NEURAL NETWORKS APPLICATIONS IN ESTIMATING CONSTRUCTION COSTS

By
Khalil G. Rouhana

Thesis submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fullfillment of the requirements of the degree of

MASTER OF SCIENCE
in
Civil Engineering

APPROVED

J.M. De La Garza (CHAIRMAN)

M.C. Vorster

C. Schexnayder

A. Wiezel

June, 1994
Blacksburg, VA

NEURAL NETWORKS APPLICATIONS IN ESTIMATING CONSTRUCTION COSTS

By

Khalil G. Rouhana

Committee Chairman: J. M. De La Garza
Civil Engineering

(ABSTRACT)

This thesis deals with the potential application of neural networks technology to construction cost estimating problems. This is done by developing neural networks applications for a number of case studies constructed from the historical cost data of actual construction projects.

Parameter-based cost estimating applications, which require the application of analysis and prediction techniques to the cost data of a given estimating problem, were chosen as the major field of investigating the implementation of neural networks in this thesis. The objective of this thesis is to investigate whether or not neural network computing technology should be considered as a viable alternative in cost estimating applications by comparing it with conventional parameter-based analysis tools or predictive methodologies currently used to estimate construction costs. Both methodologies, parametric estimating and neural networks, use a parameter-based approach in modeling cost. However, the computational techniques used by the two methodologies to analyze cost data and produce results are significantly different.

Four case studies were the subject of comparison. The four case studies were compiled from the records of two construction companies and focus mainly on two areas: (1) Industrial projects and (2) Bridge construction.

ACKNOWLEDGEMENTS

I would like to thank Dr. Jesus De La Garza, the chairman of my advisory committee, for his numerous reviews of this document and his continual effort and guidance throughout this research. I would like to express my sincere appreciation to my advisory committee members Dr. Michael Vorster, Dr. Cliff Schexnayder, and Dr. Avi Weizel for their kind review of this document and their valuable remarks.

My committee made significant contributions to this thesis and to my education in this university in the past year and guided me in shaping my career objectives. I feel very fortunate to have worked with such fine people. I thank them all.

Finally, I give my greatest appreciation to my parents, Gabriel and Salwa Rouhana. They teached me the value of education and their support and encouragement allowed me to get this far in my academic career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In the United States, construction is nearly its largest industry. In 1986, the construction industry employed about 4.4 million people in all its phases; expenditures were 389 billion dollars, or 9 percent of the gross national product [Oglesby, 1989].

In these times of intensifying competition, construction companies are searching for new ways to enhance their performance and maintain their competitiveness in the market. In this context, artificial intelligence applications in construction are receiving more attention and getting more acceptance from practitioners in the construction industry today.

Project managers rely heavily on computing technology to plan, control, and execute projects [Ritz, 1986]. Traditional computing technology has given the project management professional a wide variety of tools supporting functions ranging from scheduling and estimating to detailed cost control. However, traditional computer modeling may not be sufficient for today's project manager who is faced with increasingly complex data, decreasing resources and increasing management demands. Neural network computing technology is a novel form of artificial intelligence(AI) [Caudill, 1992] that has tremendous potential for project management applications as well as for increasing the ability of the project manager to effectively execute projects.

The neural network approach to project management applications is quite simple. The historical data from past projects are presented to the network. The network, by an iterative process self-organizes and generalizes its own rules about those projects. This process is referred to as "network learning". When we present the network with a sufficient number of projects, the network will become a "trained network" capable of analyzing and predicting the performance of future projects of the same type of the projects on which the network was trained.

Traditional computing technologies utilize the strong arithmetical capabilities of computers. These technologies are fast and accurate when carrying out detailed instructions concerning numerical form data. However, these technologies are traditionally poor at intuitive types of problems that require the integration of past experience and which involve making decisions that cannot be clearly defined in mathematical form. The human brain is efficient with this type of computing and a neural network attempts to model the structure of the human brain and the methods the brain uses to process data [Wasserman, 1989]. Within recent years, neural networks have experienced a resurgence of interest as a paradigm of computation and knowledge representation [Garrett et al., 1992].

The increased interest in this new "form" of artificial intelligence is demonstrated by the number of research and application papers concerning neural networks, appearing in conferences [IEEE 1987, 1988]. Neural networks are finding applications in many industrial settings ranging from control systems [Ansaklis, 1990] to robotics [Lehay et al, 1991]. Neural networks take a different approach to AI, than the more traditional

techniques such as expert systems, by attempting to replicate the mechanism by which our brain manipulates data and reaches decisions [Alexander, 1989].

This thesis deals with the potential application of neural network technology to construction cost estimating problems. This is done by developing neural networks applications for a number of case studies constructed from the historical cost data of actual construction projects. Neural networks hold great promise for applications where data interpretation and based on this data future outcomes are predicted due to their strong predictive and learning capabilities. In this context, parameter-based cost estimating applications, which require the application of analysis and prediction techniques to the cost data of a given estimating problem, were chosen as the major field of investigating the implementation of neural networks in this thesis.

1.2 SIGNIFICANCE OF NEURAL NETWORKS APPLICATIONS IN COST ESTIMATING

Cost estimating is essentially a computational process which attempts to predict the final cost of a future project even though not all the parameters and conditions concerning this project are known or not fully quantified when the cost estimate is prepared. In general, estimating methods vary considerably depending upon the available information, the nature of the project, and the time available to prepare the estimate [Jelen et al., 1991].

The objective of this thesis is to prove whether or not neural network computing technology should be considered as a viable alternative in cost estimating applications by comparing it with conventional parameter-based analysis tools or predictive

methodologies currently used to estimate construction costs. Both methodologies, parametric estimating and neural networks, use a parameter-based approach in modeling cost. However, the computational techniques used by the two methodologies to analyze historical cost data and generate estimates are significantly different.

1.3 RESEARCH METHODOLOGY

This thesis examines the application of neural network computing technology to four case studies where preliminary, and sometimes detailed cost estimates, are carried out using traditional parametric estimating methods, like scaling and regression analysis. The four case studies to be used as applications for both, the neural network methodology and the parameter-based estimating techniques, are extracted from actual construction projects and the cost reports of these projects as documented by either the contractor or the owner.

The four studies are compiled from the records of two construction companies and focus mainly on two areas: (1) Industrial projects and (2) Bridge construction. The case studies are the following:

I. Estimating the cost of major industrial equipment of process plants based on preliminary equipment specifications.

II. Estimating the capital cost of process plants from the equipment list.

III. Estimating the material and installation costs of pressure vessels.

IV. Estimating the cost of bridges in roadway projects.

For each study, cost estimates will be performed by using one or more of the following parametric cost estimating techniques:

1. Equipment factor method.
2. Power law and sizing method.
3. Parametric estimating and regression analysis method.

A neural network application will also be developed for each of the four case studies and its results will be compared to those produced by either of the three parametric estimating methods. In addition, the cost estimates produced by both methodologies will be also compared to the actual cost.

After experimenting with the four case studies, the results achieved during that process will be discussed as well as outlining the benefits of using this novel form of computing technology. Then, we will summarize the findings and the contribution of this thesis to the body of knowledge of construction engineering and management. Finally, future applications and further research opportunities in this field will be outlined.

1.4 SCOPE AND LIMITATIONS

Four case studies will be the subject of comparison. A neural network application will be developed for each of the four case studies and its cost estimates results will be

compared with those produced by three other conventional parametric estimating methods.

The case studies as well as the estimating methods presented in this thesis do not cover all the estimating problems and analysis techniques of the construction industry. However, these problems do exist in a variety of construction projects and the application of neural network techniques to these problems in order to find better and more effective solutions is certainly a desired goal.

It should be noted here that there are some limitations that apply to the procedure and methodology used in the thesis. First, the geographic location factors will not be part of the analysis and cost will not be broken down into cost elements since we are dealing only with total cost estimates. The major goal of this thesis is eventually to prove a concept and not to develop a comprehensive approach dealing with the cost details or the cost breakdown structure.

1.5 PRESENTATION OF THE THESIS

The thesis is documented and presented in six chapters which form the body of this report.

Chapter 1 presents an introduction to the thesis, the background and significance of neural networks applications in cost estimating, and the research methodology.

