] = regress(Yno3,Xno3);
[bno4,bintno4,rno4,rintno4,statsno4] = regress(Yno4,Xno4);
[bno5,bintno5,rno5,rintno5,statsno5] = regress(Yno5,Xno5);
[bno6,bintno6,rno6,rintno6,statsno6] = regress(Yno6,Xno6);
[bno7,bintno7,rno7,rintno7,statsno7] = regress(Yno7,Xno7);
[bno8,bintno8,rno8,rintno8,statsno8] = regress(Yno8,Xno8);
[bno9,bintno9,rno9,rintno9,statsno9] = regress(Yno9,Xno9);
[bno10,bintno10,rno10,rintno10,statsno10] = regress(Yno10,Xno10);
[bno3479,bintno3479,rno3479,rintno3479,statsno3479]=regress(Yno3479,Xn
o3479);
```

STATS = [stats; statsno1; statsno2; statsno3; statsno4; statsno5; statsno6; statsno7; statsno8; statsno9; statsno10; statsno3479]

B = [bno1 bno2 bno3 bno4 bno5 bno6 bno7 bno8 bno9 bno10]

Equation 4:

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$

The first matrix, $\mathbf{X_{all}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5}$, $\mathbf{X_6}$, $\mathbf{X_7}$, a 1000 by 8 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1000 rows.

The second matrix, $\mathbf{X_{no1}}$, will contain ones and the input variables $\mathbf{X_2, X_3, X_4, X_5, X_6, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no1}}$ will have the output variable, a vector with 1000 rows.

The third matrix, $\mathbf{X_{no2}}$, will contain ones and the input variables $\mathbf{X_1, X_3, X_4, X_5, X_6, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no2}}$ will have the output variable, a vector with 1000 rows.

The fourth matrix, $\mathbf{X_{no3}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_4}$, $\mathbf{X_5}$, $\mathbf{X_6, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no3}}$ will have the output variable, a vector with 1000 rows.

The fifth matrix, $\mathbf{X_{no4}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_5}$, $\mathbf{X_6, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no4}}$ will have the output variable, a vector with 1000 rows.

The sixth matrix, $\mathbf{X_{no5}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_6, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The seventh matrix, $\mathbf{X_{no6}}$, will contain ones and the input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_7}$, a 1000 by 7 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, a vector with 1000 rows.

The eighth matrix, $\mathbf{X_{no7}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, $\mathbf{X_4}$, $\mathbf{X_5, X_6}$, a 1000 by 7 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, vector with 1000 rows.

The ninth matrix, $\mathbf{X_{no345}}$, will contain ones and input variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_6, X_7}$, a 1000 by 5 matrix, while $\mathbf{Y_{no5}}$ will have the output variable, vector with 1000 rows.

Thus, this set of data has nine matrices in total. These are the matrices that were created in Excel spreadsheets and exported to Matlab for the regression for the fourth dataset.

Actual commands in Matlab

```
[b,bint,r,rint,stats] = regress(Yall,Xall);
[bno1,bintno1,rno1,rintno1,statsno1] = regress(Yno1,Xno1);
[bno2,bintno2,rno2,rintno2,statsno2] = regress(Yno2,Xno2);
[bno3,bintno3,rno3,rintno3,statsno3] = regress(Yno3,Xno3);
[bno4,bintno4,rno4,rintno4,statsno4] = regress(Yno4,Xno4);
```

```
[bno5,bintno5,rno5,rintno5,statsno5] = regress(Yno5,Xno5);
[bno6,bintno6,rno6,rintno6,statsno6] = regress(Yno6,Xno6);
[bno7,bintno7,rno7,rintno7,statsno7]=regress(Yno7,Xno7);
[bno345,bintno345,rno345,rintno345,statsno345]=regress(Yno345,Xno345);
```

STATS = [stats; statsno1; statsno2; statsno3; statsno4; statsno5; statsno6; statsno7; statsno345]

B = [bno1 bno2 bno3 bno4 bno5 bno6 bno7]

3.3 NEURAL NETWORK

3.3.1 Backpropagation Algorithm

There are many variations of the backpropagation algorithm, depending on the different methods of optimisation. The simplest is the generalisation of the steepest descent method (Rumelhart & McClelland, 1986) applied to a feedforward neural network. The network learning updates the weights and biases in the negative direction in which the performance function decreases most rapidly. The architecture that is commonly used for the feedforward backpropagation algorithm is a multilayer network as represented in Figure 10 below.

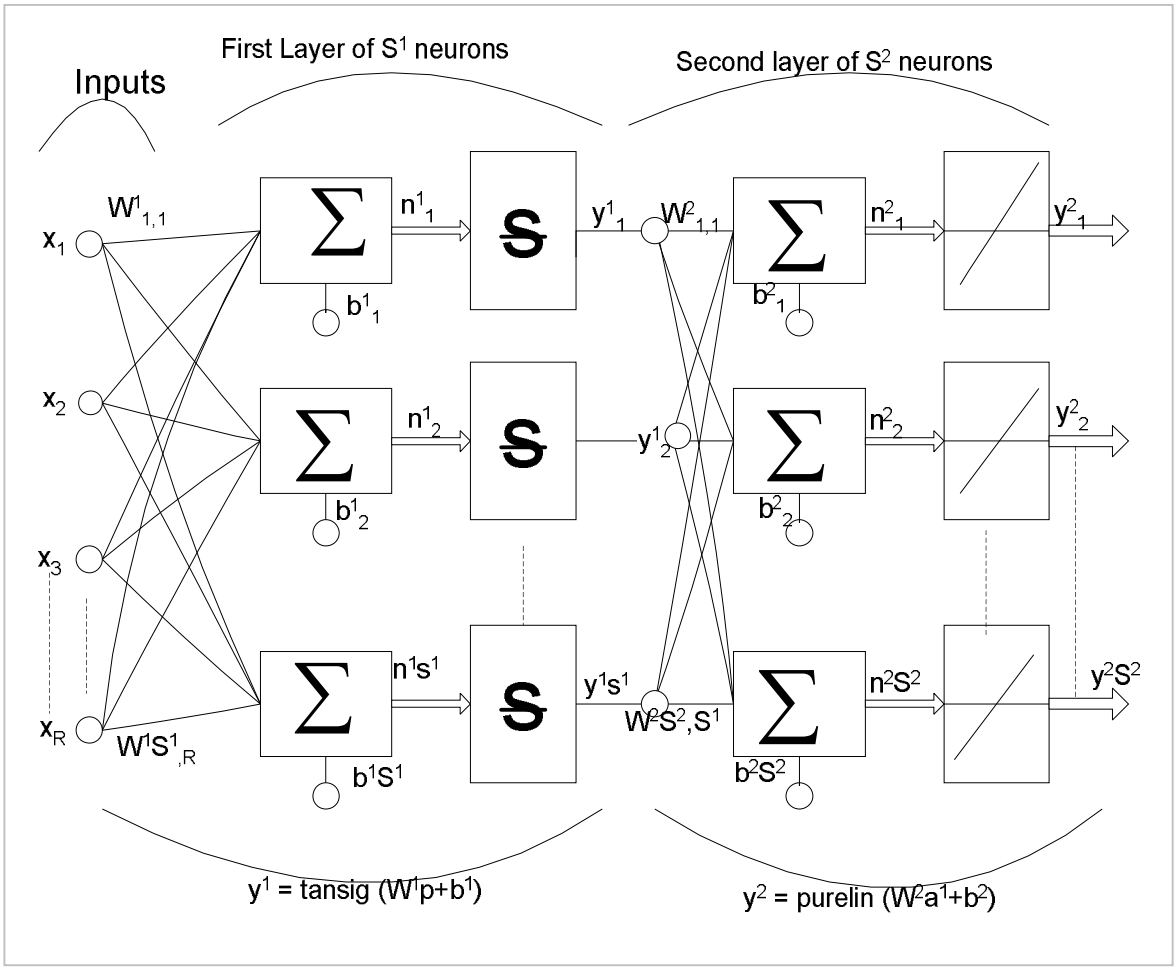

Figure 3.1: Structure of a feedforward neural network with two transfer functions

The neural network model is made up of three layers: input, hidden and output.

The system learns by adjusting its weights, $\mathbf{W_{j,k}}$, while the inputs and outputs are presented to the model in a normalised format. During the fitting process, there are a number of network parameters that can be adjusted to give a better approximation of the output. The parameters are:

Number of neurons in the various layers
Number of layers

Transfer functions chosen in each of those layers

The learning parameters: rate of change of damping/accelerating factor

The rate of learning

Optimisation method chosen (search algorithms)

Figure 3.1 shows three layers: input, hidden and output. Each layer comprises weight matrix, summation units, the bias vector **b**, the transfer function boxes and the output vector **a**. The inputs are represented by xp_1, xp_2,…., xp_R, where R is the number of elements in the input vector. The inputs are each assigned a weight: $w_{1,1}$, $w_{1,2}$….$w_{1,R.}$ The input elements enter the network and are represented as a matrix **W**.

$$= \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,R} \\ w_{2,1} & w_{2,2} & w_{2,R} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} \\ w_{S,1} & w_{S,2} & w_{S,R} \end{pmatrix} \tag{3.6}$$

Each element in the input vector **y** is connected to each neuron through the weight matrix **W**, and each neuron has a bias $\mathbf{b_i}$, a summation, a transfer function ***f*** and an output $\mathbf{a_i}$. All together, the outputs $\mathbf{y_i}$ give the output vector **y**.

(3.7)

This equation is written in the matrix form:

(3.8)

In the equation above, the neuron has a bias b, which is summed with the weighted inputs to form the equation. Generally the summation units find the net value once the inputs are fed to the neuron; this net value is calculated by finding the product of each input value and the corresponding connection weights.

The two neuron output can be written as:

(3.9)

(3.10)

In this particular case,

and (3.11)

Thus, from Figure 10, the output a becomes:

(3.12)

All together, the equation for output for our particular system is:

(3.13)

3.3.2 Creating a neural net using backpropagation.

The following steps were followed in creating a neural net.

3.3.2.1 Build matrix

The matrix for the neural net was derived from the random data that were generated from the four equations in the range of 1 to 10.

X_{all} comprises all the input variables x_1, x_2,……., x_k.

Y_{all} comprises the output variable y

3.3.2.2 Invert the matrix

The Xall and Yall matrices were inverted, with xall and yall becoming the inverted matrices respectively. The inversion is important, as all matrix operations in the Matlab neural net are row operations, while the operations in Excel are done in columns.

3.3.2.3 Separate the set into training, validation and test set

Training set: The dataset for the training set comprises a quarter of the dataset to form a training subset.

Validation set: The dataset for the validation set comprises a quarter of the dataset to form a validation set

Test set: The dataset for the test set comprises half of the remaining data, which then forms the test set.

All the sets were set as equally spaced points throughout the original dataset.

3.3.2.4 Scale the training set

The input training subset was rearranged from minimum values to maximum values using the command minmax.

Pr = minmax(input training subset) (3.14)

The command minmax takes the input training subset, an R x Q matrix, and returns an R x 2 matrix, Pr. This has minimum and maximum values for each row of the input training subset.

The purpose of arranging the training subset from minimum to maximum is to determine the range of the inputs to be used in creating the network.

3.3.2.5 Create the net

The first step in creating the net is to set the feedforward structure in which each layer only receives inputs from the previous layers. The function newff creates the feedforward network.

Initially, when training a feedforward network, it is vital to create the network object. This requires four inputs and returns the network object, which are as follows:

The first input is an R by 2 matrix of minimum and maximum values for each of the R elements of the input vector that were created previously.
The second input is a range containing the sizes of each layer. This is the number of neurons, including weights and bias, a summing junction and an output transfer function.

The third input is a group that contains the names of the transfer functions to be used in each layer.

The final input contains the name of the training function to be used.

There are a number of transfer functions that could be used and tansigmoid is the default transfer function of Matlab. Tansig, as it is known in short in Matlab, is used frequently, as it is differentiable and is a prerequisite in the hidden layer in

backpropagation. The training function serves the purpose of mapping the net output of the neurons (or layers) according to their actual output.

The training functions are created in Matlab to provide a training algorithm for the feedforward backpropagation. The different algorithms use the gradient of the performance function to determine how to adjust the weights to reduce performance. The task of the training algorithms is to reduce the mean square error (mse), which is the average squared error between the network outputs and the target output. The default performance function is mse (mean square error).

All the algorithms use the gradient of the performance function to determine weight adjustments to minimise the mean square error. The gradient determination provides the name, backpropagation – a technique that executes computations through the network backwards.

The several different backpropagation training algorithms have a variety of different computational and capacity requirements, and no one algorithm is best suited to all situations. Examples of backpropagation training functions used by Matlab include trainlm, trainbfg, trainrp and traingd. The transfer functions can be any differentiable transfer function, such as tansig, logsig or purelin.

A caution is issued when using trainlm as the default training function, because it is very fast and therefore requires a lot of memory to run.

The default training algorithm used is trainlm, a network training function that updates weight and bias values according to Levenberg-Marquardt optimisation. The Levenberg-Marquardt optimisation is an algorithm for least squares estimation of nonlinear parameters that outperforms gradient descent and conjugate algorithms by approaching the second order training speed through approximating the Hessian matrix rather than computing it.

The trainlm of the training function has the following default values:

100 epochs: presentation of the set of training input and target pairs to a network and the calculation of new weights and biases. This is done in batch

Minimum performance gradient of

A factor of 1 to use for memory/speed

Performance goal of zero

A weight/bias learning function, 'learngdm' which is a gradient descent weight and bias learning function with momentum.

3.3.2.6 Train the net

Once the feedforward backpropagation network has been created, the net can be trained, using the command:

net,TR,Y] = train(net,P,T,Pi,Ai,val,test) (3.15)

The command net, takes the seven inputs, which are:

The neural network net
P, the network inputs

T, the network targets

Pi, which has a default value of zero, the initial input delay conditions

Ai, which also has a default value of zero, the initial layer delay conditions

val, which is the structure of the validation vectors

test, the structure of the test vectors

The command train then returns the three outputs as follows:

Net, a new network
TR, the training record (showing epoch and performance ‘mse’)

Y, the network outputs

3.3.3 Mutual net for the four equations

The general equations are used for generating random data of a range between 1 and 10. A matrix with inputs and output is constructed for all variables and then exported to Matlab.

The matrix $\mathbf{X_{all}}$ will contain all the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5}$ etc., while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows. These are exported from Excel to Matlab, and these matrices are transposed on xall and yall respectively.

A network is created with a two-layer network, a tansigmoid transfer function in the hidden layer, and a purelin linear transfer function in the output layer.

An initial guess is made of the number of neurons in the hidden layer. It is usually prudent to start the initial guess according to the following equation:

The dataset has one output, thus the network should have one output neuron. The Levenberg-Marquardt algorithm is used for training.

The actual commands are:

```
xall = Xall';                          %transposes Xall
yall = Yall';                          %transposes Yall
iitst = 2:4:1000;                      %space 4 starting 2nd column
iival = 4:4:1000;                      %space 4 starting 4th column
iitr = [1:4:1000 3:4:1000];            %space 4 starting at multiple area
val.P = xall(:,iival); val.T = yall(:,iival); %validation set ¼ of the data
test.P = ptrans(:,iitst); test.T = tn(:,iitst);%test set ¼ of the data
ptr = ptrans(:,iitr); ttr = tn(:,iitr);        %training set ½ of the data
net=newff(minmax(ptr),[noofneuronsinhiddenlayer,noofneuronsintheouter
layer],{'tansig' 'purelin'},'trainlm');

%create a feedforward backpropagation network with two layers and
%tansigmoid, purelin as the transfer functions
%the training function is trainlm, the Levenberg-Marquardt optimisation
% the training set from the input dataset is used
```

[net,tr]=train(net,ptr,ttr,[ ],[ ],val,test);

%a net is created with default zero values for input delay conditions and input

%layer delay conditions. The training function utilises the training data subset

%to return a new net, giving a record of the epochs.

%The training will stop after a number of iterations when errors increase.

plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)

legend('Training','Validation','Test',-1);

ylabel('Squared Error'); xlabel('Epoch')

The above 3 commands are for plotting training, validation and test errors to check the progress of training.

a = sim(net,xall);

Simulates the network to return the simulated y values

[m,b,r] = postreg(a, yall) %

Regression analysis of Yall (target) and 'a' which is the Y values as simulated by the network. This gives a feel of how accurate or good a fit is the simulated model estimating the actual model.

3.3.3.1 Equation 1: — dummy variables x_3, x_4

The matrix $\mathbf{X_{all}}$ will contain all the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5}$, thus a 1 000 by 5 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1000 rows. These are exported from Excel to Matlab, and these matrices are transposed to xall and yall respectively

In Equation 1, a network is created with a two-layer network, a tansigmoid transfer function in the hidden layer, and a purelin linear transfer function in the output layer.

An initial guess is made of the number of neurons in the hidden layer, which is five. The dataset has one output, thus the network should have one output neuron. The Levenberg-Marquardt algorithm is used for training.