Chapter 2 presents an overview of parameter-based cost estimating. It also includes a presentation of three methods commonly used in parametric estimating applications.

Chapter 3 introduces the concept of neural networks computing technology. It presents an overview of the theory and basic definitions of this novel form of computing technology. The basics of neural computing and the development of neural networks applications are presented .

Chapter 4 describes the research methodology. The programming of neural networks in Mathematica is discussed in this chapter as well as the preliminary research work dealing with two sample applications. In addition, the case studies presentation and parametric cost modeling are discussed.

Chapter 5 presents the implementation of neural networks and parameter-based estimating techniques to the four case studies and the process of generating cost estimates for these studies using the neural network and parametric estimating methodologies.

Chapter 6 presents a discussion of the results of analyzing the four case studies. A summary of the findings, conclusions, and recommendations for further research opportunities and future work are presented.

CHAPTER 2

PARAMETER-BASED COST ESTIMATING

2.1 INTRODUCTION

The literature review of this thesis covers two major subjects : (1) Parameter-based cost estimating and (2) Neural networks computing technology.

In this chapter we review the basic concepts and techniques used in parametric estimating as well as the typical applications of the parameter-based estimating methodology to a certain class of estimating problems. Three different methods, considered as commonly used techniques in typical parameter-based estimating applications, were investigated. These methods are the following:

1. The factor method.
2. Power law and sizing method.
3. Correlation and regression analysis method.

These methods are generally used to arrive at a preliminary cost estimate inexpensively and quickly with a reasonable accuracy of 15 to 30 % [Jelen et al., 1983]. Techniques referred to as, ratio estimating, parameter estimating, module estimating, Lang factor estimating, and percentage estimating involve substantially the same estimating methodology of the three methods outlined earlier.

2.2 PARAMETER-BASED COST ESTIMATING

The parametric method of cost estimating may utilize statistical techniques ranging from simple graphical curve fitting to multiple correlation analysis. In either case, the objective is to find a functional relationship between changes in cost and the factor or factors upon which the cost depends. Parametric cost estimating methods often lead to a mathematically fitted function called a cost estimating relationship (CER). A cost estimating relationship is a functional model that mathematically describes the cost of a structure, module or a system as a function of one or more independent variables [Ostwald, 1980]. The factor method, the power law and sizing method, and the correlation and regression analysis method are three typical cost estimating relationships applications.

2.2.1 The factor method

Factor estimating is a technique generally used to arrive at a preliminary capital cost estimate quickly with a reasonable accuracy of ±30 % [Sinha,1988]. The factor method is a basic and important method in parameter-based estimating. Other terms such as ratio, parameter and percentage methods are about the same thing. This method is commonly used to obtain a quick estimate of the capital cost of a chemical or industrial plant based on the equipment list. Essentially the factor method gives an estimate of the fixed capital investment, or in some cases the total capital investment estimate by multiplying the delivered or purchased equipment cost by a factor.

The earliest of these methods was the Lang factor method which suggested that the total plant cost could be obtained by multiplying the delivered cost of equipment by a single factor depending on whether the plant processes solids, fluids or both. Improvements and refinements have been made to increase the accuracy without running into extensive computations. A recent improvement in equipment factors has been proposed by [Cran, 1981]. The breakdown in types of equipment is more extensive, a distinction is made in factors for materials of construction, and the instrument and indirect costs are factored separately. Cran's cost estimating relationship is :

$$C = \left(\sum EF_D + IF_I\right) * (1 + F_O) \qquad [2.1]$$

Where C = Total plant cost

E = The purchase cost of a particular type of equipment.

F_D = The direct cost factor that varies with the different type of equipment and its material of construction.

I = The sum of the costs of all the instruments.

F_I = The direct cost factor of the instruments

F_O = The indirect cost factor

There are practical considerations in applying the factor method. Variations between estimates and the level of accuracy achieved by the factor method are highly dependent on the following:

1. Size of the basic equipment selected;

2. Materials of construction;
3. Operating pressures, temperatures;
4. Processing technology or plant type such as fluids processing, fluids-solids processing or solids processing;
5. Location of plant site;
6. Timing of construction.

Traditionally, the accuracy range assigned to the estimate prepared from the first equipment lists has been ±30 % compared to the final cost [Sinha, 1988]. However, if the equipment list is, indeed, accurate, the accuracy of the estimate will be a function of the cost correlations available for the equipment and of the factors used for translating the equipment cost to the total capital cost. For example, the direct cost factor assigned to pressure vessels is 2.8 if their material of construction is carbon steel and 1.7 if it is stainless steel [Cran, 1981]. Thus, depending on the accuracy of these factors, the overall accuracy can be significantly better than ±30 % .

2.2.2 Power law and sizing model

The power law and sizing model is frequently used for estimating equipment costs. This model deals mainly with equipment designs varying in size but similar in type. The basic CER used in the power law and sizing model is given as:

$$C = C_r\left(\frac{Q_c}{Q_r}\right)^m \qquad [2.2]$$

Where C = Cost estimate to be obtained for design Qc

Cr	=	Known cost for a reference size Qr
Qc	=	Design size
Qr	=	Reference design size
m	=	Correlating exponent

In the past , a common value for *m* has been taken as 0.6, or what is referred to as the six-tenths rule. However, the application of the 0.6 rule of thumb for most purchased equipment is an oversimplification since the actual values for the correlation exponent m vary from less than 0.2 to greater than 1. Thus, the 0.6 factor should only be used in the absence of any other information. The exponent *m* could be determined by plotting actual construction costs for the equipment versus the equipment size on a log-log paper. The slope of the resulting line will be the cost correlation exponent *m*. An example on the calculation of the exponent m for the cost of Shell & Tube Heat Exchangers expressed as a function of the external heat exchange area is illustrated in Figure 2.1. Cost index factors could be applied to this method in order to express all the cost figures in dollar values of a base year (like 1979 in the case of the heat exchangers example).

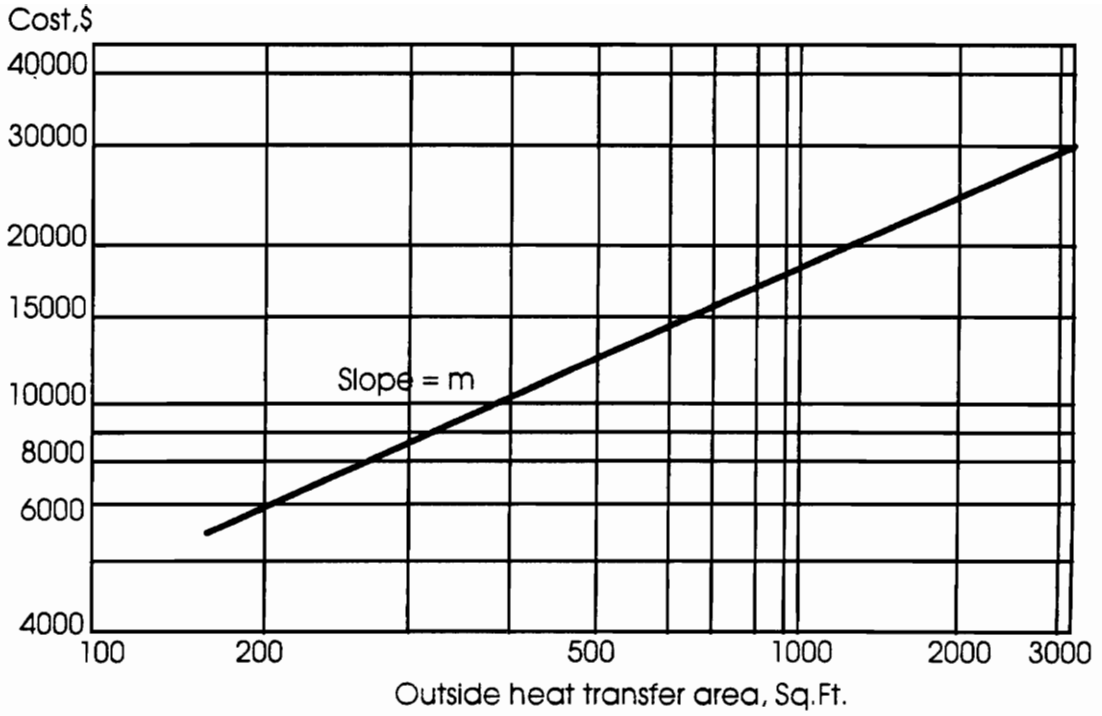


Figure 2.1 Calculation of the "m" exponent for Shell & Tube Heat Exchangers [Peters et al., 1980]

2.2.3 Correlation and regression analysis method.

In general, regression analysis and correlation techniques use the least-square regression method in computing the coefficients of a given cost estimating relationship (CER). Linear and non-linear models could be used. Basically, this method is used to establish a CER based on the general cost model :

$$C = F(x1, x2, .., xn) \qquad [2.3]$$

where x1,x2, .., xn are the cost variables, and C is the cost value.

For linear models, multiple regression analysis would lead to a linear cost function of the form:

$$C = a_0 + a_1 x_1 + a_2 x_2 + \ldots\ldots + a_n x_n \qquad [2.4]$$

Where $a_0, a_1, .., a_n$ are the coefficients of the cost variables.

For non-linear models, multiple regression analysis would lead to a non-linear function of the form:

$$C = a_0 + a_1 x_1^{p1} + a_2 x_2^{p2} + \ldots\ldots + a_n x_n^{pn} \qquad [2.5]$$

where $p_1, p_2, .., p_n$ are the exponents of the correlated cost variables and $a_0, a_1, .., a_n$ are their coefficients.

In the case of process plants, the regression method suggests that more than one independent variable (like process equipment cost) be used in developing the cost factors based on historical construction data. The general form of the cost estimating relationship (CER) used in the regression method is as follows:

$$C = \left(\sum_j \sum_i F(x_i) \right) (1 + F_o) \qquad [2.6]$$

Where C = Cost of project being evaluated

x_i = Independent cost variable.

$F(x_i)$ = Regression function for estimating the direct cost elements (Buildings, instrumentation, piping, equipment

installation, etc.) as a function of a number of independent cost variables.

F_O = Multiplying factor for indirect costs and expenses such as engineering, contractor's profit and contingency.

j = 1..n, factor index for n direct cost elements.

i = 1..m , Correlation index for m independent cost variables.

In this method, it was hypothesized that many of the direct cost elements could be correlated more appropriately with independent variables, other than the process equipment cost. For example, piping for a vessel generally runs the width and height of the vessel; therefore, its cost should be correlated with the volume of the equipment. The total process piping cost would then depend on the total volume of the equipment as well as the number of pieces of equipment. The cost of instrumentation, on the other hand, is not very sensitive to the volume and cost of the equipment and it could be directly correlated with the number of pieces of equipment.

Following this methodology, different regression functions could be developed for the major direct cost elements, the sum of which will give us an estimate of the direct cost. The direct cost is then multiplied by $(1+F_O)$ to give us an estimate of the total capital cost of the plant.

CHAPTER 3

NEURAL NETWORKS

3.1 INTRODUCTION

This chapter presents an overview of neural networks applications, the theory and concepts of neural computing as well as the learning algorithms and the development process of neural networks applications. In general, neural networks could be looked upon as massively parallel systems that rely on dense arrangements of interconnections and surprisingly simple processors. Neural networks provide an effective approach for a broad spectrum of applications ranging from control systems [Ansaklis, 1990] to robotics [Lehay et al., 1991]. Neural nets excel at problems involving patterns like pattern mapping, pattern completion, and pattern classification. Typical applications include translating images into keywords, translating financial data into financial predictions, or mapping visual images into robotics commands [Dayhoff, 1990].

3.2 BASICS OF NEURAL COMPUTING

Neural networks, as the name implies, are loosely modeled after the biological structure of the brain [Garrett, 1992]. The human brain consists of tens of billions of computing units called neurons. Each neuron can be considered as a single microprocessor that receives signals, performs some form of processing on the signals, and then transmits another signal.

A diagram of a nerve cell (neuron) typical of those in the human brain is shown in Figure 3.1. The output area of the neuron is a long, branching fiber called the axon. The input area of the nerve cell is a set of branching fibers called dendrites. The connecting point between an axon and a dendrite is the synapse. When a series of impulses is received at the diendritic areas of a neuron, the result is usually an increased probability that the neuron will fire an impulse down its axon in the direction of other neurons dendrites (inputs).

An artificial neural network can be constructed using conventional computing elements to mimic the analog action of a series of neurons [Wasserman, 1989]. In the artificial neural network the unit analogous to the biological neuron is called "processing element" (PE).

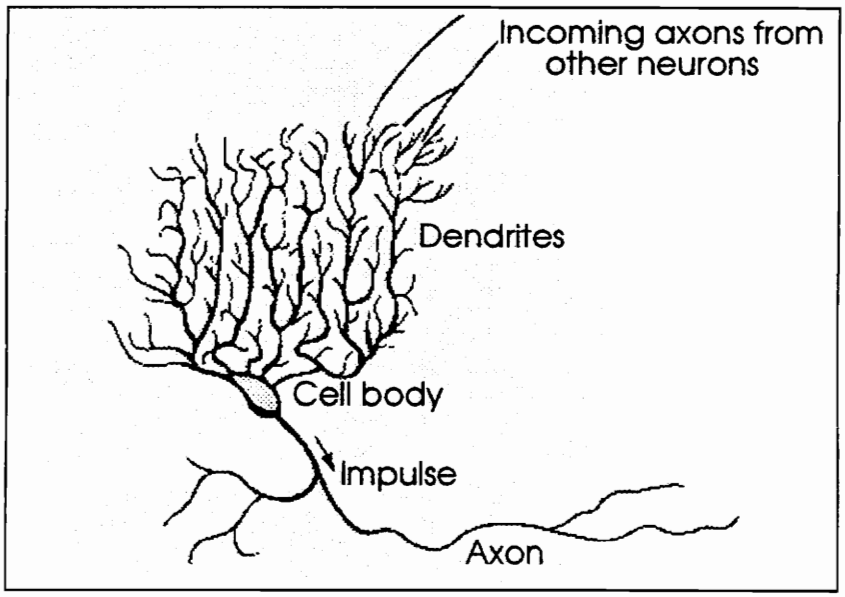
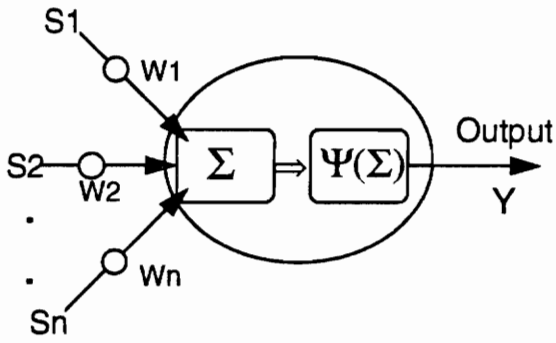


Figure 3.1 Schematic drawing of a biological brain cell (Dayhoff, 1990)

A neural net processing element can have many input paths and combines usually by summation, the values of the input signals. The result of the summation translates into an internal activity level for the processing unit which will output a signal according to rules dictated by a "transfer function" [Caudill, 1992]. The transfer function is generally a threshold level of the processing unit activity at which the unit will output a signal. The summation and transfer functions of a typical processing element are illustrated in Figure 3.2.



S1, S2, Sn - Input signals

W1, W2, Wn - Input signal weights

Ψ - Transfer function

$Y = \Psi(\Sigma)$ - Output signal

Figure 3.2 - Typical neural network processing element

Although a single processing element can perform simple pattern detection functions, the power of neuro-computing comes from connecting processing elements into networks. A multi-layer neural net is normally constructed by arranging processing units in a number of layers. The output of a single layer provides the input to the subsequent layer. The strength of the output is determined by the connection weights between the processing units of two adjacent layers [Wasserman, 1989].

We can look at the inputs to a single processing element and the weights on the inputs as vectors, with components (S1, S2, .., Sn) and (W1, W2, .., Wn), respectively. Thus, the total input signal is the dot product of the weight vector and the input vector. Mathematically the dot product is equivalent to $|w||s| \cos \beta$ where $|w|$ and $|s|$ are the magnitudes of the vectors **S** and **W** and β is the angle between the two vectors (Figure 3.3).