The actual commands are:

```
xall = Xall';
yall = Yall';
iitst = 2:4:1000;
iival = 4:4:1000;
iitr = [1:4:1000 3:4:1000];
val.P = xall(:,iival); val.T = yall(:,iival);
test.P = ptrans(:,iitst); test.T = tn(:,iitst);
ptr = ptrans(:,iitr); ttr = tn(:,iitr);
net = newff(minmax(ptr),[5 1],{'tansig' 'purelin'},'trainlm');
[net,tr]=train(net,ptr,ttr,[ ],[ ],val,test);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test',-1);
ylabel('Squared Error'); xlabel('Epoch')
a = sim(net,xall);
[m,b,r] = postreg(a, yall)
```

3.3.3.2 Equation 2: x2, x4

dummy variables

The matrix $\mathbf{X_{all}}$ will contain all the input variables $\mathbf{X_1, X_2, X_3, X_4, X_5}$, thus a 1 000 by 5 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows. These are exported from Excel to Matlab, and these matrices are transposed to xall and yall respectively

In Equation 2, a network is created with a two-layer network, a tansigmoid transfer function in the hidden layer, and a purelin linear transfer function in the output layer.

An initial guess is made of the number of neurons in the hidden layer, which is five. The dataset has one output, thus the network should have one output neuron. The Levenberg-Marquardt algorithm is used for training.

The actual commands are:

```
xall = Xall';
yall = Yall';
```

```
iitst = 2:4:1000;
iival = 4:4:1000;
iitr = [1:4:1000 3:4:1000];
val.P = xall(:,iival); val.T = yall(:,iival);
test.P = ptrans(:,iitst); test.T = tn(:,iitst);
ptr = ptrans(:,iitr); ttr = tn(:,iitr);
net = newff(minmax(ptr),[5 1],{'tansig' 'purelin'},'trainlm');
[net,tr]=train(net,ptr,ttr,[ ],[ ],val,test);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test',-1);
ylabel('Squared Error'); xlabel('Epoch')
a = sim(net,xall);
[m,b,r] = postreg(a, yall)
```

3.3.3.3 Equation 3:

$$\overline{\qquad} \qquad \overline{\sqrt{\qquad}}$$

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_7}$, $\mathbf{x_9}$

The third equation is a combination of the first two equations. The matrix $\mathbf{X_{all}}$ will contain all the input variables, $\mathbf{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}}$, thus a 1 000 by 10 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows. These are exported from Excel to Matlab, and these matrices are transposed to xall and yall respectively

In Equation 2, a network is created with a two-layer network, a tansigmoid transfer function in the hidden layer, and a purelin linear transfer function in the output layer.

An initial guess is made of the number of neurons in the hidden layer, which is 10. The dataset has one output, thus the network should have one output neuron. The Levenberg-Marquardt algorithm is used for training.

The actual commands are:

```
xall = Xall';
yall = Yall';
iitst = 2:4:1000;
```

```
iival = 4:4:1000;
iitr = [1:4:1000 3:4:1000];
val.P = xall(:,iival); val.T = yall(:,iival);
test.P = ptrans(:,iitst); test.T = tn(:,iitst);
ptr = ptrans(:,iitr); ttr = tn(:,iitr);
net = newff(minmax(ptr),[10 1],{'tansig' 'purelin'},'trainlm');
[net,tr]=train(net,ptr,ttr,[ ],[ ],val,test);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test',-1);
ylabel('Squared Error'); xlabel('Epoch')
a = sim(net,xall);
[m,b,r] = postreg(a, yall)
```

3.3.3.4 Equation 4:

Dummy variables are $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$

The matrix $\mathbf{X_{all}}$ will contain all the input variables, $\mathbf{X_1, X_2, X_3, X_4, X_5, X_6, X_7}$, a 1 000 by 7 matrix, while $\mathbf{Y_{all}}$ will have the output variable, a vector with 1 000 rows. These are exported from Excel to Matlab, and these matrices are transposed to xall and yall respectively

In Equation 2, a network is created with a two-layer network, a tansigmoid transfer function in the hidden layer, and a purelin linear transfer function in the output layer.

An initial guess is made of the number of neurons in the hidden layer, which is seven. The dataset has one output, thus the network should have one output neuron. The Levenberg-Marquardt algorithm is used for training.

The actual commands are:

```
xall = Xall';
yall = Yall';
iitst = 2:4:1000;
iival = 4:4:1000;
iitr = [1:4:1000 3:4:1000];
val.P = xall(:,iival); val.T = yall(:,iival);
```

```
test.P = ptrans(:,iitst); test.T = tn(:,iitst);
ptr = ptrans(:,iitr); ttr = tn(:,iitr);
net = newff(minmax(ptr),[7 1],{'tansig' 'purelin'},'trainlm');
[net,tr]=train(net,ptr,ttr,[ ],[ ],val,test);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test',-1);
ylabel('Squared Error'); xlabel('Epoch')
a = sim(net,xall);
[m,b,r] = postreg(a, yall)
```

3.3.3.5 Using the trained net to check for suspect variables

The creation and testing of the net was discussed in the previous section, and the assumption is that the net is a good fit for the proposed model and that the regression constant for simulated Y values and actual values is greater than 98%. Thus, if the network is presented with previously unseen values for inputs, would the network actually give a good estimate of the outputs? Would we be able to pick up trends enforced in the input in the simulated output?

Bearing this questions in mind, we set out to use the trained network to simulate outputs and check whether the simulated outputs pick up suspect variables.

3.3.3.6 Same matrix as created for the net

The same matrices as above were used, as we were not recreating a new network but rather using the trained net.

3.3.3.7 Create data in ascending order

The matrices that were used previously were arranged in ascending order, using the Excel function of sorting the columns of $\mathbf{x_1, x_2, x_3}$, etc. in ascending order.

3.3.3.8 Extract input quantiles from the ascending matrix

Quantiles are a set of 'cut points' that divide a sample of data into groups containing (as far as possible) equal numbers of observations. For the data, a 20%, 50% and 80% quantile were chosen, as they best represent the data and create a fair statistical sample.

3.3.3.9 Calculate average of the input variables from the matrix created

The calculated average is used to form a new matrix of input quantiles, averages. The matrix will have the following number of rows and columns:

The aim is to report the quantile for the one input variable while keeping the other input variables constant by reporting the average for the rest of the input variables. Below is an example, of three variables in the format they were to be presented for training in the neural net:

This matrix is exported to Matlab, and transposed.

3.3.3.10 Simulation

Once the new matrix is formed, the network that was created before is used to simulate new output values, Ysim (the output from the net, which is the simulated output).

The simulated output for where the dummy variables are should not change as the input changes. This is because the dummy variables do not contribute to the output and therefore do not alter the simulated output. The variables that do not show any

significant change in the simulated output can then be declared suspect variables. The new simulated matrix will be in the form:

3.3.3.11 Check if the Ysim changes

The output of the new simulation is then exported to Excel and a comparison is made to see if there is actual change in the Ysim compared to the input.

4. RESULTS

Linear regression is the most commonly used method to analyse process data, and is a method that has been tested thoroughly and is known universally. The success of linear regression stems from its ability to be predictive and provide explanatory results (Gevrey, 2003:259). Linear regression, however, is unable to model complex nonlinear systems, therefore the use of ANN in process data is justified, as the relationships between variables are often nonlinear. Linear regression may have another shortcoming in that one has to examine the final values of R^2 and the beta weights to determine variable importance. A methodology is needed for determining the hierarchy of the "importance" of the variables, and variable contribution. The ANN models are able to make good predictions, although methods need to be developed to clarify the black-box approach to ANNs.

The four equations were modelled using the Matlab regression function and a neural net was developed.

4.1 Equation 1

4.1.1 Matlab regression values using "dropping method"

The results for Equation 1 after regression show R^2 to be 93.31%. This is a good model. When X_1 is left, the R^2 value drops to 53.65%, and when X_5 is left out, R^2 is still low at 55.11%. When X_2 was left out R^2 values drops to 80.25%, while when X_3 and X_4 was left out, R^2 values were not altered from 93%.

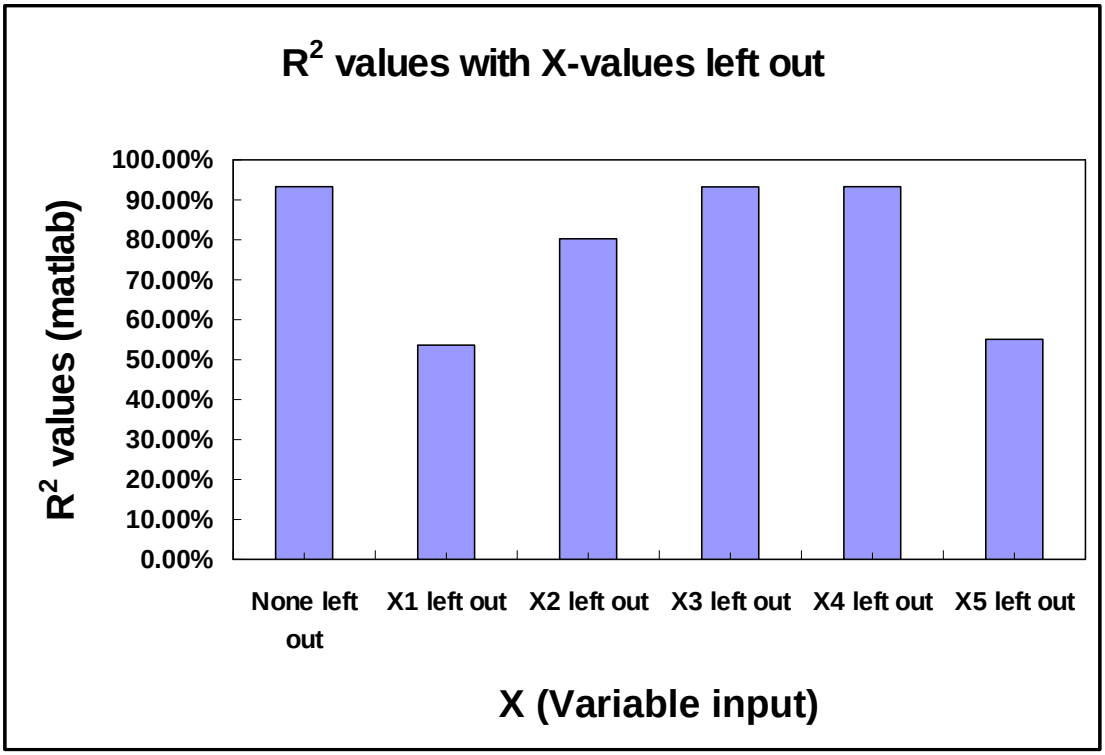

Figure 4.1: Equation 1, R2 values obtained in Matlab with successive variables explanatory variables left out.

The dummy variables added were X_3 and X_4, and the result of R^2 when these two variables were left out show that they are insignificant to the regression value for the whole equation.

4.1.2 Matlab regression model

The actual data for Equation 1 were plotted together with the regression model data. The output from Matlab provides an output with R^2, beta values and P values. The beta values are used to determine the models output using the following equation:

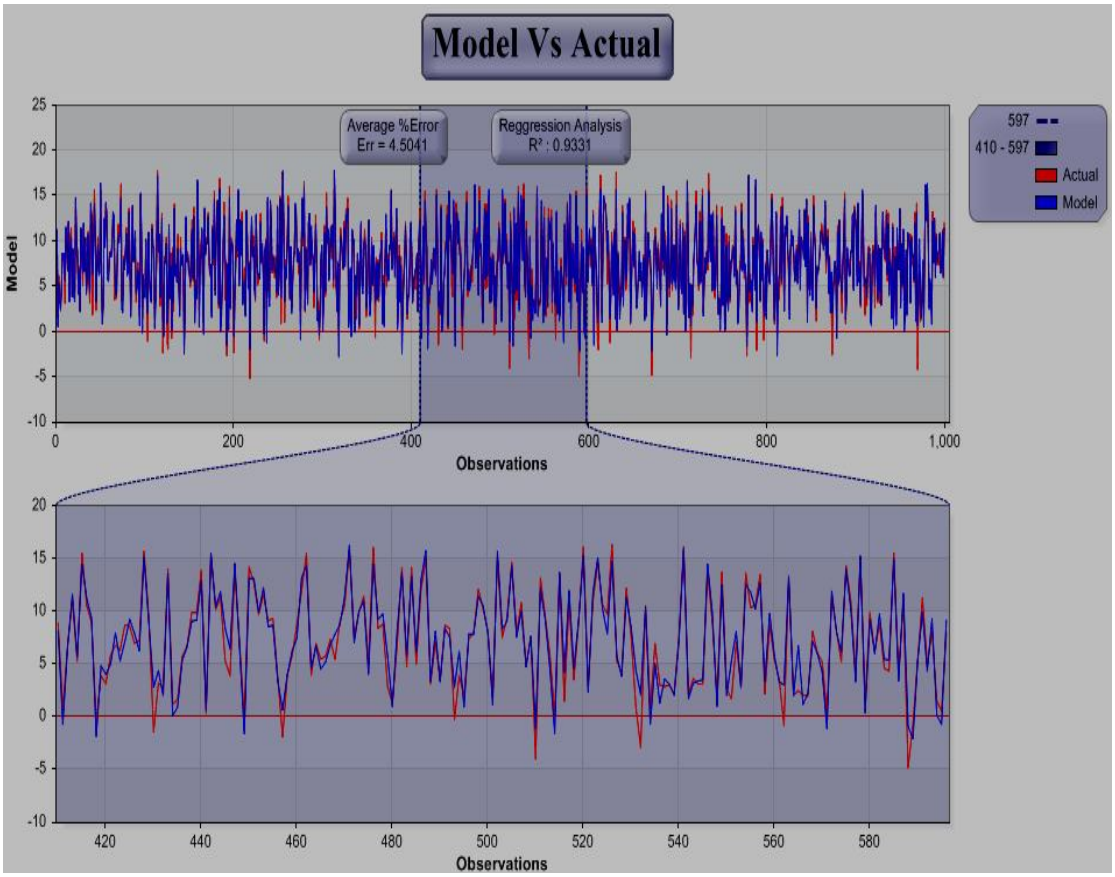

Figure 4.2: Equation 1, comparison of actual data with the Matlab regression model data.

The model is a good estimate of the actual data, although the simulated model underestimates where actual data spikes sharply. The comparison of the actual output values and the simulated output values yields a regression value of 93.31%.

4.1.3 Neural network feedforward backpropagation

The created network has two layers with five neurons in the first layer and one neuron in the second layer.

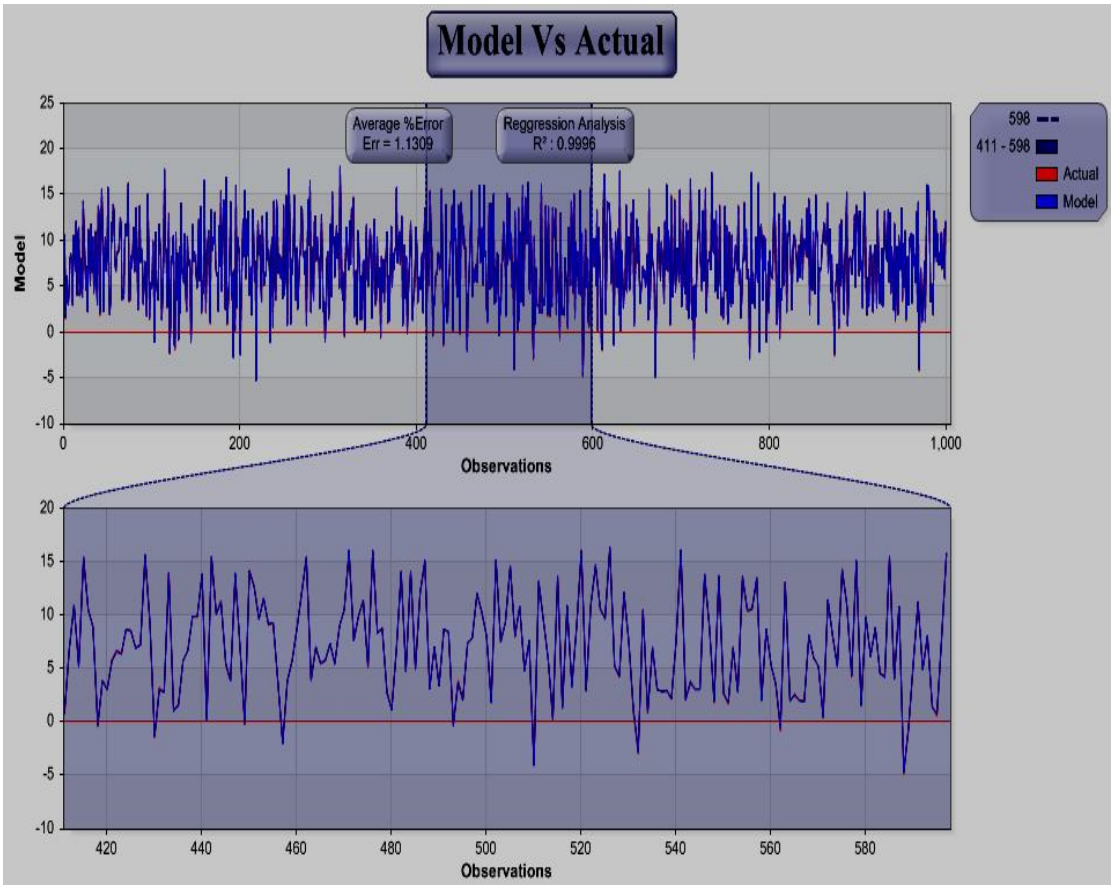

Figure 4.3: Equation 1, comparison of actual data with the neural network model data

The model of the created network is a perfect estimate of the actual data and this is demonstrated by the high regression value comparing the actual data and the simulated, of 99.96%.