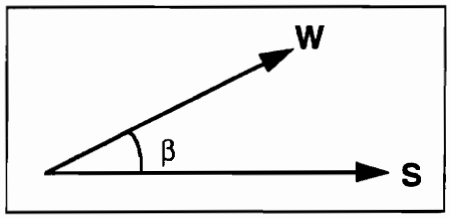


Figure 3.3 Weight and input vectors

This is a very good visual image that tells us a lot about neural networks' operations. Since the PE total input is equal to the dot product, then this input will be largest when the weight vector and the input vector are very close to each other (β is very small ≈ 0). Suppose we had four different PEs having the same input vector but different sets of weights on the input vector as shown in the neural net of Figure 3.4.

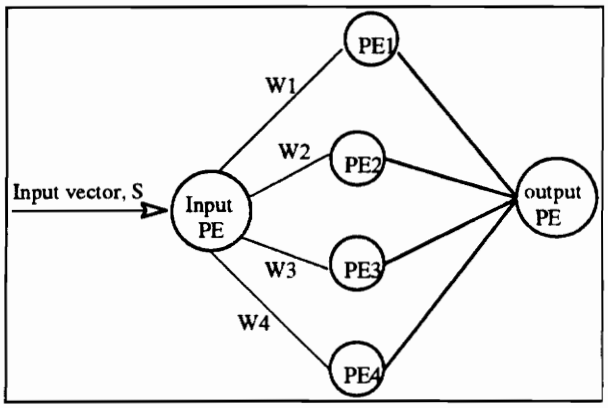


Figure 3.4 Single input neural network

What we have here are four weight vectors {W1, W2, W3, W4} pointing out in the weight space illustrated here by a circle and one input vector S as shown in Figure 3.5.

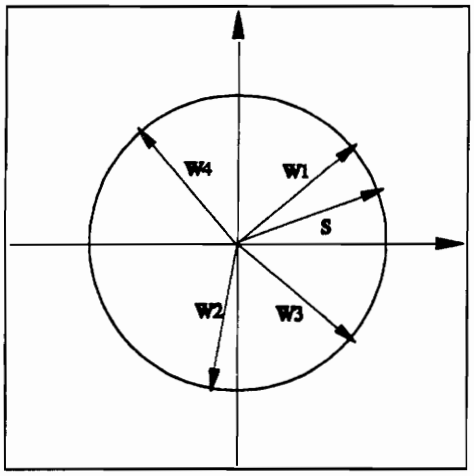


Figure 3.5 Spatial distribution of weight and input vectors

It can be seen that the processing element that will have the highest weighted input and therefore the most influence on the final output of the network is the one with the weight vector pointing most nearly in the direction of the input vector.

We already have a primitive but useful system here. Let's imagine we want to use this system to recognize four distinct patterns, all we have to do is set up each of the four PEs with a weight vector pointing out in the direction of (parallel to) one of the patterns we want to recognize. Then we present the PEs with an input signal from some unknown sample, the PE with best match will fire with the greatest strength and thus will tell us which pattern the input most closely resembles. For this particular example, if we imagine the output of a PE is connected to a light bulb of a certain color and each color is associated with a given input (Figure 3.6), then when we present the PEs with an

unknown input, the net weighted input signals will behave as a voltage that will light up the bulb connected to the PE having a weight vector closest to the input vector.

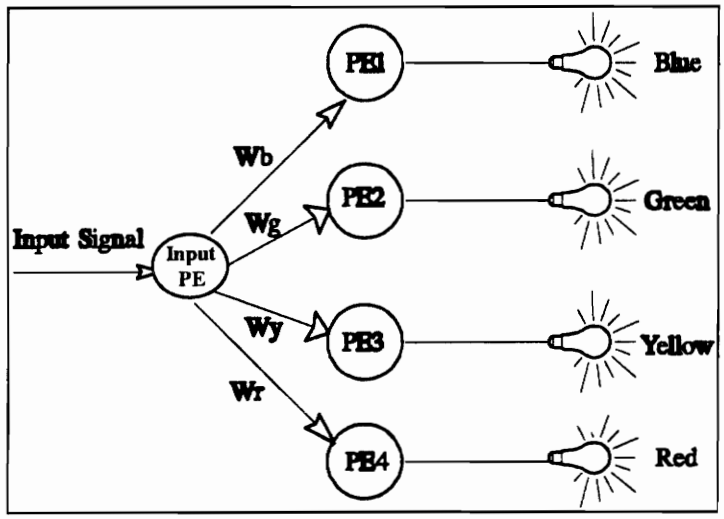


Figure 3.6 Light bulb example - Pattern recognition

The weighted input signal, also known as NET, to a particular PE is usually further processed by a transfer function Ψ to produce the PE's output signal, Y. In many cases, the transfer function is chosen so that Y never exceeds some low or high limits regardless of the value the net input signal (NET) and such transfer function is called a "squashing function" [Wasserman, 1990]. The squashing function is often chosen to be a "sigmoidal" function (S- shaped) and it is expressed mathematically as $F(x) = \frac{1}{1+e^{-x}}$. The PE's output signal can then be expressed as: $Y = \Psi(NET) = \frac{1}{1+e^{-NET}}$. The significance of this transfer function is that it allows the network to handle any range of input signals (large or small) by squashing their values and producing outputs which always fall within a controlled range. Other commonly used transfer functions (Squashing and non-squashing) are hyperbolic tangent (Tanh) and hyperbolic sine (Sinh) functions. Non-squashing functions

could be used when the range of the input or output values of the network normally fall within a pre-specified range. Plots of the sigmoid, hyperbolic tangent and hyperbolic sine functions are shown in Figure 3.7.

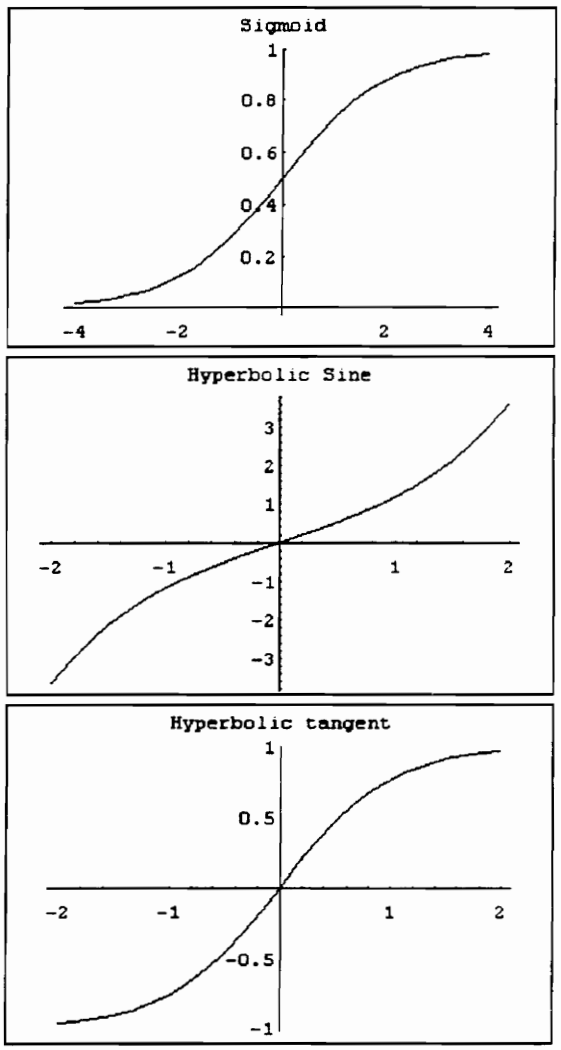


Figure 3.7 Transfer functions

The output path of a processing unit can be connected to the input of one or many other processing units. The strength of the output signal is determined by the "connection weight", Wij, between processing unit (i) and processing unit (j) [Wasserman, 1989].

In general , a neural network comprises :

a- An input layer.

b- A number of hidden layers.

c- An output layer.

In its simplest forms, a neural network is composed of three layers: an input, a hidden and an output layer. Such simple architecture is illustrated in Figure 3.8.

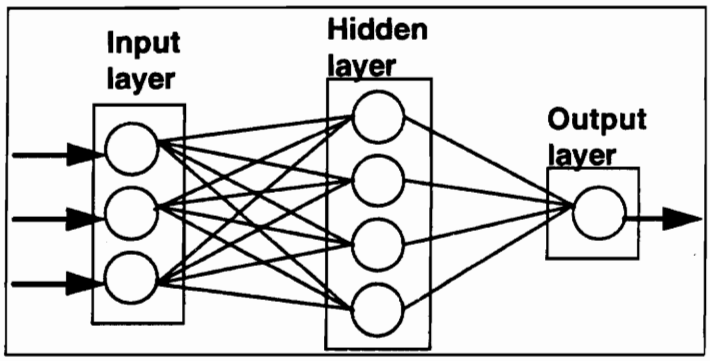


Figure 3.8 Network architecture

3.3 NETWORK LEARNING AND BACK PROPAGATION

One of the most important characteristics of neural networks that excited so many people is their ability of learning and self-organization [Garrett et al., 1992]. The objective of the learning process is to train the network so that the application of a set of inputs produces the desired (or at least a consistent) set of outputs.

predetermined procedure. During training, the network weights gradually converge to values such that each input vector produces the desired output vector.

In general there are two types of learning or training in a neural network:

- Supervised training.
- Unsupervised training.

In supervised training both the inputs and outputs for a data set are presented to the network during the learning process. It requires that each input vector be associated with a target vector representing the desired output, together these are called training pairs. Usually a network is trained over a number of such training pairs for a number of learning "cycles." A learning cycle starts with applying an input vector to the network then it is propagated in a forward propagation mode which ends with an output vector. The resulting output vector is then compared to the corresponding target vector. Next the network evaluates the errors between the desired output vector and actual output vector. It uses these errors to shift the connection weights according to a "learning rule" that tends to minimize the error. This process is generally referred to as "error back propagation." The adjusted weights are then used to start a new cycle.

A typical back propagation cycle in a neural network is illustrated in Figure 3.9. For a certain number of cycles the weights are shifted until the deviations from the desired outputs are minimized.

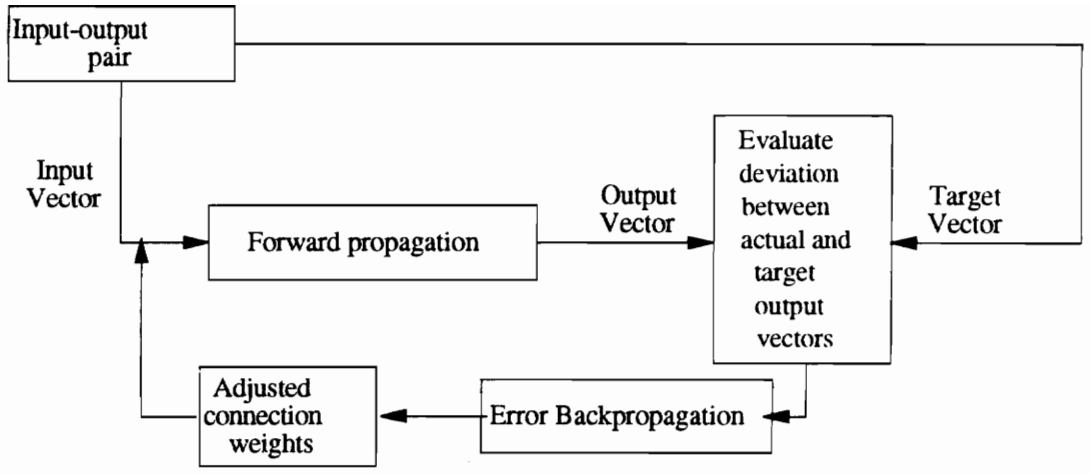


Figure 3.9 Typical back propagation cycle in a neural network

Back propagation learning in a neural network very much resembles the "tuning" process of a guitar. The guitar strings represent the connections, and the string tensions represent the connections weights. Each time we strike a string to produce a new sound (actual output). We evaluate how close or how far this sound is from what is desired and use this deviation to make a new tension adjustment in a string (adjust weights). We keep on striking the strings with the new tensions (learning cycles) until we get as close as possible to the desired sound. At this point, the guitar is tuned (trained network), and ready to play.

Unsupervised training , presented by [Kohonen, 1984] and many others, requires no target vectors for the outputs and hence, no comparisons for predetermined ideal responses. The training set consists solely of input vectors. The training algorithm modifies network weights to produce output vectors that are consistent; that is, repeated applications of one of the training vectors or the application of a vector that is sufficiently similar to it will produce the same pattern of outputs. In short, the training process

extracts the statistical properties of the training set and groups similar vectors into classes. One disadvantage for unsupervised learning is that the network has no chance to assess how much error the connection weights are creating within the system. In this thesis, we dealt with supervised learning and the "generalized delta rule" [Rumelhart et al., 1986] as the learning rule in the back propagation process.

3.3.1 The Generalized delta rule

One alternative for supervised learning, developed by Rumelhart, Hinton and Williams, is based on the "generalized delta rule" [Rumelhart et al., 1986]. This rule is one of the most commonly used learning mechanisms in back propagation neural networks. It is the rule which is used to adjust the weights of input signals before another learning cycle is started. In this section, the formal mathematical description of back propagation neural nets' operation, via the "generalized delta rule", is presented.

A back propagation neural network is a layered, feed-forward network that is fully interconnected by layers. Thus, there are no connections that bypass one layer to go directly to the other layer. Although three layers are used in the discussion, the same principles apply to networks with more than three layers (i.e., more than one hidden layer).

In general, back propagation neural networks act as mapping networks in situations when we use such networks to establish a functional relationship between their input and their output. For example, if the input to a network is the intensity of the solar energy I at a given point X, collected by a plate P, and the output is the useful energy E resulting from the process, then the network performs the mapping $\{I, X, P\} \rightarrow \{E\}$. This

mapping function becomes increasingly important when we want to perform a very complicated mapping where we don't know how to describe the functional relationship in advance. In this situation, the power of a neural network to discover a mapping function between its inputs and outputs is extremely useful.

Suppose we have a set of vector pairs, { (X1,Y1), (X2,Y2), .., (Xn,Yn) } which are examples of a functional mapping $Y = \Phi(X)$, where Φ is the mapping function that transforms X into Y. We want to train the network so that it will learn an approximation:

$$Y' = \Phi'(X) = 0 \qquad [3.1]$$

Where Y' and $\Phi'(X)$ are the first order derivatives of Y and Φ with respect to X. The generalized delta rule is the learning rule that will enables us to find the appropriate set of weights that fulfill the conditions imposed by Eq. [3.1]. To begin, let's state the equations for forward propagation in the three-layered network of Figure 3.10. An input vector, $X_p = \{x_{p1}, x_{p2}, .., x_{pn}\}$, is applied to the n-unit input layer of the network. The input units distribute the values to the hidden layer (h).

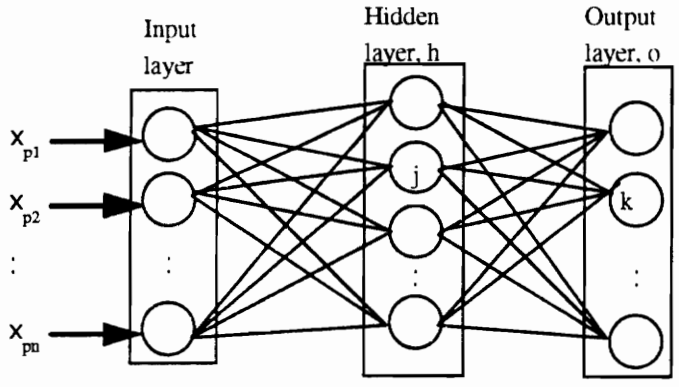
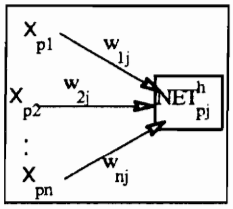


Figure 3.10 Three-layered neural network

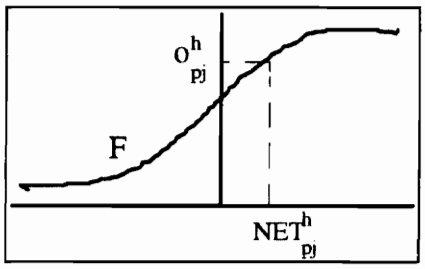
Then, the net weighted input to the j^{th} unit of the hidden layer is given by:

$$NET_{pj}^{h} = \sum_{i=1}^{n} W_{ij} x_{pi} \qquad [3.2]$$



The NET value computed in [3.2] is then processed by a "Transfer Function" which converts it to an output signal as follows:

$$o_{pj}^{h} = F\left(NET_{pj}^{h}\right) = i_{pj}^{o} \qquad [3.3]$$



where i_{pj}^{o} is the input to the output layer (o) from the j^{th} unit in the hidden layer (h). The computations performed in the j^{th} processing unit of the hidden layer according to [3.2] and [3.3] are illustrated in Figure 3.11.