4.1.4 Ysim from previously unseen data

The matrix, consisting of quantiles and averages, was presented to the trained network above. The network provided an output, Ysim. The table below consists of the matrix presented to the network and the output that was simulated. The explanatory variables X_1, X_2 and X_5 show a marked change in their output, while the X_3 and X_4 simulated outputs remain relatively constant at .

Table 4.1: Quantiles of Equation 1

Quantiles	X1	X2	X3	X4	X5	Ysim
X1	1.42763536	0.573207651	4.9393637	5.03077135	2.856547	5.0869
X1	2.1592924	0.573207651	4.9393637	5.03077135	2.856547	7.7489
X1	2.77568171	0.573207651	4.9393637	5.03077135	2.856547	10.681
X2	2.1189251	0.295436527	4.9393637	5.03077135	2.856547	5.9445
X2	2.1189251	0.584171067	4.9393637	5.03077135	2.856547	7.607
X2	2.1189251	0.847060902	4.9393637	5.03077135	2.856547	8.0775
X3	2.1189251	0.573207651	1.9100051	5.03077135	2.856547	7.584
X3	2.1189251	0.573207651	4.8182463	5.03077135	2.856547	7.581
X3	2.1189251	0.573207651	8.0212947	5.03077135	2.856547	7.5278
X4	2.1189251	0.573207651	4.9393637	2.06321184	2.856547	7.5457
X4	2.1189251	0.573207651	4.9393637	5.08771183	2.856547	7.5804
X4	2.1189251	0.573207651	4.9393637	7.94943554	2.856547	7.5436
X5	2.1189251	0.573207651	4.9393637	5.03077135	1.773372	5.1625
X5	2.1189251	0.573207651	4.9393637	5.03077135	2.860941	7.5917
X5	2.1189251	0.573207651	4.9393637	5.03077135	3.966951	10.642

All the input variables have a fair variation of quantiles in the matrix represented below, although the output for X_3 and X_4 does not indicate this.

4.2 Equation 2

4.2.1 Matlab regression values using “dropping method”

The results for Equation 2 after regression show R^2 to be 92.91%. This is a good regression model. When X_1 is left out, the R^2 value drops to 53.01%, and when X_5 is left out, R^2 is still low at 50.32%. This shows that X_1 and X_2 are the most important explanatory variables due to their contribution to the output. When X_3 was left out R^2 drops to 81.48%, while the R^2 values are not altered from 93% when X_2 and X_4 are left out.

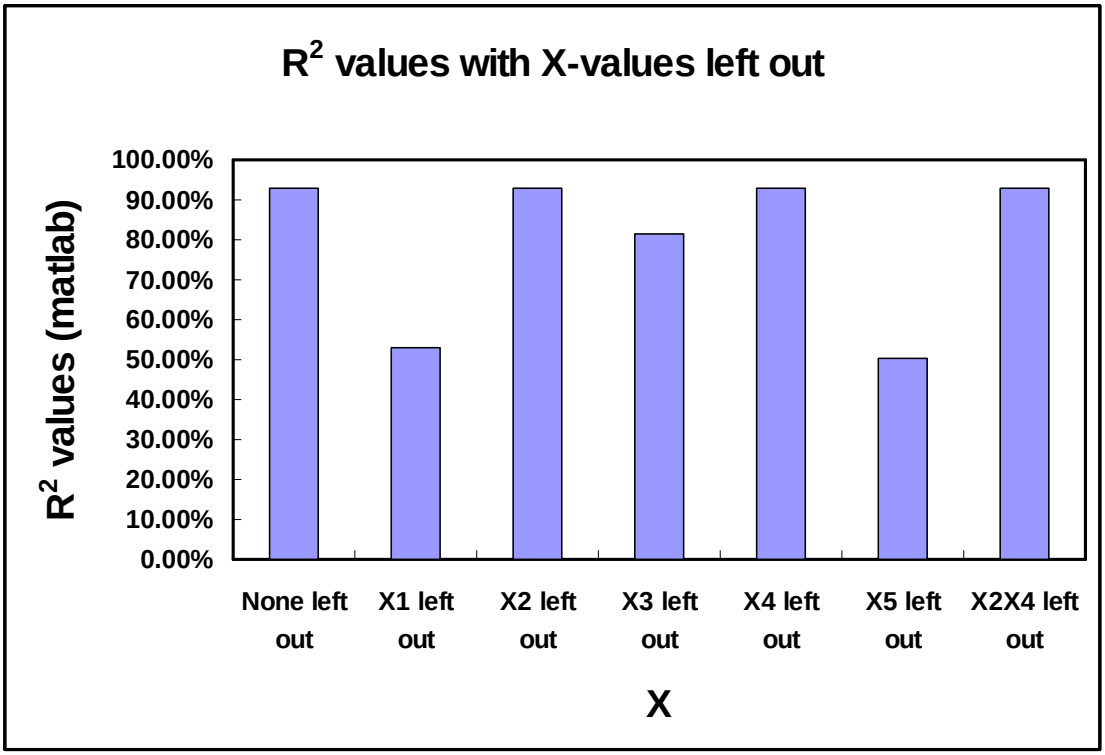

Figure 4.4: Equation 2, R2 values obtained in Matlab with successive explanatory variables left out.

The dummy variables added were X_2 and X_4, and the result of R^2 when these two variables are left out show that they are insignificant for the regression value of the whole equation.

4.2.2 Matlab regression model

The actual data for Equation 2 were plotted together with the regression model data. The output from Matlab provides an output with R^2, beta values and P values. The beta values are used to determine the value of the models output using the following equation:

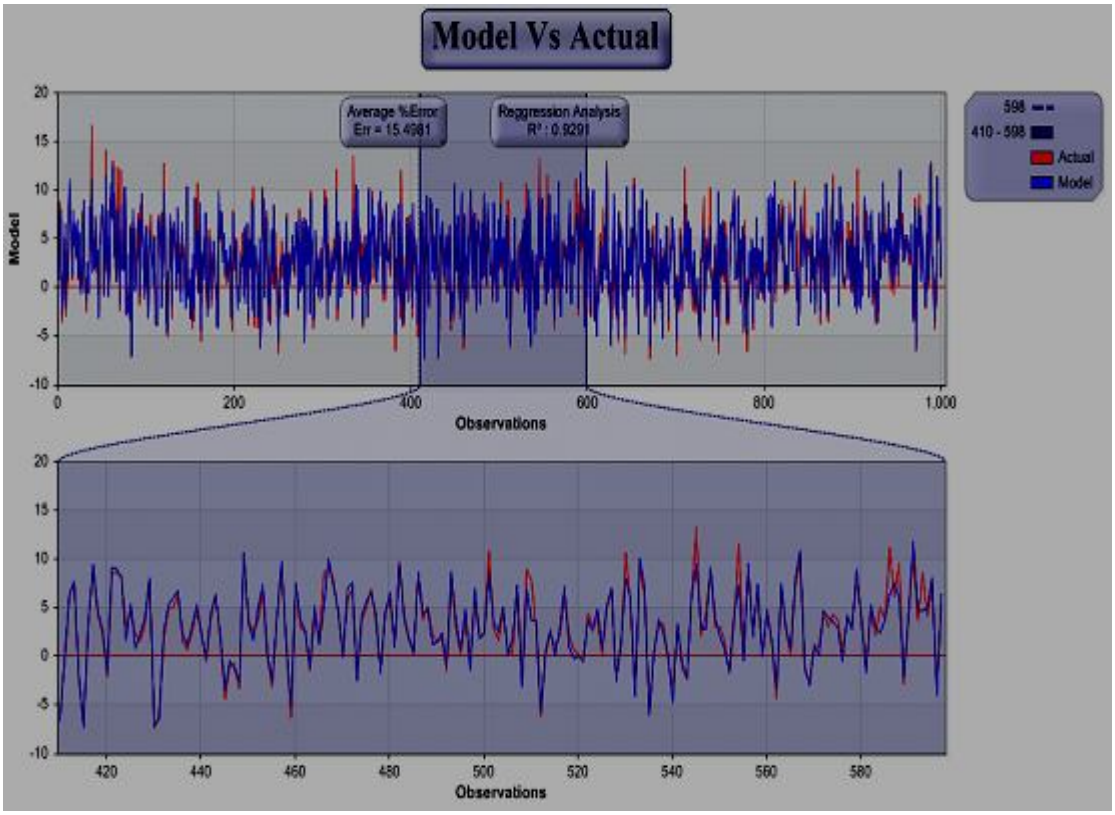

Figure 4.5: Equation 2, comparison of actual data with the Matlab regression model data.

The comparison of the actual data and the simulated data yields an R^2 value of 92.91%. This is a good regression model, but it falls short where the actual data increase or decrease sharply; for example, at observation (record) 499, the actual Y output was -0.08 while the simulated Y was -1.34, yielding an error of 1600%. The occurrence of high errors at some points is compensated for by the number of records, with 1 000 data points.

4.2.3 Neural network feedforward backpropagation

The network was created with two layers. The first layer had five neurons and the second layer had one neuron. The transfer function was: tansigmoid in the first layer and purelin in the second layer.

The model of the created network is a perfect estimate of the actual data and this is demonstrated by the high regression value comparing the actual output data and the simulated output data, namely 99.91%.

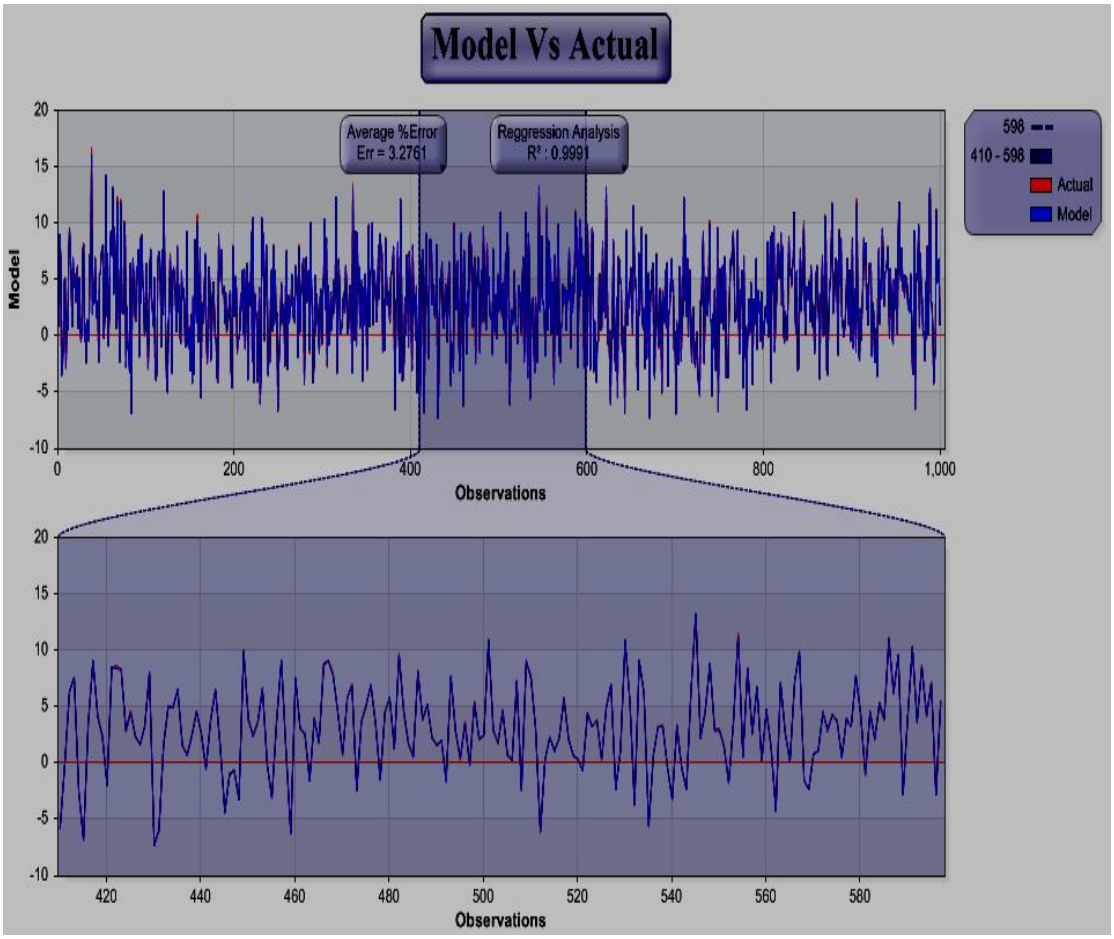

Figure 4.6: Equation 2, comparison of actual data with the neural network model data.

The data points that were underestimated in the regression model were estimated well with the network that was created, thus the R^2 value also increases to a satisfactory value.

4.2.4 Ysim from previously unseen data

The matrix consisting of quantiles and averages was presented to the trained network above. The network provided an output, Ysim. The table below consists of the matrix presented to the network and the output that was simulated. The explanatory variables X_1, X_3 and X_5 showed a marked change in their output, while the simulated output of X_2 and X_4 remained relatively constant at . All the input variables had a fair variation of quantiles in the matrix represented below, although the output for X_2 and X_4 does not indicate this.

Table 4.2: Quantiles of Equation 2

Quantiles	X1	X2	X3	X4	X5	Ysim
X1	6.881318	5.48509967	0.514869	5.5532861	6.720411	-0.28205
X1	13.54928	5.48509967	0.514869	5.5532861	6.720411	2.803
X1	19.35185	5.48509967	0.514869	5.5532861	6.720411	4.7846
X2	13.31903	2.83774044	0.514869	5.5532861	6.720411	2.714
X2	13.31903	5.49112884	0.514869	5.5532861	6.720411	2.7144
X2	13.31903	8.13965449	0.514869	5.5532861	6.720411	2.7149
X3	13.31903	5.48509967	0.221608	5.5532861	6.720411	3.9225
X3	13.31903	5.48509967	0.505194	5.5532861	6.720411	2.7338
X3	13.31903	5.48509967	0.803993	5.5532861	6.720411	2.1813
X4	13.31903	5.48509967	0.514869	2.8495995	6.720411	2.7124
X4	13.31903	5.48509967	0.514869	5.6155923	6.720411	2.7145
X4	13.31903	5.48509967	0.514869	8.306802	6.720411	2.7165
X5	13.31903	5.48509967	0.514869	5.5532861	3.262273	5.492
X5	13.31903	5.48509967	0.514869	5.5532861	6.88632	2.5839
X5	13.31903	5.48509967	0.514869	5.5532861	10.23826	0.017981

4.3 Equation 3

4.3.1 Matlab regression values using “dropping method”

The results for Equation 3 after regression show R^2 to be 48.45%. This is a poor regression model and would give a poor fit to the actual data. When X_8 is left out, the R^2 value drops to 12.73% – the largest change in R^2 among all the input variables. When the other input variables are left out, there is a small change in the value of R^2, with X_1 at 44.61% and X_6 at 44.83%. Dropping X_{10} yields R^2 at 45.61%, and X_5 dropped follows closely at 46.61%. For the remaining input variables, X_3, X_4, X_7 and X_9, R^2 remains unaltered at 49.4%.