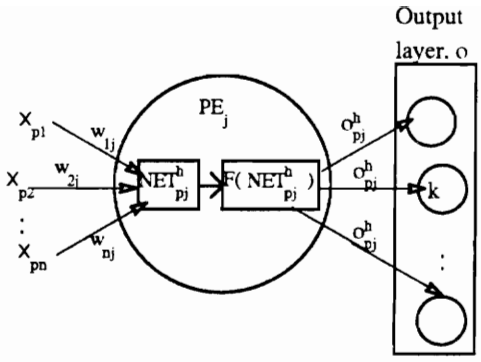


Figure 3.11 Typical hidden layer processing element

Similarly, the net weighted input to the k^{th} unit of the output layer is given by:

$$NET_{pk}^{o} = \sum_{i=1}^{n} W_{jk} i_{pj} \qquad [3.4]$$

If we assume that the same transfer function of the hidden layer units is also used in the output layer units, the k^{th} unit output will be:

$$O_{pk}^{o} = F\left(NET_{pk}^{0}\right) \qquad [3.5]$$

The error back propagation starts with computing the deviation between the network's output O and the desired output Y. Using this notation the error for the p^{th} input vector and the k^{th} output unit is given by:

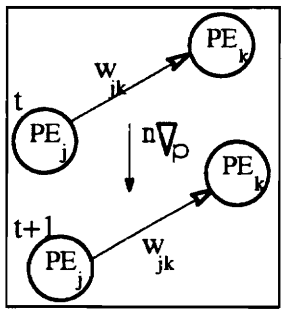
$$e_{pk} = y_{pk} - o_{pk} \qquad [3.6]$$

The error that is minimized by the generalized delta rule is the sum of the squares of the errors for all the output units:

$$E_p = \frac{1}{2}\sum_{k=1}^{m} e_{pk}^2 \qquad [3.7]$$

Where m is the number of PEs in the output layer. The factor of 1/2 in [3.7] is there for convenience in calculating derivatives later. To determine the direction in which to change the weights of the connections leading to the output layer, we calculate the negative gradient of E_p, ∇E_p, with respect to the weights W_{jk}. Then we can adjust the values of the weights in such a way that total error is reduced. As far as the magnitude of the weight change is concerned, we take it to be proportional to the negative gradient.

Thus, the weights leading into the output layer are updated according to:



$$W_{jk}^{o}(t+1) = W_{jk}^{o}(t) + \eta \nabla_p W_{jk}^{o}(t) \qquad [3.8]$$

Where η is the learning coefficient parameter. This coefficient is nothing but the amount of the gradient with which the weights are changed. Equation [3.8] is normally referred to as the *weight update equation.*

The negative value of the gradient $\nabla_p W_{jk}^0$ is computed as follows:

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)^2 \qquad [3.9]$$

$$\frac{\partial E_p}{\partial W_{jk}^o} = -\left(y_{pk} - o_{pk}\right)\frac{\partial F \partial\left(NET_{pk}^o\right)}{\partial\left(NET_{pk}^o\right)\partial W_{jk}^o} \qquad [3.10]$$

$$\frac{\partial\left(NET_{pk}^o\right)}{\partial W_{jk}^o} = i_{pj}^o = o_{pj}^h \qquad [3.11]$$

$$\nabla_p W_{jk}^o = -\frac{\partial E_p}{\partial W_{jk}^o} = \eta\left(y_{pk} - o_{pk}\right)F'\left(NET_{pk}^o\right)i_{pj}^o \qquad [3.12]$$

In the case where the transfer function is the sigmoidal function $F\left(NET_{pk}^o\right) = \frac{1}{\left(1 + e^{-NET_{pk}^o}\right)}$, the derivative of that function is F(1-F), then we will have:

$$F'\left(NET_{pk}^o\right) = o_{pk}\left(1 - o_{pk}\right) \qquad [3.13]$$

Substituting [3.13] in [3.12], and then replacing the value of the gradient in [3.8], we will get the *weight update equation* for the weights connecting the hidden layer units to the output unit k as follows:

$$W_{jk}^o(t+1) = W_{jk}^o(t) + \eta o_{pk}\left(1 - o_{pk}\right)\left(y_{pk} - o_{pk}\right)i_{pj}^o \qquad [3.14]$$

We want to summarize the weight update equation by defining the quantity:

$$\delta^{o}_{pk} = o_{pk}(1-o_{pk})(y_{pk}-o_{pk}) \qquad [3.15]$$

We can then write the *weight update equation* for any unit k on the output layer as:

$$W^{o}_{jk}(t+1) = W^{o}_{jk}(t) + \eta\delta^{o}_{pk} i^{o}_{pj} \qquad [3.16]$$

The same type of calculation is done for the hidden layer units. Here again, we define the following quantity:

$$\delta^{h}_{pj} = F'(NET^{h}_{pj})\sum_{k}\delta^{o}_{pk}W^{o}_{jk} \qquad [3.17]$$

Then the weight update equation for the hidden layer units becomes analogous to those for the output layer:

$$W^{h}_{ij}(t+1) = W^{h}_{ij}(t) + \eta\delta^{h}_{pj}x_{pi} \qquad [3.18]$$

Where x_{pi} is the input from the i^{th} input layer unit to the j^{th} hidden unit.

The error back propagation process in a neural network using the generalized delta rule is a gradient-descent technique whereby the weights are shifted in the direction of the minimum error in proportion to the negative value of the gradient $\nabla_{p} W^{0}_{jk}$ and the learning coefficient η. In general, the network is trained using a number of examples where the weights are updated using the generalized delta rule as each training example is processed.

The objective here is to adjust the weights in a way to reach the global minimum of the sum of the squares of the errors, E_p, of all the learning examples.

Figure 3.12 gives an illustration of the gradient descent process towards the global minimum error of a given training set over a hypothetical error response surface plotted as a function of two connection weights. Zmin is called the global minimum, and Z is the current value of the error sum, E_p, that is reduced towards Zmin by the amount of the negative gradient ∇E_p.

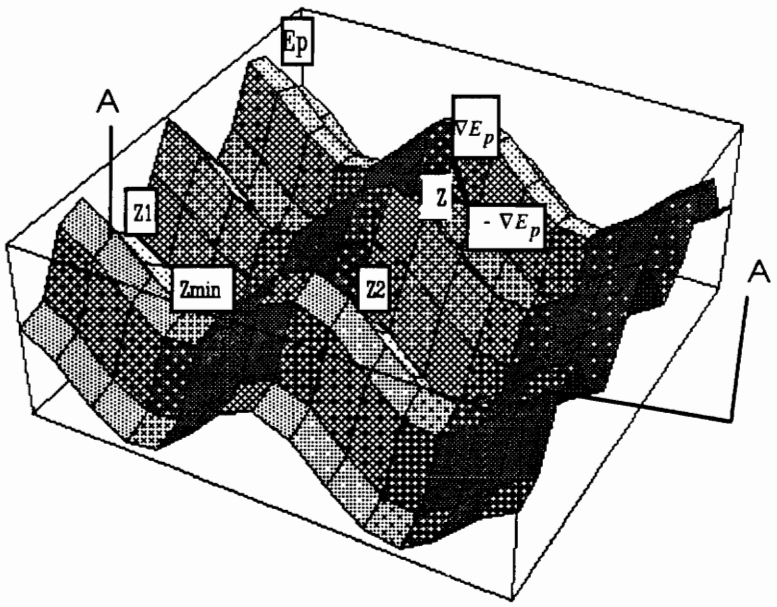


Figure 3.12 Hypothetical error response surface in weight space

Notice, however, that there are other minimum points, Z1 and Z2. A gradient descent search for the global minimum might accidentally hit one of these local minima instead of the global minimum. A cross section of the hypothetical error response surface

(along A-A) shows the local and global minimum locations in weight space as illustrated in Figure 3.13.