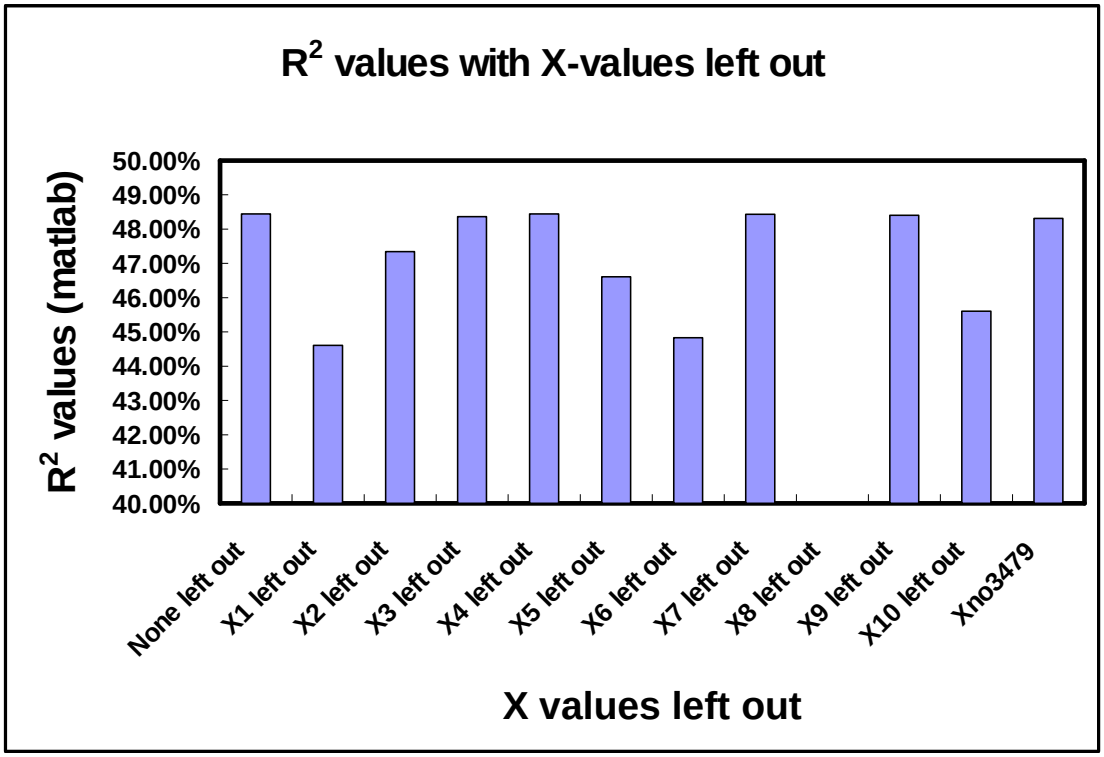

Figure 4.7: Equation 3, R2 values obtained in Matlab with successive explanatory variables left out.

The regression model had an unacceptably low R^2 value. To enable differentiation in the variable contribution, a better model had to be developed. However, a trend emerged when the scale of the R^2 value was scaled down to between 40% and 50%. The dummy variables added X_3, X_4, X_7 and X_9, the R^2 value does not change when compared to when none of the variables were left out. The result for R^2 when these variables were left out showed that they were insignificant for the regression value for the whole equation.

4.3.2 Matlab regression model

The actual data for Equation 3 were plotted together with the regression model data. The output from Matlab provided an output with R^2, beta values and P values. The beta values were used to calculate the value of the models output using the following equation:

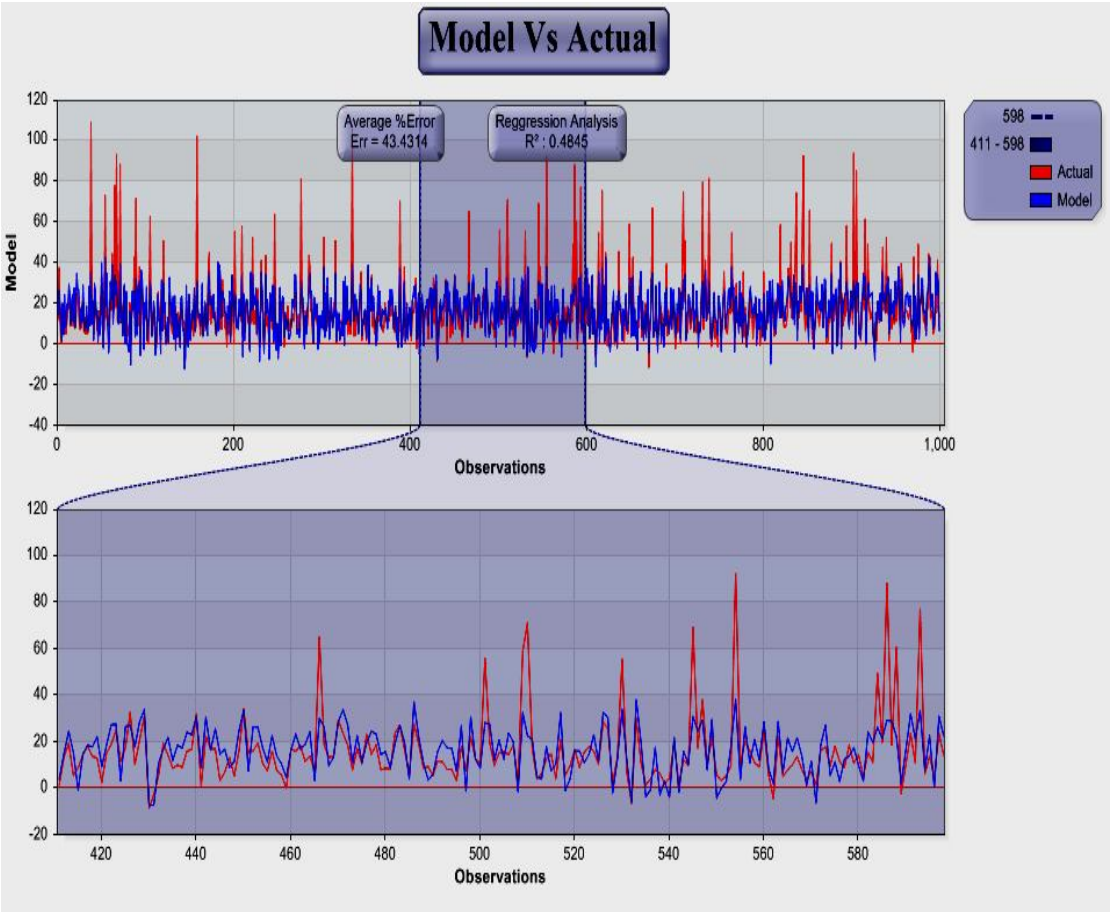

Figure 4.8: Equation 3, comparison of actual data with the Matlab regression model data.

The comparison of the actual data and the simulated data yields an R^2 value of 48.45%. It is a poor regression model, as the simulated output falls short where the actual data increase or decrease sharply and provides a poor fit for all data points. The occurrence of a big error at all points means that this is a particularly unreliable model to ascertain the importance of the contribution of the variables.

4.3.3 Neural network feedforward backpropagation

Two layers are created in the network, with ten neurons in the first layer and one neuron in the second layer. The transfer function in the first layer is tansigmoid and in the second layer it is purelin.

The model of the created network is a perfect estimate of the actual data and this is demonstrated by the high regression value comparing the actual output data and the simulated output data, namely 98.64%.

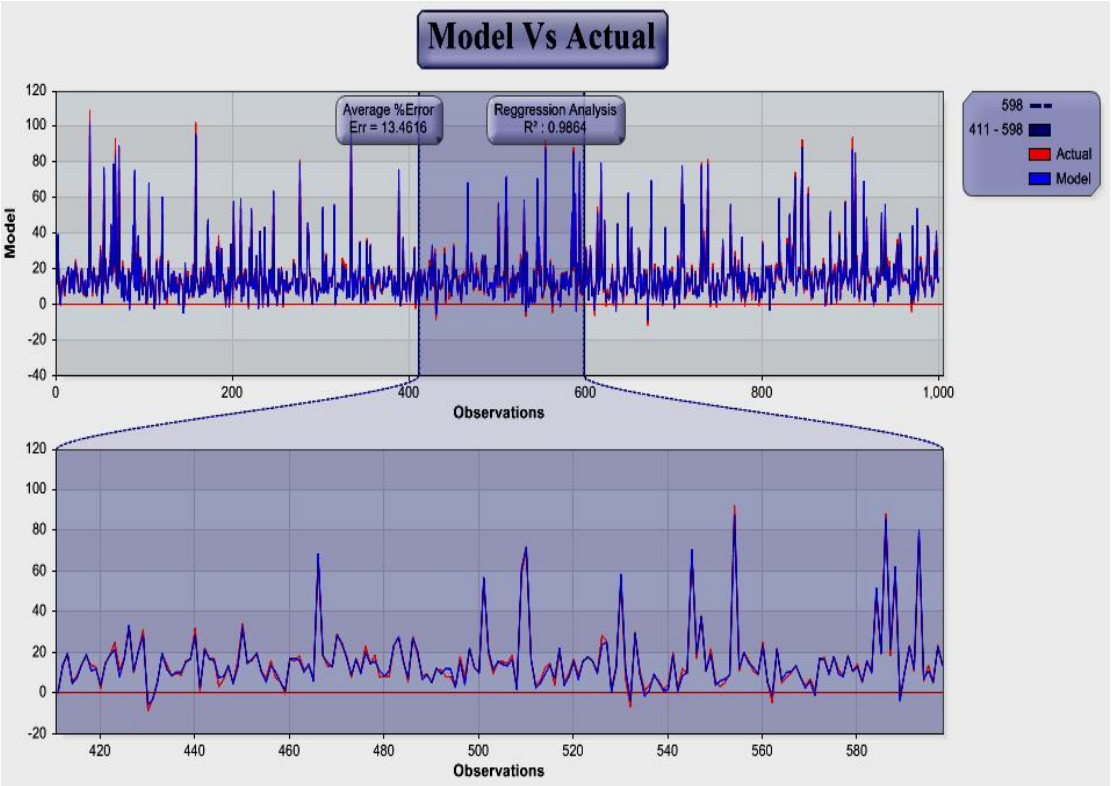

Figure 4.9: Equation 3, comparison of actual data with the neural network model data.

The data points that were grossly underestimated or overestimated in the regression model are well estimated with the network that has been created, thus the R^2 value also increases to a satisfactory value.

4.3.4 Ysim from previously unseen data

The matrix consisting of quantiles and averages was presented to the trained network above. The network provided an output, Ysim. The table below consists of the matrix presented to the network and the output that was simulated. The explanatory variables X_1, X_2, X_5, X_6, X_8, and X_{10} showed a marked change in their output, while the simulated output of X_3, X_4, X_7 and X_9 remained relatively constant at . All the input variables had a fair variation of quantiles in the matrix represented below, although the output for X_3, X_4, X_7 and X_9 does not indicate this.

Table 4.2: Quantiles of Equation 3

Quantiles	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Ysim
X1	1.427635	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	8.0717
X1	2.159292	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.276
X1	2.775682	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	13.966
X2	2.116225	0.2954365	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	9.5845
X2	2.116225	0.5841711	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.155
X2	2.116225	0.8470609	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	12.563
X3	2.116225	0.5720615	1.910005	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.333
X3	2.116225	0.5720615	4.818246	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.082
X3	2.116225	0.5720615	8.021295	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.318
X4	2.116225	0.5720615	4.927822	2.06321184	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.287
X4	2.116225	0.5720615	4.927822	5.08771183	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.096
X4	2.116225	0.5720615	4.927822	7.94943554	2.852065	13.319029	5.4851	0.514869	5.5532861	6.720411	11.387
X5	2.116225	0.5720615	4.927822	5.01864959	1.773372	13.319029	5.4851	0.514869	5.5532861	6.720411	9.0002
X5	2.116225	0.5720615	4.927822	5.01864959	2.860941	13.319029	5.4851	0.514869	5.5532861	6.720411	11.107
X5	2.116225	0.5720615	4.927822	5.01864959	3.966951	13.319029	5.4851	0.514869	5.5532861	6.720411	15.316
X6	2.116225	0.5720615	4.927822	5.01864959	2.852065	6.8813178	5.4851	0.514869	5.5532861	6.720411	8.3176
X6	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.549276	5.4851	0.514869	5.5532861	6.720411	11.188
X6	2.116225	0.5720615	4.927822	5.01864959	2.852065	19.351847	5.4851	0.514869	5.5532861	6.720411	14.077
X7	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	2.83774	0.514869	5.5532861	6.720411	10.749
X7	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.491129	0.514869	5.5532861	6.720411	11.091
X7	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	8.139654	0.514869	5.5532861	6.720411	11.431
X8	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.221608	5.5532861	6.720411	15.539
X8	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.505194	5.5532861	6.720411	11.197
X8	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.803993	5.5532861	6.720411	9.5905
X9	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	2.8495995	6.720411	11.008
X9	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.6155923	6.720411	11.092
X9	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	8.306802	6.720411	11.168
X10	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	3.262273	15.143
X10	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	6.88632	10.963
X10	2.116225	0.5720615	4.927822	5.01864959	2.852065	13.319029	5.4851	0.514869	5.5532861	10.23826	8.8138

4.4 Equation 4

4.4.1 Matlab regression values using “dropping method”

The results for Equation 4 after regression show R^2 to be 77.43%. This is an average regression model and would give a poor fit to the actual data. When X_7 is left out, the R^2 value drops to 18.76%, and this is the largest change in R^2 among all input variables for Equation 4, followed by X_6 at 61.37%. When the other input variables are left out, there is a small change in the R^2 value, with X_1 at 76.53% and X_2 at 75.66%. The X_1 and X_2 variables, however, have an R^2 value that is uncomfortably close to the R^2 value for when all input variables are included. For the remaining input variables, X3, X4 and X5, R^2 remains unaltered at .

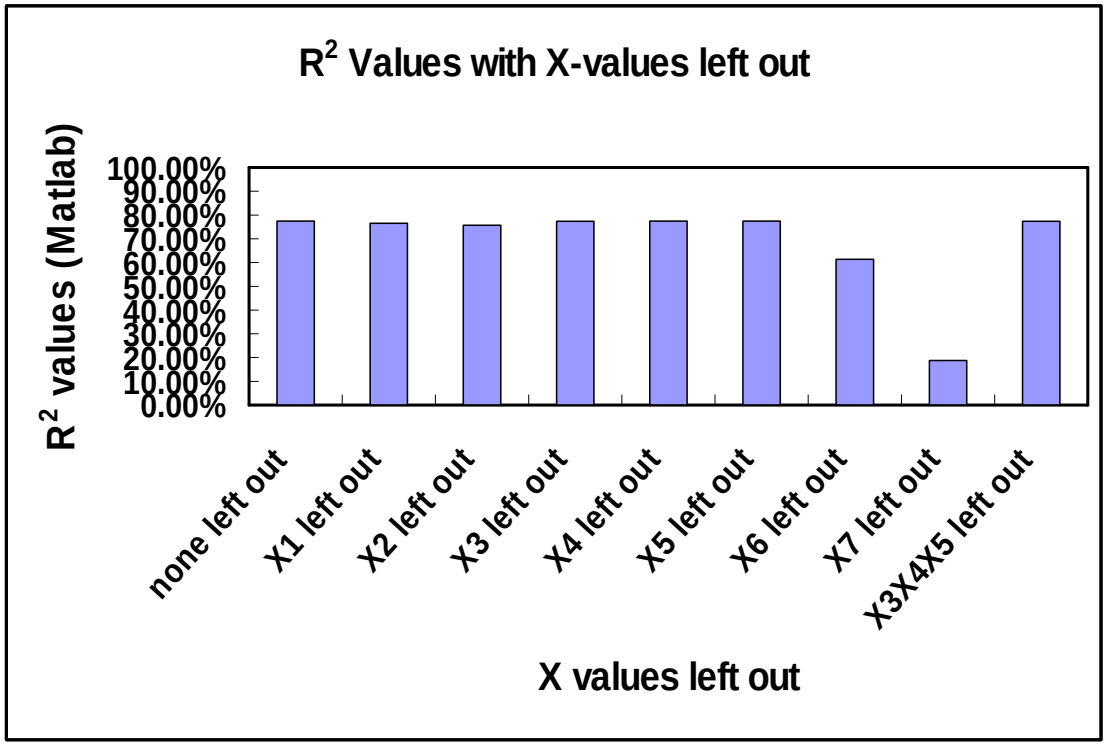

Figure 4.10: Equation 4, R2 values obtained in Matlab with successive explanatory variables left out.

The regression model had a moderately low R^2 value. To enable differentiation in the variable contribution, a better model had to be developed. However, a trend emerged when the scale of the R^2 value was scaled down to between 60% and 80%. When the dummy variables X_3, X_4, X_5 are added the R^2 value does not change. The R^2 value for explanatory variable X_1 and X_2, does not change when compared to when none of the variables were left out.