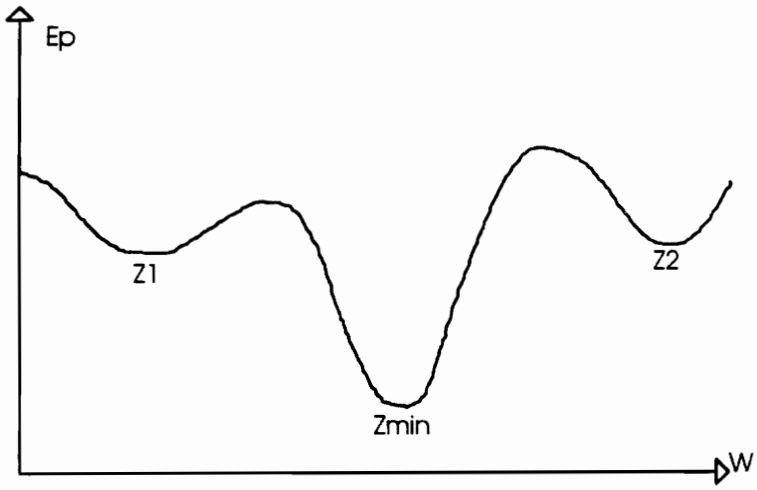


Figure 3.13 Cross section A-A of the hypothetical error response surface

We note here, that once a network settles on a minimum, whether local or global, learning ceases. If a local minimum is reached, the error at the network outputs may still be unacceptably high. Fortunately, this problem does not appear to be much of a difficulty in practice. If a network stops learning before reaching an acceptable solution, a change in the number of hidden nodes or in the learning parameters will often fix the problem [Freeman, 1992]. When a network reaches an acceptable solution from an error standpoint, it does not matter if the minimum is global, local or even if training was halted before the true minimum was reached.

3.3.2 Back propagation learning in a neural network

The learning process of a multi-layered back propagation neural network, via the generalized delta rule is an iterative process [Rumelhart et al., 1986]. Each step involves the determination of error associated with each processing unit and then the modification of weights on the connections coming out of each unit.

Each presentation of one training case and subsequent modification of connection weights is called a *Cycle* [Garrett et al., 1992]. A typical back propagation neural network is illustrated in Figure 3.14. It can be seen how the input signals are propagated forward through the processing units of the hidden and output layers, and errors between the network's outputs and the desired outputs are propagated backwards. The purpose of the back propagation of error is to adjust the connection weights of the network before the start of a new cycle in a way that minimizes the errors at the end of this cycle.

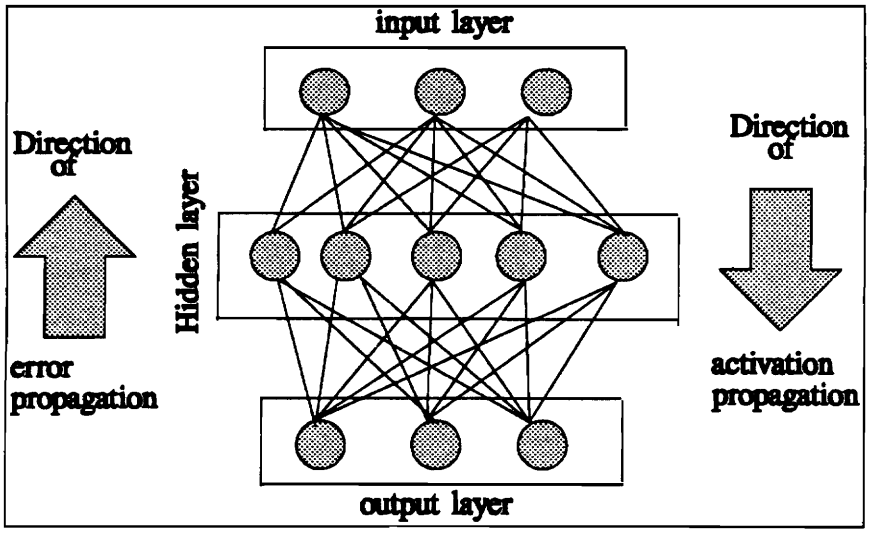


Figure 3.14 Sample Back propagation neural network

An interesting remark to note here is that the learning rule (the generalized delta rule) allows some flexibility in training the network by manipulating the value of the learning coefficient (η). The learning coefficient is nothing but the fraction of gradient that determines the amount of shifts in the value of the interconnection weights at the end of each cycle. The higher the value of η the faster the change in the value of the connection weights and in some instances a high value of η may result in overshooting the solution space where the network exhibits its best performance. The learning coefficient also affects the rate of convergence. If η is large (> 0.5) the speed of convergence is fast but this may cause "over-shooting" as mentioned earlier. If η is small (< 0.2), the weights will be changed in smaller increments thus causing the system to converge more slowly, but with little oscillation.

3.3.3 Developing a neural network application using a back propagation algorithm

After finding a problem that fits into a pattern association framework and for which a fairly good number of examples exist, the development of a back propagation neural network for that problem involves the following:

1. The selection of the network architecture.

2. The selection of the cases on which the network is to be trained (The network cannot learn what it is not exposed to).

3. The actual training of the network to produce the expected output for a given set of inputs.

4. The determination of the ability of the network to generalize when presented with patterns on which it was not explicitly trained.

When a new network is to be trained, its initial connection weights are randomized. If they were all set to the same number, the generalized delta rule, which back propagates error in proportion to the weights, would not be able to change the relative proportion of the weights and thus never leaves the starting point. The learning process is repeated for a given number of cycles, or until the error for all patterns is below a pre-specified value. The pattern associations learned by the network are embodied in the strength of the connections between the processing elements. These weights can be saved and reloaded representing a "programmed" or "trained" network that could be used to propagate activation again without any training taking place.

The final and last step in the development of a neural network is to determine how well the network performs on input patterns for which it was not trained. This is basically a test to see how well the network has discovered features and sub-features in the training cases and the strength of its predictive capabilities in handling new cases. One must be careful not to present to the network input patterns possessing features that were not present in the training cases, for the results will most likely be discouraging. What is most often done in the testing and training of a back propagation neural network is to take the body of cases collected for a given problem and randomly divide them up into training and test cases, with the hope that the training cases randomly selected have the features and

sub-features present in the test cases. If a particular test case is not handled properly by the trained network, it is an indication of some untrained feature and that case is then added to training set.

3.4 CONCLUSION

Within the literature describing neural network computing technology, one can sense the benefit that this technology can bring to engineering related applications. In situations where outcomes are predicted, like cost estimating for example, neural networks hold great promise for this type of applications. In this context, parameter-based cost estimating applications can benefit a great deal from the use of neural networks in analyzing cost data and making predictions. The neural net eliminates the need to find a good cost estimating relationship and simplifies the parametric cost modeling process.

CHAPTER 4

METHODOLGY AND CASE STUDIES PRESENTATION

4.1 INTRODUCTION

The general approach as well as the means and methods that were used to achieve the goals of this thesis are outlined through the following steps:

1. Programming the network in Mathematica.
2. Preliminary work.
    - i- Estimating costs of a piping section.
    - ii- Estimating costs of bridges.
3. Case studies presentation.
4. Parametric cost modeling of case studies.

4.2 PROGRAMMING THE NEURAL NET IN MATHEMATICA

Any researcher or student of neural networks will find several tools with which to conduct their work. One approach is to use a commercially available software package that comes with predefined neural network architectures and a tutorial, like NeuroForecaster, WinBrain and Neuralyst. Another approach is to code networks directly in high level languages such as C or PASCAL. Programming neural networks in Mathematica 2.2 for Windows could be considered as an intermediate approach for

experimenting with neural networks. This approach lies closer to the programming approach than it does to the prewritten, commercial-software approach.