4.4.2 Matlab regression model

The actual data for Equation 4 were plotted together with the regression model data. The output from Matlab provided an output with R^2, beta values and P values. The beta values were used to calculate the value of the models output using the following equation:

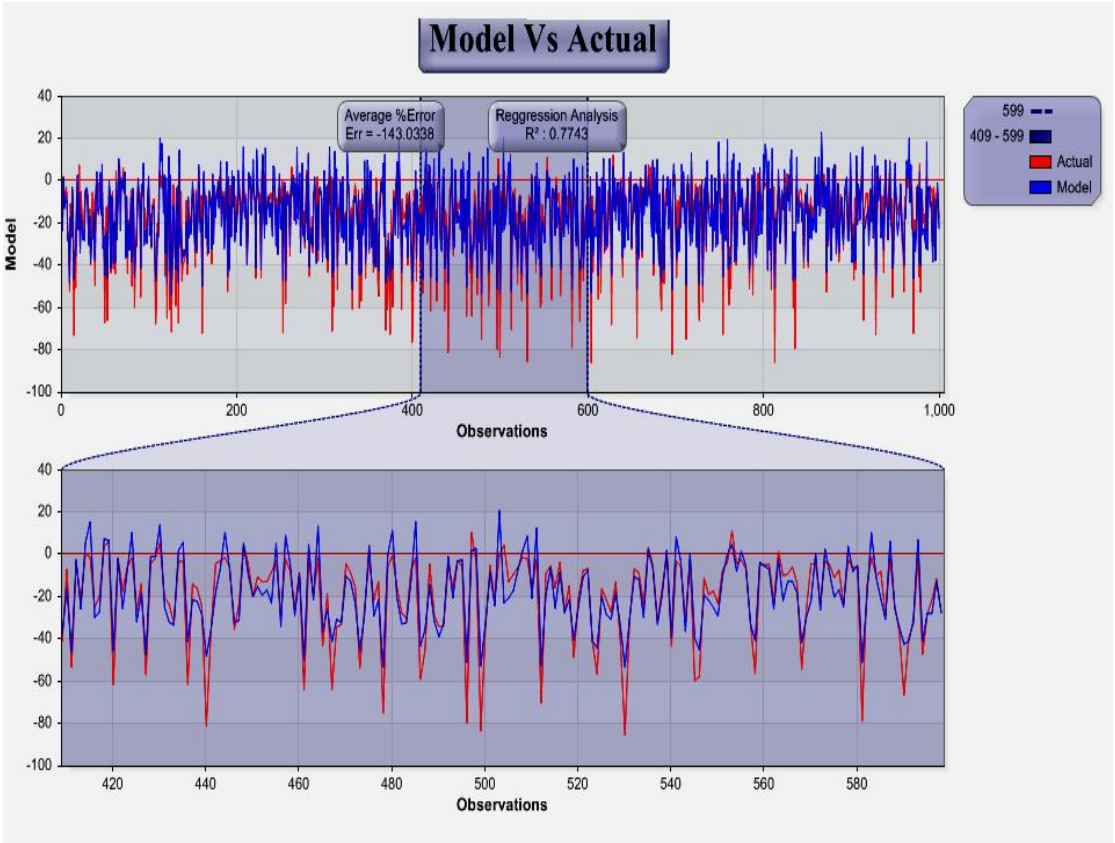

Figure 4.11: Equation 4, comparison of actual data with the Matlab regression model data.

The comparison of the actual data and the simulated data yields an R^2 value of 77.43%. It is a moderately good regression model, as the simulated output falls short where the actual data increase or decrease sharply and provides a poor fit for all data points. The occurrence of a big error at some extreme points means that this is a particularly unreliable model to determine the importance of the contribution of the variables.

4.4.3 Neural network feedforward backpropagation

The network was created with two layers. There were seven neurons in the first layer and one neuron in the second layer. The transfer functions are tansigmoid in the first layer and purelin in the second layer.

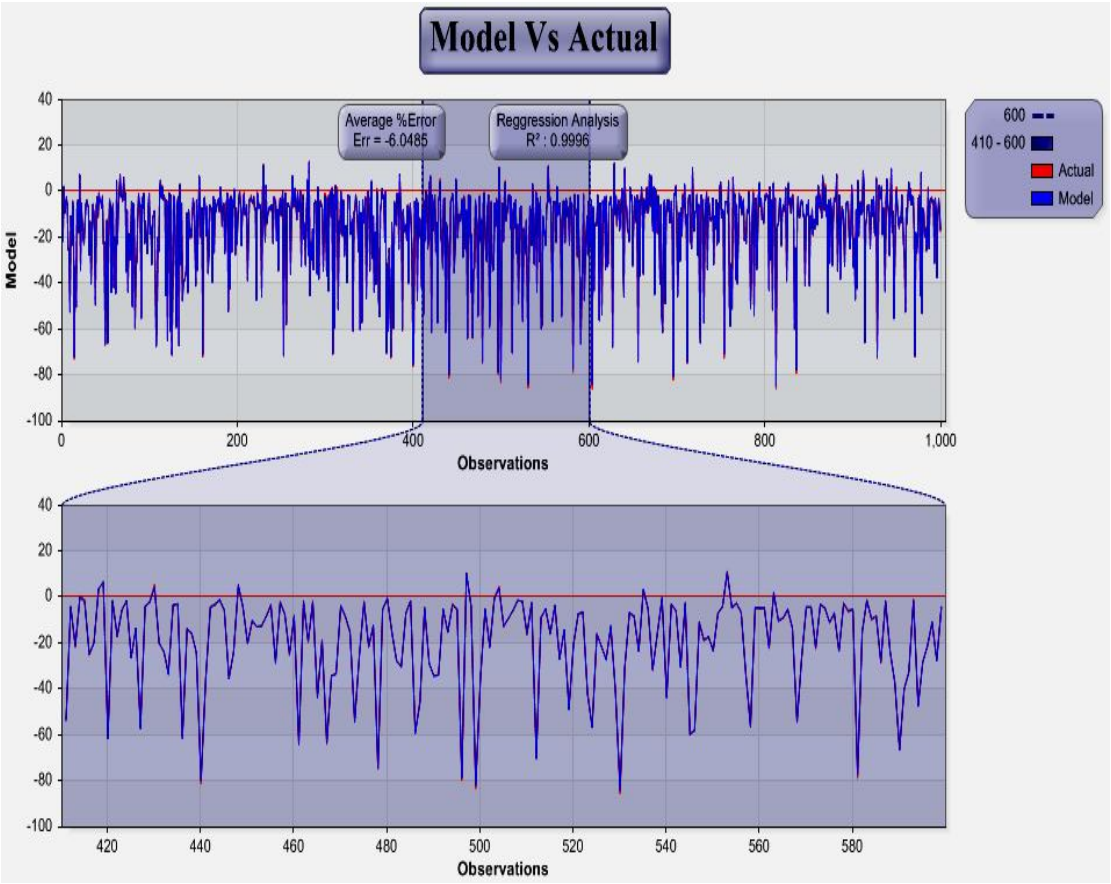

Figure 4.12: Equation 4, comparison of actual data with neural network model data.

The data points that were underestimated or overestimated in the regression model are well estimated with the network that has been created, thus the R^2=0.9996 also has increased to a satisfactory value.

4.4.4 Ysim from previously unseen data

The matrix consisting of quantiles and averages was presented to the trained network above. The network provided an output, Ysim. The table below consists of the matrix presented to the network and the output that was simulated. The explanatory variables X_1, X_2, X_5, X_6 and X_7 showed a marked change in their output, while the simulated output of X_3, X_4 and X_5 remained relatively constant at .

All the input variables had a fair variation of quantiles in the matrix represented below, although the output for X_3, X_4, and X_5 does not indicate this.

Table 4.3: Quantiles of Equation 4

Quantiles	X1	X2	X3	X4	X5	X6	X7	Ysim
X1	1.869113	0.99313188	9.931814	10.138081	10.00544	53.59547	1.99958	-13.528
X1	2.984816	0.99313188	9.931814	10.138081	10.00544	53.59547	1.99958	-12.713
X1	4.268888	0.99313188	9.931814	10.138081	10.00544	53.59547	1.99958	-11.362
X2	3.042004	0.41460771	9.931814	10.138081	10.00544	53.59547	1.99958	-13.948
X2	3.042004	0.99724234	9.931814	10.138081	10.00544	53.59547	1.99958	-12.651
X2	3.042004	1.56256757	9.931814	10.138081	10.00544	53.59547	1.99958	-9.7692
X3	3.042004	0.99313188	3.894756	10.138081	10.00544	53.59547	1.99958	-12.705
X3	3.042004	0.99313188	9.829613	10.138081	10.00544	53.59547	1.99958	-12.665
X3	3.042004	0.99313188	15.94862	10.138081	10.00544	53.59547	1.99958	-12.624
X4	3.042004	0.99313188	9.931814	4.356194	10.00544	53.59547	1.99958	-12.674
X4	3.042004	0.99313188	9.931814	9.8730379	10.00544	53.59547	1.99958	-12.665
X4	3.042004	0.99313188	9.931814	15.97538	10.00544	53.59547	1.99958	-12.655
X5	3.042004	0.99313188	9.931814	10.138081	3.780543	53.59547	1.99958	-12.685
X5	3.042004	0.99313188	9.931814	10.138081	10.26103	53.59547	1.99958	-12.664
X5	3.042004	0.99313188	9.931814	10.138081	15.8695	53.59547	1.99958	-12.645
X6	3.042004	0.99313188	9.931814	10.138081	10.00544	26.12803	1.99958	-7.3873
X6	3.042004	0.99313188	9.931814	10.138081	10.00544	54.14575	1.99958	-12.751
X6	3.042004	0.99313188	9.931814	10.138081	10.00544	80.73805	1.99958	-16.234
X7	3.042004	0.99313188	9.931814	10.138081	10.00544	53.59547	1.423478	-4.1057
X7	3.042004	0.99313188	9.931814	10.138081	10.00544	53.59547	1.953872	-11.7
X7	3.042004	0.99313188	9.931814	10.138081	10.00544	53.59547	2.61589	-34.523

5. DISCUSSION

The addition of average error determination to the R^2 value serves to add other criteria to determine the importance of variables. In regression, the average error is the average absolute difference between the actual output and the simulated output for all input variables, and as the input variables is “dropped”.

In the ANNs created here, the R^2 values obtained are very high – more than 99.0%. The ANN does not return the actual coefficient of the input variables, as the backpropagation algorithm adjusts weights and bias to produce the simulated output values, Ysim. Thus, average error in the ANN is determined from the Ysim. This was done by finding the differences between the simulated output values, and then comparing the magnitude of the differences. The magnitude of the Ysim differences was then sorted and a percentage of the contribution was determined.

The importance of the variables in both the linear regression method and in ANN was determined as a percentage of the total differences. The differences are defined as “gaps”, as they are the differences between the R^2 value of the preceding variables, the average error, or the Ysim value.

The additive percentage is the cumulative of the percentage importance of the variables. This was calculated for the purpose of determining the suspect variables. The additive percentage gives a sum total of the variables that are included in the model. The advantage of using a cumulative variable importance percentage instead of variable importance is more evident in a model that has many explanatory variables. For example, in a model with 10 variables, as in Equation 3, the cumulative percentage gives a more accurate depiction of the effect a particular variable would have and its contribution to the output, in addition to the preceding explanatory variable contribution.

The criteria for identifying suspect variables are based on the cumulative variable importance percentage (additive percentage). The additive percentage has to be below 5% for the explanatory variable to be considered a suspect variable.

5.1 Equation 1

5.1.1 Variable importance based on Matlab regression variables.

The R2 values of the explanatory variables were compared as the variables are dropped from the regression model. The "gaps" were determined, that is the differences between the R^2 values. The smallest differences in R^2 values occurred when X4 was left out of the regression model, and the second smallest difference was when X3 was left out of the regression model. The importance of both variables was less than 1%.

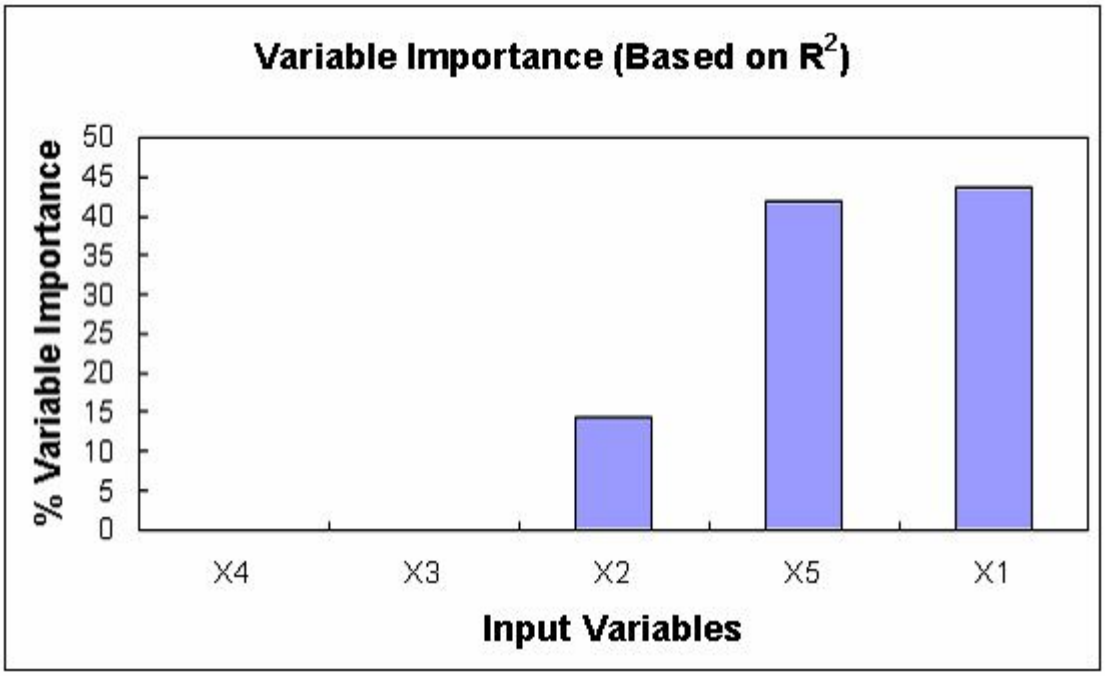

Figure 5.1: Equation 1, variable importance expressed as a percentage, based on Matlab R2 values.

The biggest difference occurred when X_1 and X_5 were left out of the regression model, with the importance of the variables being more than 40%. X_2 falls between the small percentage contribution of X_3, X_4 and X_5, X_1 at 14.35%.

5.1.2 Suspect variables based on Matlab regression variables.

The additive percentage contribution of X_3 and X_4 is 0.07%, thus they are both considered suspect variables. The next explanatory variable is X_2, with an additive

percentage of 14.42%, putting it above the 5% requirement for explanatory variables to be declared suspect variables.

Table 5.1: Equation 1, Matlab R2 values, used in identifying suspect variables.

Variable Importance using R^2 Value				
Input Variables	**sort gaps**	**% variable importance**	**Additive %**	**Suspect Variables**
X4	8E-05	0.00879314	0.00879314	X3
X3	0.00057	0.06265113	0.07144427	X4
X2	0.13057	14.3515058	14.4229501	
X5	0.38197	41.9839525	56.4069026	
X1	0.39661	43.5930974	100	
Total	**0.9098**	**100**		

The other explanatory variables are not declared to be suspect variables following the 5% rule of thumb.

5.1.3 Variable importance based on Matlab average errors.

The smallest difference in R^2 value occurred when X_3 was left out of the regression model, and the second smallest difference was when X_4 was left out of the regression model; the importance of both variables was less than 1%. The variable contribution of X_3 has increased to 0.3% from 0.06% when R^2 variable contribution is compared to average error. The contribution of the other variables, X_2, X_1 and X_5, increased steadily from less than 1% to just over 15% to 33% and finally to over 50% respectively.

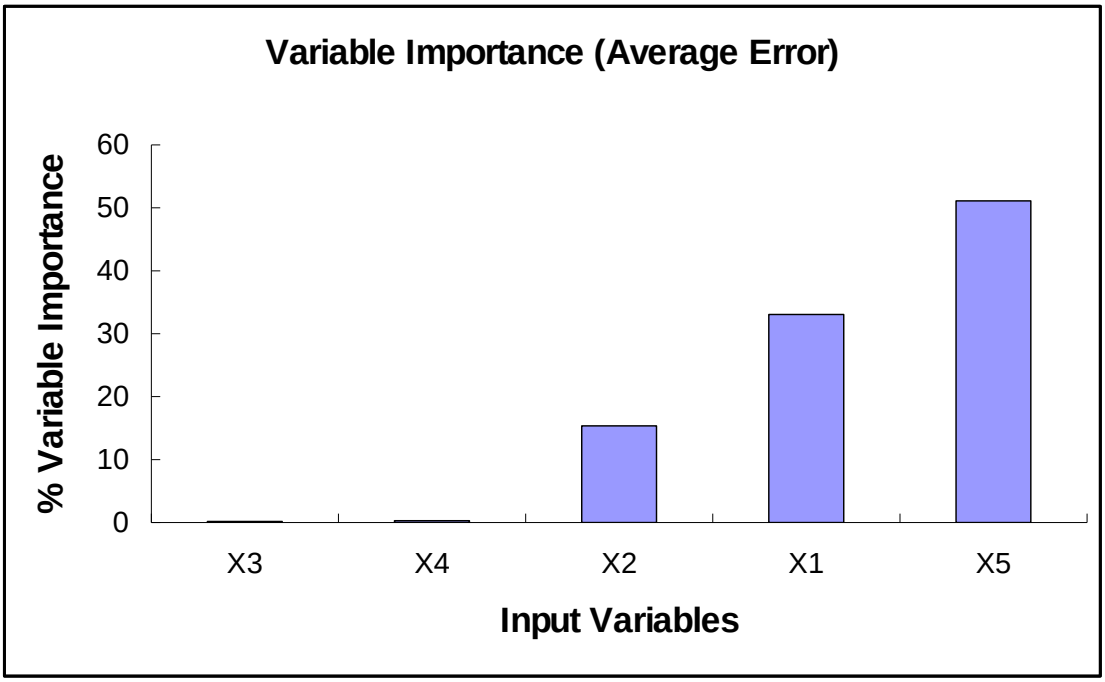

Figure 5.2: Equation 1, variable importance expressed as a percentage, based on Matlab average values.

5.1.4 Suspect variables based on Matlab average errors.

The additive percentage contribution of X_3 and X_4 was 0.44%, thus they were both considered suspect variables. The next explanatory variable was X_2 and the additive percentage soared to 15.82%, putting it above the 5% requirement for explanatory variables to be declared suspect variables.

Table 5.2: Equation 1, Matlab average values, used in identifying suspect variables.

Variable Importance using Average Error

Input Variables	Sort Average Error	% variable importance	Additive %	Suspect Variables
X3	0.001080363	0.165167006	0.165167006	X3
X4	0.001800073	0.275197181	0.440364187	X4
X2	0.10056966	15.37519938	15.81556357	
X1	0.216315316	33.0705217	48.88608527	
X5	0.334337713	51.11391473	100	
Total	**0.654103124**	**100**		

The other explanatory variables are not declared to be suspect variables following the 5% rule of thumb.

5.1.5 Variable importance based on neural network Ysim values.

The differences in the simulated output variables, Ysim, were compared as the quantiles and averages of the explanatory variables were presented and trained. The "gaps" were determined, in other words the differences between the Ysim values for the various explanatory variables in the dataset. The smallest difference in Ysim value occurred when the X_4 quantiles and average were presented to the ANN model, and the second smallest difference occurred when the X_3 quantiles and averages were presented to the ANN model – both with a variable importance of less than 1%.

The explanatory variables X_5 and X_1 had an almost equal variable contribution of over 40%, while X_2 fell in the middle with a contribution of just above 16%.

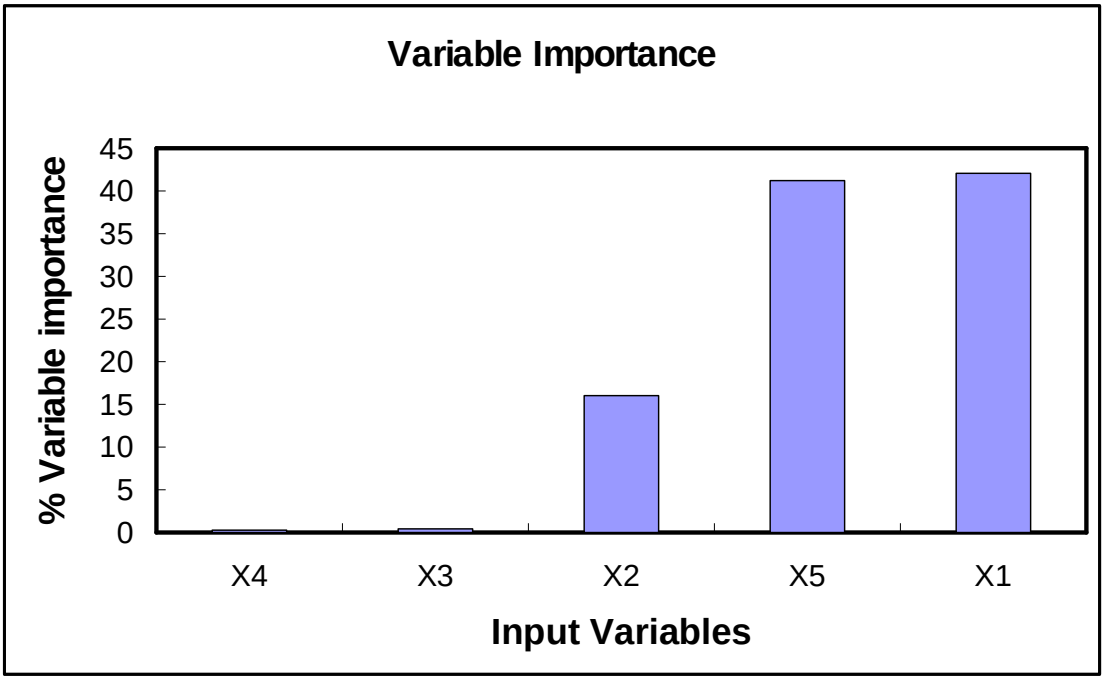

Figure 5.3: Equation 1, variable importance expressed as a percentage, based on neural network Ysim values.

5.1.6 Suspect variables based on neural network Ysim values.

The additive percentage contribution of X_3 and X_4 was 0.07%, thus they were both considered suspect variables. When the next explanatory variable, X_2, was included, the additive percentage was 16.74%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is noteworthy that the

suspect variables, X_3 and X_4, which satisfy the predetermined criteria, are the predefined dummy variables.

Table 5.3: Equation 1, neural network average values, used in identifying suspect variables.

Variable importance				
Input Variable	Gap average	% variable importance	Additive %	Suspect Variables
X4	0.02453333	0.276700051	0.2767001	X4
X3	0.03746667	0.4225691	0.6992692	X3
X2	1.422	16.03807633	16.737345	
X5	3.653	41.20048723	57.937833	
X1	3.7294	42.06216728	100	
Total	**8.8664**	**100**		

Following the 5% rule of thumb, the other explanatory variables were not declared suspect variables.

5.2 Equation 2

5.2.1 Variable importance based on Matlab regression variables.

The R^2 values of the explanatory variables were compared as the variables were dropped from the regression model. The “gaps” were determined, that is the differences between the R^2 values. The smallest differences in R^2 values occurred when X_2 was left out of the regression model, and the second smallest difference was when X_4 as left out of the regression model. The importance of both variables was less than 1%.

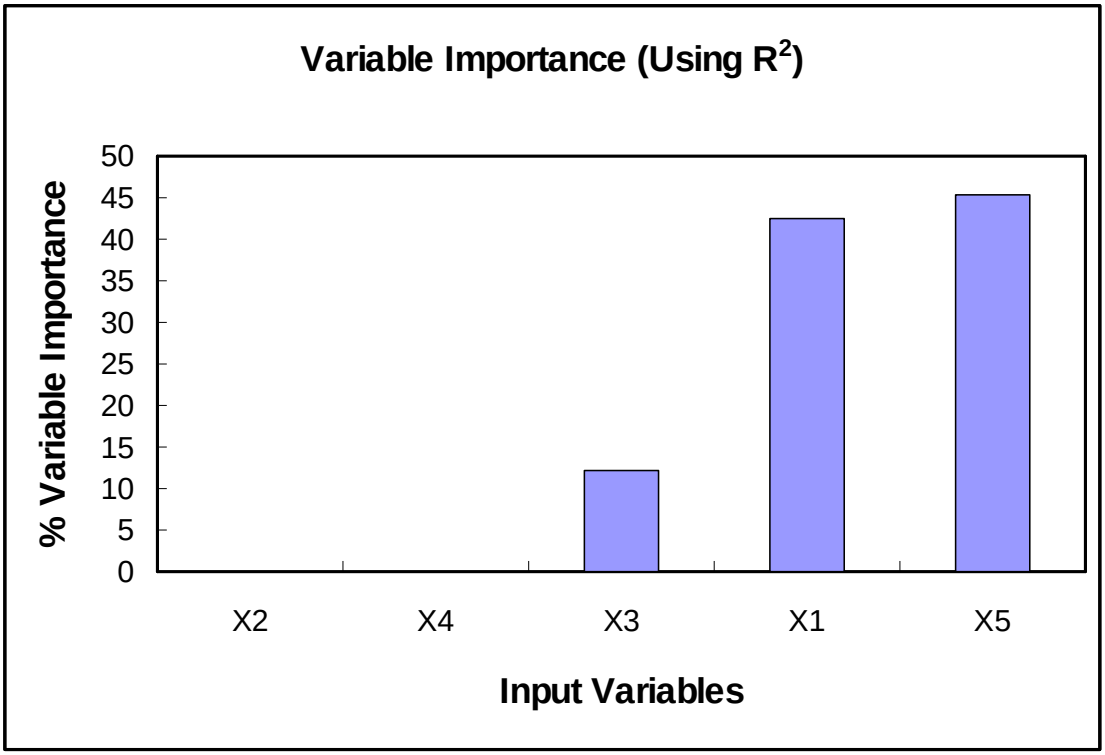

Figure 5.4: Equation 2, variable importance expressed as a percentage based on Matlab R2 values.

The biggest difference occurred when X_2 and X_4 were left out of the regression model, with the importance of the variables being more than 40%. X_3 fell between the small percentage contribution of X_2 and X_4 and the significant percentage contribution of X_1 and X_5, at 12.17%.

5.2.2 Suspect variables based on Matlab regression variables.

The additive percentage contribution of X_2 and X_4 is 0.05%, thus they are both considered suspect variables. The next explanatory variable is X_3, with an additive percentage of 12.17%, putting it above the 5% requirement for explanatory variables to be declared suspect variables.

Table 5.4: Equation 2, Matlab R2 values, used in determining suspect variables.

Variable Importance using R^2 Value				
	sort gaps	% variable importance	Additive %	Suspect Variables
X2	2E-05	0.00212879	0.0021288	X2
X4	3E-05	0.00319319	0.005322	X4
X3	0.11437	12.1734965	12.178819	
X1	0.39909	42.4789782	54.657797	
X5	0.42599	45.3422033	100	
Total	**0.9395**	**100**		

The other explanatory variables, X_3, X_1 and X_5, are not declared to be suspect variables following the 5% rule of thumb.

5.2.3 Variable importance based on Matlab average errors.

The smallest difference in average error values occurs when X_2 is left out of the regression model, and the second smallest difference is when X_4 is left out of the regression model. Both have a variable importance of less than 1%.

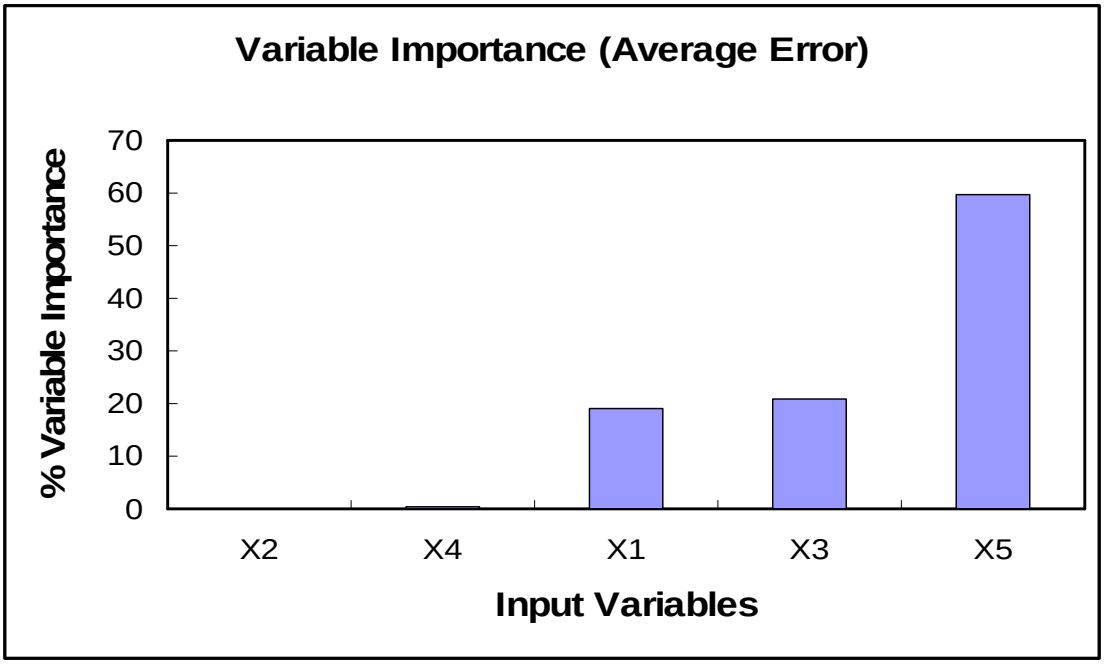

Figure 5.5: Equation 2, variable importance expressed as a percentage based on Matlab average value.

Although, it is noteworthy that the variable contribution of X_4 has increased to 0.4% from 0.02% in average error variable contribution compared to its contribution as per the R^2 value.

5.2.4 Suspect variables based on Matlab average errors.

The additive percentage contribution of X_2 and X_4 was 0.39%, thus they were both considered suspect variables. The next explanatory variable was X_1 and the additive percentage increased rapidly to 19.43%, putting it above the 5% requirement for explanatory variables to be declared suspect variables.

Table 5.5: Equation 2, Matlab average values, used in identifying suspect variables.

Variable Importance using Average Error				
Input Variables	**sort gaps**	**% variable importance**	**Additive %**	**Suspect Variables**
X2	1.72857E-05	0.001985944	0.0019859	X2
X4	0.003381034	0.388444629	0.3904306	X4
X1	0.165734625	19.04113291	19.431563	
X3	0.181668093	20.87171777	40.303281	
X5	0.519602134	59.69671875	100	
Total	**0.870403172**	**100**		

The other explanatory variables, X_1, X_3 and X_5, are not declared suspect variables following the 5% rule of thumb.

5.2.5 Variable importance based on neural network Ysim values.

The differences in the simulated output variables, Ysim, were compared as the quantiles and averages of the explanatory variables were presented and trained.

The “gaps” were determined, in other words the differences between the Ysim values for the various explanatory variables in the dataset. The smallest difference in Ysim value occurs when the X_2 quantiles and average are presented to the ANN model, and the second smallest difference is when the X_4 quantiles and averages are presented to the ANN model – both with a variable importance of less than 1%.

The explanatory variables X_5 and X_1 have an almost equal variable contribution of over 40%, while X_3 falls in the middle with a contribution of just above 14%.

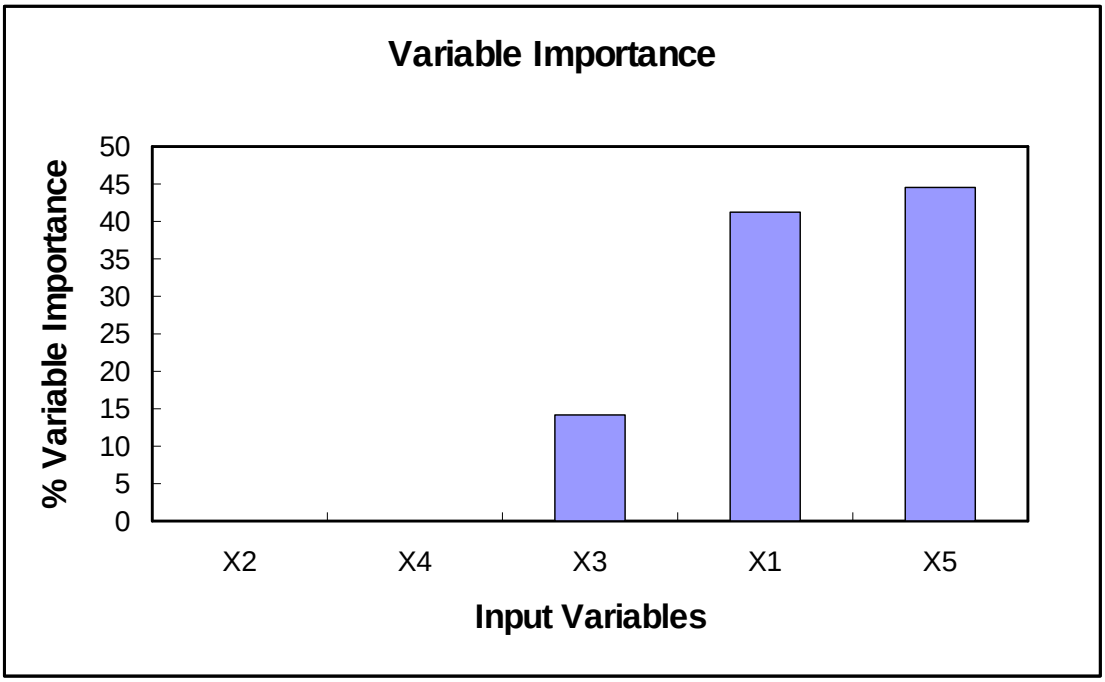

Figure 5.6: Equation 2, variable importance expressed as a percentage based on neural network Ysim value.

5.2.6 Suspect variables based on neural network Ysim values.

The additive percentage contribution of X_3 and X_4 was 0.07%, thus they were both considered suspect variables. When the next explanatory variable, X_2, was included, the additive percentage was 16.74%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is significant to note that the suspect variables, X_2 and X_4, which satisfy the predetermined criteria, are the predefined dummy variables.

Table 5.6: Equation 2, neural network average values, used in identifying suspect variables.

Variable importance				
Input Variable	Gap average	% variable importance	Additive %	Suspect Variables
X2	0.0006	0.00732489	0.007325	X2
X4	0.002733	0.03336896	0.040694	X4
X3	1.1608	14.1712262	14.21192	
X1	3.377767	41.2362987	55.44822	
X5	3.649346	44.5517813	100	
Total	**8.191246**	**100**		

The other explanatory variables, X_3, X_1 and X_5, are not declared suspect variables following the 5% rule of thumb.

5.3 Equation 3

Equation 3 was a combination of Equation 1 and Equation 2, and consequently the regression and ANN models obtained should be a reflection of the separate models of the separate equations that made up Equation 3. Equation 1 had X_3 and X_4 as suspect variables, and Equation 2 had X_2 and X_4 as suspect variables. When Equation 3 was formulated, the X_2 and X_4 of Equation 2 became X_7 and X_9 of Equation 3. The regression model for Equation 3 was especially poor, although the ANN model was vastly superior and therefore the ability to identify suspect variables improved.

5.3.1 Variable importance based on Matlab regression variables.

The R^2 values of the explanatory variables were compared as the variables were dropped from the regression model. The "gaps" were determined, that is the differences between the R^2 values. The smallest differences in R^2 values occurred when X_4 was left out of the regression model, and the second smallest difference was when X_7 was left out. Similarly, when X_9 and then X_3 were left out of the regression model there was a minute difference in the change in R^2 value. The importance of the variables mentioned thus far made was less than 1%.

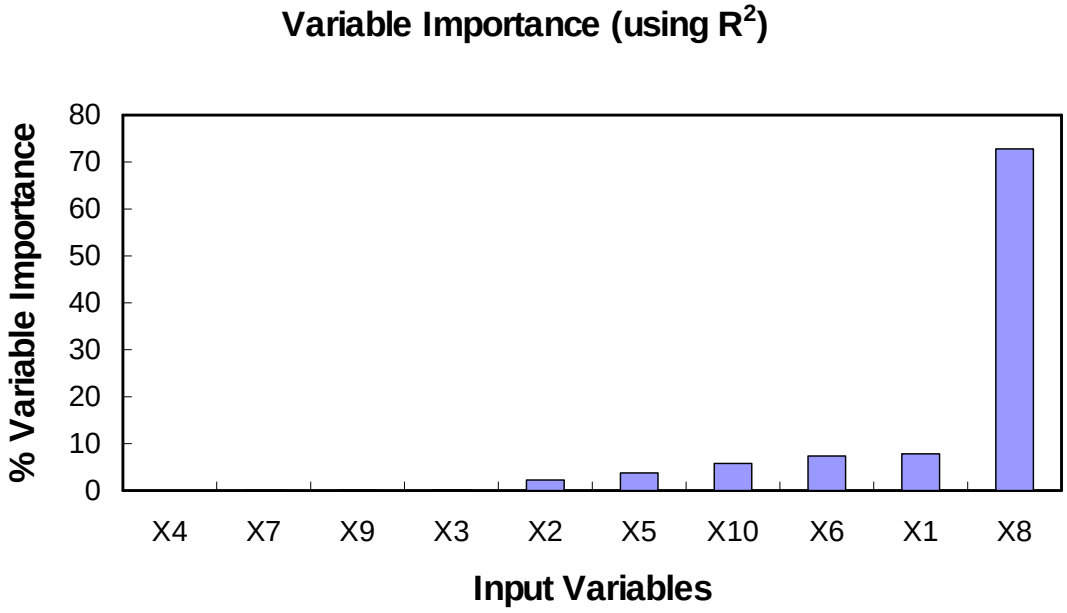

Figure 5.7: Equation 3, variable importance expressed as a percentage based on Matlab R2 values.

The biggest difference occurred when X_8 was left out of the regression model, with an importance of more than 70%. The contribution of the other explanatory variables increased to 2.24% for X_2, to 3.75% for X_5, and to 5.78% for X_{10}. X_6 and X_1 both have a contribution of over 7%.

The regression model for Equation 3 however was very poor, at 48.45%, and therefore the resulting values for variable importance were not accurate.

5.3.2 Suspect variables based on Matlab regression variables.

The additive percentage contribution of X_4, X_7, X_9, X_3, and X_2 was 2.52%, thus they were all considered suspect variables. The next explanatory variable was X_5, with an additive percentage of 6.26%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in Equation 3, X_2 meets the criteria for a suspect variable, contrary to Equation 1 and Equation 2, which form Equation 3.

Table 5.7: Equation 3, Matlab R2 values, used in identifying suspect variables.

Variable Importance using R^2 Value

Input Variables	sort gaps	% variable importance	Additive %	Suspect Variables
X4	3E-05	0.00611309	0.0061131	X4
X7	0.00011	0.02241467	0.0285278	X7
X9	0.00042	0.08558329	0.1141111	X9
X3	0.00079	0.16097809	0.2750891	X3
X2	0.01101	2.24350484	2.518594	X2
X5	0.01838	3.74528782	6.2638818	
X10	0.02836	5.77890983	12.042792	
X6	0.03614	7.36423841	19.40703	
X1	0.03835	7.81456954	27.2216	
X8	0.35716	72.7784004	100	
Total	**0.49075**	**100**		

The other explanatory variables, X_5, X_{10}, X_6, X_1 and X_8, are not declared suspect variables following the 5% rule of thumb.

5.3.3 Variable importance based on Matlab average errors.

The smallest difference in average error value occurred when X_9 was left out of the regression model, and the second smallest difference was when X_5 was left out; the importance of both variables was less than 1%. The other explanatory variables, X_4, X_7 and X_2, all had an individual variable contribution of about 1%. X_3 and X_{10} had an intermediate variable contribution of ~5%. The remaining explanatory variables, X_1, X_6 and X_8, made the bulk of the contribution, with X_8 leading at 60.78%.

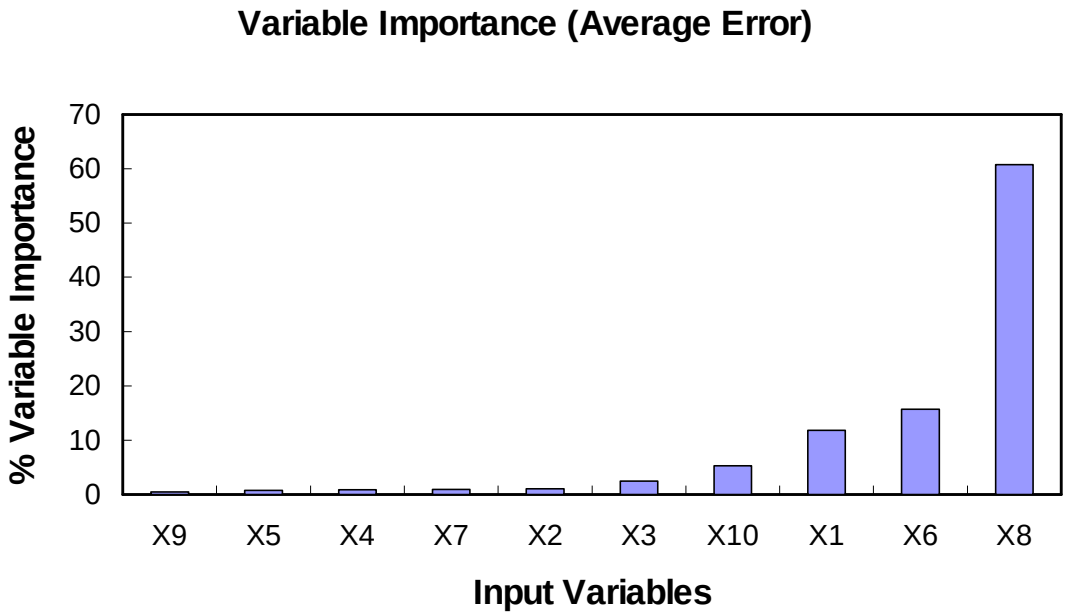

Figure 5.8: Equation 3, variable importance expressed as a percentage based on Matlab average values.

5.3.4 Suspect variables based on Matlab average errors.

The additive percentage contribution of X_9, X_5, X_4, X_7, and X_2 was 3.98%, thus they were all considered suspect variables. The next explanatory variable was X_3 and the additive percentage was 6.44%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in the average error difference in Equation 3, X_5 meets the criteria for a suspect variable, contrary to what is the case in Equation 1 and Equation 2, which form Equation 3.

Table 5.8: Equation 3, Matlab average values, used in identifying suspect variables.

Variable Importance using Average Error				
Input Variables	**sort gaps**	**% variable importance**	**Additive %**	**Suspect Variables**
X9	0.001448308	0.444272135	0.4442721	X9
X5	0.002348066	0.720275217	1.1645474	X5
X4	0.002727642	0.836711146	2.0012585	X4
X7	0.003091644	0.948369608	2.9496281	X7
X2	0.003343115	1.025509098	3.9751372	X2
X3	0.008060823	2.472677719	6.4478149	
X10	0.017262867	5.295428218	11.743243	
X1	0.038420222	11.78550021	23.528743	
X6	0.051166227	15.69536959	39.224113	
X8	0.198126768	60.77588706	100	
Total	**0.325995683**	**100**		

5.3.5 Variable importance based on neural network Ysim values.

The smallest difference in Ysim values occurred when X_9 quantiles and average were presented to the ANN model, the second smallest difference was when the X_3 quantiles and averages were presented to the ANN model, followed by X_4 and X_7; all these had a variable importance of less than 2%. X_2 fell in the middle, with 8.61%. The other explanatory variables, X_6, X_1, X_8, X_5 and X_{10}, had a nearly equal percentage variable contribution of 17 to 18% and higher. The way in which the contribution of the explanatory variables was represented was an accurate portrayal of Equation 3.

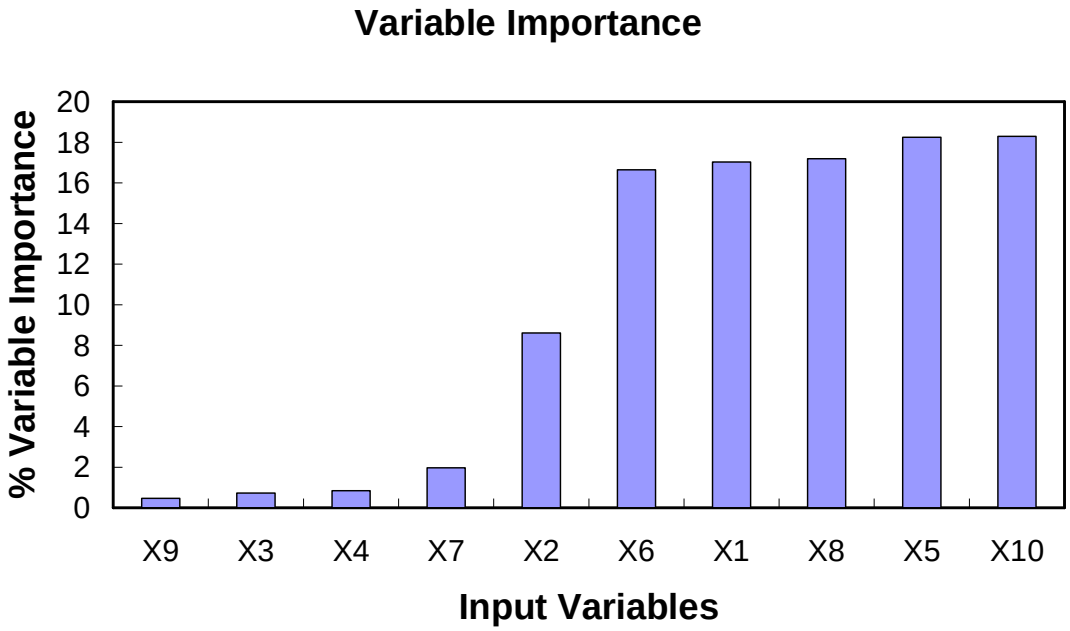

Figure 5.9: Equation 3, variable importance expressed as a percentage based on neural network Ysim values.

5.3.6 Suspect variables based on neural network Ysim values.

The additive percentage contribution of X_9, X_3, X_4 and X_7 was 4.00%, thus they were all considered suspect variables. The next explanatory variable was X_2 and the additive percentage was 12.60%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in Equation 3, the same explanatory variables meet the criteria for a suspect variable as in Equation 1 and Equation 2.

Table 5.9: Equation 3, neural network average values, used in identifying suspect variables.

Variable importance				
Input Variable	**Gap average**	**% variable importance**	**Additive %**	**Suspect Variables**
X9	0.106667	0.46229814	0.462298	X9
X3	0.167333	0.72523021	1.187528	X3
X4	0.194	0.84080475	2.028333	X4
X7	0.454667	1.97054583	3.998879	X7
X2	1.985667	8.60596885	12.60485	
X6	3.8396	16.6409995	29.24585	
X1	3.929533	17.0307746	46.27662	
X8	3.965667	17.1873781	63.464	
X5	4.210533	18.2486413	81.71264	
X10	4.219467	18.2873587	100	
Total	**23.07313**	**100**		

5.4 Equation 4

Equation 4 would be considered as one creating a complex nonlinear system, and thus more complicated to solve. The regression model was comparatively poor, with an R^2 value of 77.43%. The previous regression models of Equation 1 and Equation 2, which provided excellent results in identifying all the suspect variables from the explanatory variables, had excellent regression models, with an R^2 value of over 90%. The ANN model for Equation 3, however, had a high R^2 value of 99.9%, and thus was valuable in identifying the suspect variables.

5.4.1 Variable importance based on Matlab regression variables.

The R^2 values of the explanatory variables were compared as the variables were dropped from the regression model. The “gaps” were determined, that is the difference between the R^2 values. The smallest differences in R^2 values occurred when X_4 was left out of the regression model, and the second smallest difference was when X_5 is left out. Equally, when X_3, X_1 and X_2 were left out of the regression model there was a very minuscule difference in the change in R^2 value. The variables mentioned thus far make less than a 5% contribution to the variable importance.

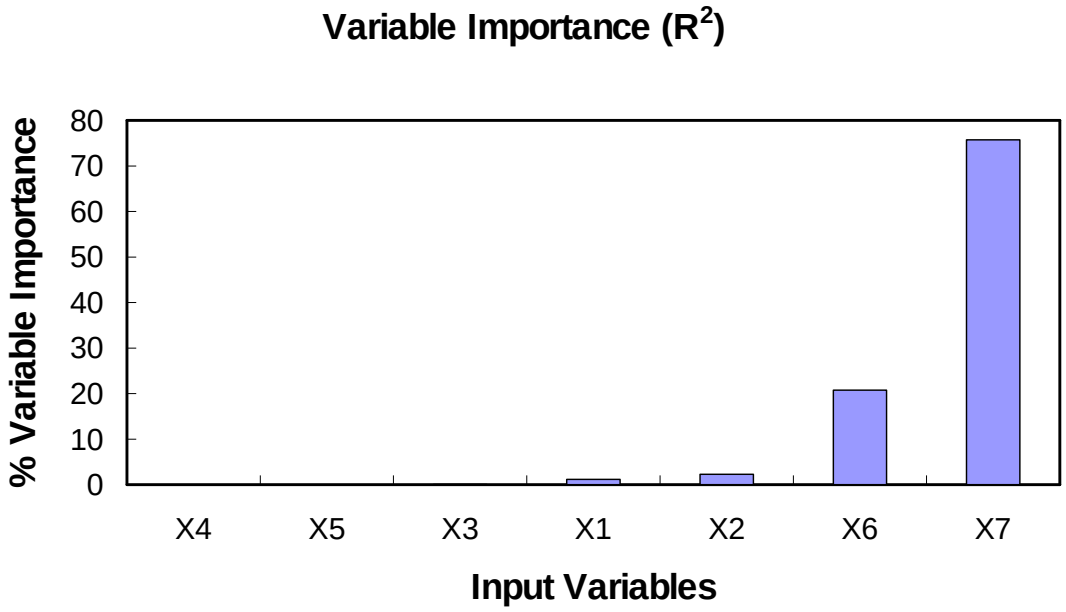

Figure 5.10: Equation 4, variable importance expressed as a percentage based on Matlab R2 value.

5.4.2 Suspect variables based on Matlab regression variables.

The additive percentage contribution of X_4, X_5, X_3, X_1 and X_2 was 3.51%, thus they were all considered suspect variables. The next explanatory variable was X_6 and the additive percentage was 24.25%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in Equation 4, X_1 and X_2 meet the criteria for a suspect variable, contrary to the setup of Equation 4, where the dummy variables added were, X_3, X_4 and X_5.

Table 5.10: Equation 4, Matlab R2 values, used in identifying suspect variables.

Variable Importance using R^2 Value

Input Variables	sort gaps	% variable importance	Additive %	Suspect Variables
X4	7E-05	0.00903786	0	X4
X5	0.00011	0.01420234	0.0232402	X5
X3	0.0003	0.03873367	0.0619739	X3
X1	0.00897	1.15813665	1.2201105	X1
X2	0.01776	2.2930331	3.5131436	X2
X6	0.16063	20.7392966	24.25244	
X7	0.58668	75.7475598	100	
Total	**0.77452**	**100**		

The other explanatory variables, X_7 and X_{10}, were not declared suspect variables following the 5% rule of thumb.

5.4.3 Variable importance based on Matlab average errors.

The smallest difference in average error, at 0.55%, occurred when X_4 was left out of the regression model. The second smallest difference, at 2.21%, was when X_3 was left out, followed by X_5 with 2.58%. The variables mentioned thus far make a contribution of less than 5% to the variable percentage importance, with X_3 and X_5 having an almost identical variable importance.

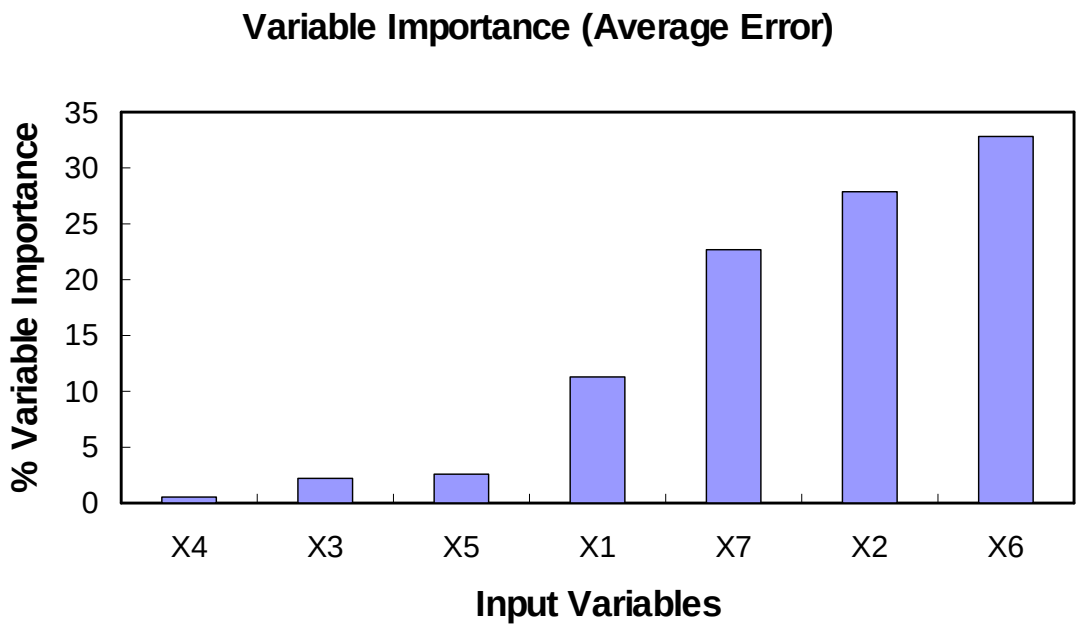

Figure 5.11: Equation 4, variable importance expressed as a percentage based on Matlab average value.

5.4.4 Suspect variables based on Matlab average errors.

The additive percentage contribution of X_4 and X_3 was 2.75%, thus they were both considered suspect variables. The next explanatory variable was X_5 and the additive percentage was 5.34%, putting it above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in Equation 4, X_4 and X_3 meet the criteria for a suspect variable, although X_5, which was added as a dummy variable, was above the suspect variable criteria of 5%.

Table 5.11: Equation 4, Matlab average values, used in identifying suspect variables.

Variable Importance using Average Error				
Input Variables	sort gaps	% variable importance	Additive %	Suspect Variables
X4	0.00730601	0.544976625	0.544976625	X4
X3	0.02964867	2.211581976	2.756558601	X3
X5	0.03464456	2.584240504	5.340799104	
X1	0.15134867	11.28954617	16.63034527	
X7	0.30412599	22.68565958	39.31600485	
X2	0.37363168	27.8702944	67.18629925	
X6	0.43990343	32.81370075	100	
Total	**1.340609**	**100**		

The other explanatory variables, X_5, X_1, X_7, X_2 and X_6, were not declared suspect variables following the 5% rule of thumb.

5.4.5 Variable importance based on neural network Ysim values.

The least difference in Ysim value occurred when the X_4, X_5 and X_3 quantiles and averages were presented to the ANN model, with their variable importance being less than 1%.The other explanatory variables, X_1, X_2, X_6 and X_7, far outweighed the contribution of X_4, X_5 and X_3. X_7 is presented as making the largest contribution of all the explanatory variables, at 66.49%.

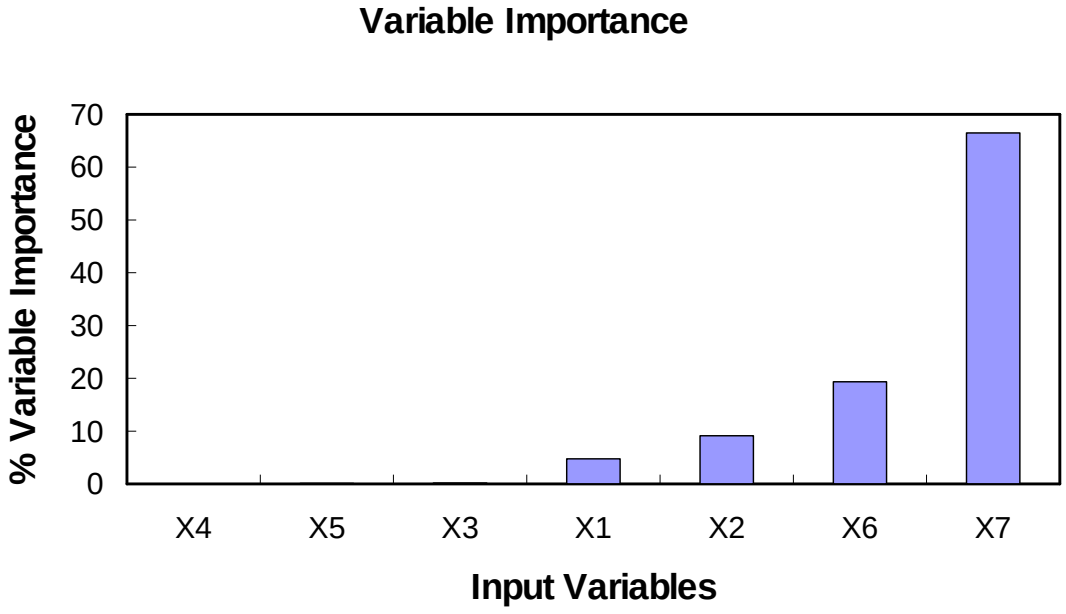

Figure 5.12: Equation 4, variable importance expressed as a percentage based on neural network Ysim value.

5.4.6 Suspect variables based on neural network Ysim values.

The additive percentage contribution of X_5, X_4 and X_3 was 0.31%, thus they were both considered suspect variables. The next explanatory variable was X_1 and the additive percentage as 5.04%, putting it just above the 5% requirement for explanatory variables to be declared suspect variables. It is worth noting that, in Equation 4, X_5, X_4 and X_3 meet the criteria for a suspect variable corresponding to the dummy variables added.

Table 5.12: Equation 4, neural network average values, used in identifying suspect variables.

Variable importance				
Input Variable	**Gap average**	**% variable importance**	**Additive %**	**Suspect Variables**
X4	0.012667	0.04153114	0.041531	X5
X5	0.026667	0.08743399	0.128965	X4
X3	0.054	0.17705382	0.306019	X3
X1	1.444	4.73455041	5.040569	
X2	2.785867	9.13422866	14.1748	
X6	5.8978	19.3375564	33.51235	
X7	20.2782	66.4876456	100	
Total	**30.4992**	**100**		

The other explanatory variables, X_1, X_2, X_6 and X_7, are not declared as suspect variables following the 5% rule of thumb.

6. CONCLUSION

The four equations and the data simulated were used to create models on the basis of multiple linear regression and ANN. The methodology compares R^2 and average error to determine the importance of variables and their contribution to the output in the four equations. Suspect variables were identified less effectively with multiple linear regression and more successfully with the ANN.

Multiple linear regression and ANN of Equation 1 produced an excellent model with a high R^2 and low average error. The suspect variables identified were X_1 and X_3, in line with the dummy variables that were predefined in Equation 1. Equation 2 also had a fine multiple linear regression and ANN, with a high R^2 value and low average error. The suspect variables identified were X_2 and X_4, which have been preselected as dummy variables in Equation 2.

Equation 3, however, had a very poor multiple regression model. This was unexpected, as Equation 3 was a combination of Equation 1 and Equation 2. Consequently, the R^2 was very low and the average error was high from the multiple linear regression model. The suspect variables identified were X_2, X_3, X_5, X_4, X_7 and X_9. The multiple linear regression did not yield a good result, as the dummy variables for Equation 3, namely X_3, X_4, X_7 and X_9, were not recognised. The ANN model improved the R^2 to a satisfactory value and the average error dropped drastically. The suspect variables identified then were X_3, X_4, X_7 and X_9, which were as expected from the predefined dummy variables.

The linear regression model of Equation 4 was normal, with an average R^2 value and relatively low average error. Nevertheless, the suspect variables identified from the R^2 value were different from the ones recognised from the average error. R^2 showed all the explanatory variables to be suspect variables, except X_6 and X_7, while average error recognised only X_3 and X_4 as suspect variables. Thus the multiple linear regression model was not accurate in identifying the predefined dummy variables as suspect variables. The R^2 value of the ANN model was high, the average error was low and the explanatory variables identified as suspect variables were X_3, X_4 and X_5. The suspect variables identified in Equation 4 were the predefined suspect variables.

The criterion chosen for suspect variables was for an additive variable importance of less than 5%. In the particular equations, the criterion was sufficient, although in a set of data with 100 input variables or more, a variable contribution of 0.5% would be considered as a noteworthy contribution. However one can adjust the criterion used for declaring a explanatory variable, a suspect variable, depending on the number of explanatory variables or number of outputs.

The combination of traditional statistical modelling and ANN can better be used in determining the importance of variables and their contribution to the output. The contribution of each variable in the four equations was clearly defined and presented, and may be utilised by those with little understanding of the ANN or the process data they are modelling to reduce the number of explanatory variables. An examination of the explanatory variables declared as suspect variables and the variable contribution to the output may lead to a reduction in the complexity of sometimes overwhelming models as a pre-process in data modelling.

The summary of the performance of the multiple linear regression and ANN models for all equations was extensively explored in Chapter 5. A confirmation of the legitimacy of an explanatory variable being declared as a suspect variable was accomplished with their emergence as a suspect variables in multiple linear regression model and ANN. Their variable contribution was determined from the R^2 value and the average error and the ANN output simulated value.

7. REFERENCES

Brownlee K.A. 1967. Statistical theory and methodology in science and engineering. New York: Wiley.

Bruns, R.E. 2002. Simulation of an industrial waste water treatment plant using artificial neural networks and principal component analysis. *Brazilian Journal of Chemical Engineering*, 19(4).

Demuth, H. & Beale, M. 2004. *Neural network toolbox, for use with MATLAB®*. Massachusetts: The Math Works, Inc.

Edwards, A.L. 1976. An introduction to linear regression and correlation. San Francisco: WH Freeman.

Edwards, A.L. 1979. Multiple regression and the analysis of variance and covariance. San Francisco: WH Freeman.

Garson, D.G. 2007. Quantitative research in public administration: statistic's notes. Class notes. Northern Carolina State University.

Gevrey, M. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160:249–264.

Grieu, S. 2006. KSOM and MLP neural networks for on-line estimating the efficiency of an activated sludge process. *Chemical Engineering Journal*, 116:1–11.

Hagan, M. 1995. A Modular Control Systems Laboratory. *Computer Applications in Engineering Education*, 3(2): 89-96.

Hastie, T., Tibishrani, R. & Friedman, J. 2001. The elements of statistical learning: data mining, inference and prediction. New York: Springer.

Himmelblau, D. 2000. Application of artificial neural networks in chemical engineering. *Korean Journal of Chemical Engineering,* 17(4):373–392.

Hornik, K., Stichcombe, M. & White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Network*, 2(5):359–366.

Kemp, S., Zaradic, P. & Hansen, F. 2007. An approach for determining relative input parameter importance and significance in artificial neural networks. *Ecological Modelling*, 204(2007):326–334.

Martinez, A. 1999. Study of weight importance in neural network with collinear variables in regression problems. *International conference on industrial and engineering applications of artificial intelligence and expert systems*, 1611: 101-110.

Olden, D. 2004. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling,* 178:389–397.

Papadokonstantakis, S., Machefer, S., Schnitzlein, K. & Lygeros, A. 2005. Variable selection and data pre-processing in NN modelling of complex chemical processes. *Computers and Chemical Engineering*, 29:1647–1659.

Pham, D.T. 1995. An introduction to artificial neural networks. *Neural networks for chemical engineers*. Elsevier Science BV:1

Rumelhart, E.D. & McClelland, L.J. (with the PDP Research Group). 1986. *Learning in Boltzmann machines.* Cambridge, MA: The MIT Press.

Tsaptsinos, D. 1995. Back-propagation and its variations. *Neural Networks for Chemical Engineers*. Elsevier Science BV:33

8. BIBLIOGRAPHY

Allison, P.D. 1999. *Multiple regression.* Thousand Oaks: Pine Forge Press.

Amazedsaint blogs. 2006. Articles-Design pattern, Neural Networks, C#, programming.http://amazedsaintarticles.blogspot.com/2006/06/brainnet-ii-inside- story-of-brainnet.html [Accessed on 15 June 2007]

Anderson, James, A.1972. A Simple Neural Network Generating an Interactive Memory, *Mathematical Biosciences 14*:197-220

Association for the advancement of artificial intelligence. n.d. *Neural Networks and connectionist system* http://www.aaai.org/AITopics/html/neural.html [Accessed on 23 January 2007]

Burton, M. 2004. *Neural networks*. Rhodes University: Department of Mathematics.

Children's Mercy Hospital. 2007. *Definition of P value*.http://www.childrens-mercy.org/stats/library/pvalueci.asp [Accessed on 11 June 2007]

Duke University, Department of Statistical Science. 2000. *A framework for nonparametic regression using neural networks*. http://ftp.isds.duke.edu/WorkingPapers/00-32.pdf [Accessed 7 June 2007]

Gorni, A. 1997. The application of neural networks in the modelling of plate rolling processes. *JOM-e*: 49(4).

Glynn, W. 2007. Linear regression with Matlab. MS&E Introduction to stochastic modelling 121:1–4.

Grossberg, S. 1976. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23: 121-134.

Gurdani, R., Onimaru R.S. & Crespo, F.C.A. 2001. Neural network model for the on-line monitoring of a crystallization process. *Brazilian Journal of Chemical Engineering*, 18(3):267–275.

Hebb, DO. 1949. *The Organization of Behaviour*. New York: Wiley.

Hopfield, J. 1982. Neural network and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences of the United States of America, 79* (8)*:*2554-2558.

Kohonen, Teuvo. 1972. Correlation Matrix Memories, *IEEE Transaction on Computers*, C-21:353-359

Luxhøj, J.T. 1997. Neural network in bioprocessing and chemical engineering. *IEE Transactions*, 29(9):810–811.

Minsky, M., Papert, S. 1969. *Perceptrons – An Introduction to Computational Geometry*. The MIT Press, Cambridge, MA.

Marquardt, D.W. 1963. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math,* 11:431–441.

North Carolina State University, College of Social Science and Humanities. 1996. *Multipleregression*.http://www2.chass.ncsu.edu/garson/PA765/regress.htm [Accessed on 6 June 2007]

Oklahoma State University College of Engineering. n.d. Neural Network Design. http://hagan.okstate.edu/nnd.html [Accessed on 26 February 2007]

Rosenblatt, Frank .1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Cornell Aeronautical Laboratory*, *Psychological Review*, 65, No. 6: 386-408.

Rumelhart, D.E., J.L. McClelland and the PDP Research Group .1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition.(1): Foundations, Cambridge, MA: MIT Press

Statistica. 1994. *Descriptive Statistics*. http://www.statsoft.com/textbook/stbasic.html#Descriptive%20statistics [Accessed on 11 June 2007]

University of Glasgow. 1997. *Statistics Glossary*. http://www.stats.gla.ac.uk/steps/glossary/hypothesis_testing.html#pvalu e [Accessed on 11 June 2007]

University of Texas at Austin. n. d. *Lecture notes on the advantages of neural networks.*http://homepage.psy.utexas.edu/homepage/class/Psy387R/LectureNotes/Lectures%2013/5Advantages%20of%20networks.doc [Accessed on 5 May 2007]

University of Wollongong: School of Electrical, Computer and Telecommunications engineering. 2003. *Back Propagation Neural Network Tutorial*. http://ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English/BPN_English/BPN_English.html [Accessed on 03 July 2007]

University of North Carolina Wilmington. n.d. *Introduction to the backpropagation algorithm*.http://people.uncw.edu/tagliarinig/Courses/415/Lectures/An%20Introduction%20To%20The%20Backpropagation%20Algorithm.ppt [Accessed on 03 September 2007]

University of Sterling, Department of computing and mathematics, Centre for cognitive and computational Neuroscience. 2003. *An introduction to Neural Networks.*http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html [Accessed on 18 June 2007]

University polytechnic of Madrid. n.d. *Artificial Neural Networks*. http://www.gc.ssr.upm.es/inves/neural/ann1/anntutorial.html [Accessed on 15 May 2007]

Werbos, P. 1994. The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting, New York: Wiley.

Widrow, B. & Hoff, Marcian, E. 1960. Adaptive Switching Circuits, *1960 IRE WESCON Convention Record, New York: IRE* : 96-104

Widrow, B. & Lehr, A.M. 1990. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the Institute of Electrical Engineers and Electronics (IEEE),* 78(9):1415–1442.

Willamette University.1999. *Neural Network literature*. http://www.willamette.edu/~gorr/classes/cs449/backprop.html[Accessed on 10 July 2007]

Woinaroschy, A., Plesu, V. & Woinaroschy, K. 2001. Classification of neural networks based on genetic algorithm. Proceedings of the 6th World Congress of Chemical Engineers, Melbourne, 2001. [The Institute: 1-7].

9. LIST OF TABLES

10. LIST OF FIGURES