The program that will be developed in Mathematica to perform neural net computations will enable us to perform the following:

1. Network learning.
2. Testing and evaluation of trained networks.

For any given problem, the data will be split into a learning set and a testing set. The program will require the user to do the following:

a- Define the number of inputs and specify the range of each input.
b- Specify a value for the learning coefficient (η).
c- Define the number of processing elements in the hidden and output layers.
c- Specify the number of cycles for each run.
d- Choose between :
    i- Saving the current model.
    ii- Performing a new run.

4.2.1 Mathematica basics

Mathematica is a software package self-described as "A system for doing mathematics by computer" [Wolfram,1991]. It includes more than 750 functions, including numeric, symbolic, and graphical functions as well as all of the standard constructs found

in most high-level construct languages. In addition, the software enables the creation of user-defined functions either directly or by writing programs for defining more sophisticated functions.

Let's set down a few Mathematica notations that will be used to construct the learning and test functions of the neural network program:

- Arguments to functions are surrounded by square brackets, as in Cos[0.5 Pi]

- Lists are enclosed in curly brackets, as in {3, 4, 5, 6}. A simple list, such as {1, 2, 3} can be treated like a vector.

  A matrix is constructed by nesting a number of lists; for example:

  Matrix1 = { {1,2,3}, {4,5,6}, {7,8,9} }; This is equivalent to $\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$

- The addition of a semicolon at the end of an expression suppresses the output to the screen from the Mathematica interpreter.

- The assignment operator, :=, results in a delayed assignment. We use this symbol when defining a function; for example Sigmoid [x_] := 1/(1+E^(-x)). Sigmoid will not have a value until we specifically ask for an evaluation like Sigmoid [2] for example.

- The symbol x_ means any expression given the name x. For example x_ was expressed by 2 in the Sigmoidal function above.

- Matrix elements or lists are identified by indices within double brackets; for example:
  Matrix1 [[2]] gives {4,5,6},
  Matrix1 [[2,3]] gives 6.

- The dot product of two lists or vectors is obtained by placing a period in between; for example the dot product of vectors X = {1,2} and W= {3,4} is obtained by X.W .

Complete listings of Mathematica functions, notations and tools are found in [Wolfram, 1991].

4.2.2 Implementing a back propagation neural network in Mathematica

The programming of the neural net in Mathematica requires the representation of the basic elements of the network in Mathematica format or notations. First, the input and output variables are grouped together in a matrix row referred to as the "input-output" pair. This pair consists of two vectors: (1) input = $\{x_1, x_2, \ldots, x_n\}$ and (2) output = $\{y_1, y_2, \ldots, y_m\}$. In general, an input-output pair in Mathematica has the following form:

$$\text{Input-output pair} = \{\{x_1, x_2, \ldots, x_n\}, \{y_1, y_2, \ldots, y_m\}\} \qquad [4.1]$$

The input-output matrix will then have m rows corresponding to m input-output pairs and will be of the general form:

$$IO-Matrix=\begin{Bmatrix}\{\{x_{11},..,x_{1n}\},\{y_{11},..,y_{1m}\}\},\\ .\\ .\\ \{\{x_{p1},..,x_{pn}\},\{y_{p1},..,y_{pm}\}\}\end{Bmatrix} \quad [4.2]$$

Two input-output matrices will be used in the Mathematica program: (1) learning pairs matrix and (2) test pairs matrix.

The next step will be to define a transfer function that transforms the dot product of the input and weight vectors at any processing element to the output of that element. Since back propagation algorithms require that the transfer function be continuous and differentiable at every point of the solution space, the sigmoidal function was a natural choice to be used as our transfer function. In Mathematica, the sigmoid function was given a delayed assignment operator so that it is possible to evaluate the dot product of a given input and weight vectors combination whenever we need to do so. Hence, the definition of the sigmoid function in Mathematica was as follows:

```
Sigmoid [x_] := 1/(1+E^(-x))
```
[4.3]

This definition will allow us to evaluate any expression, x, only when we call the function. If we denote by W and I the weight and input vectors for a given PE, then the output of that element will be:

Output = Sigmoid[X] where X = W.I [4.4]

After defining the input-output matrix and the transfer function, the next step will be to initialize the connection weights. Let Wij be the connection weight between PE[i] and PE[j] as shown in Figure 4.1.

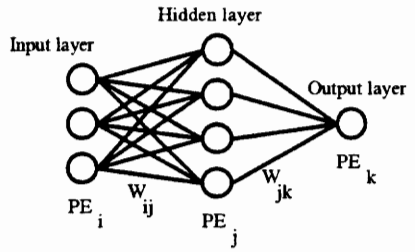


Figure 4.1 - Connection weights in a neural network

For example, a net comprising three input elements, four hidden elements and a single output (Figure 4.1) will have 12 (3 x 4) hidden weights and 4 (4 x 1) output weights. Therefore, a three-layered neural network with m inputs, n hidden elements, and p outputs will have an (m x n) hidden weight matrix and an (n x p) output weights matrix. Using these definitions, the weight initialization process in Mathematica will be as follows:

HiddenWeights = Table [Table [Random [Real, {-0.1, 0.1}, {m}], {n}]]; [4.5]

The Table function builds the list for each column of the weight matrix. Using a nested table function, we build the (m x n) hidden weight matrix by adding up *n* columns of height *m* each. Similarly, the output weights matrix can be initialized as follows:

OutputWeights = Table [Table [Random [Real, {-0.1, 0.1}, {n}], {p}]]; [4.6]

We note here that for back propagation, we typically use small, random real numbers, to initialize the interconnections weights.

Next, we will formulate the generalized delta rule in Mathematica terms and incorporate it in the back propagation process.

The forward propagation assumes that both the hidden and output layers use sigmoids as their output function. The forward propagation for a particular input I could be represented as follows:

Hidden layer outputs (Hidouts) = Sigmoid [I . HiddenWeights] [4.7]

Output layer outputs (Outs) = Sigmoid [Hidouts . OutputWeights] [4.8]

We are now ready to begin the back propagation of the error to update the weights of the hidden and output layers. The back propagation equations presented in chapter 3 will be used to implement the weight update process in Mathematica. The error on each output layer unit is the difference between the desired and actual output value for the unit. For an input-output pair, p, an output unit, k, the error, δ, will be: $\delta_{pk} = y_{pk} - o_{pk}$, where y is the desired output and o is the actual output.

As stated earlier in Chapter 3, if we define the quantity: $\delta^0_{pk} = \delta_{pk} o_{pk} (o_{pk} - 1)$, then the weight update on the connection W_{jk} between the j^{th} unit on the hidden layer, and the k^{th} output unit is given by:

$$W^o_{jk}(t+1) = W^o_{jk}(t) + \eta \delta^o_{pk} i_{pj} \qquad [4.9]$$

Where η is the learning-rate parameter, i_{pj} is the input from the j^{th} unit on the hidden layer, and t is the time step corresponding to the p^{th} input. Similarly, the weight update on the connection Wji between the i^{th} unit on the input layer, and the j^{th} unit on the hidden layer is given by:

$$W_{ji}^{h}(t+1) = W_{ji}^{h}(t) + \eta \delta_{pj}^{h} x_{pi} \qquad [4.10]$$

Where x_{pi} is the input from the i^{th} input layer unit to the j^{th} hidden unit.

With the above equations as a reference, the updates on W_h and W_o, the hidden and output weight matrices, are evaluated as follows:

$$W_o^{t+1} = W_o^t + \eta\left(\Delta_p^0 \bullet I_p^h\right)_p^0 \qquad \text{for output weights.} \qquad [4.11]$$

$$W_h^{t+1} = W_h^t + \eta\left(\Delta_p^h \bullet X_p\right) \qquad \text{for hidden weights.} \qquad [4.12]$$

Where Δ, I, X designate the matrices of the deltas, the hidden units outputs and the network inputs respectively.

We are now ready to build the learning and test functions using the notations already defined in this section.

4.2.3 Network's Learning and Test Functions

The main task of the learning function is to apply the back propagation process to the network using learning examples. The output of this function will be the connection weights of the network as well as the error list for a given number of cycles. We start the

learning function by initializing the connection weights of both, the hidden and output layer. If we denote by inN, hidN, and outN the number of processing units in the input, hidden, and output layer respectively, then the weight initialization process in the learning function is defined as follows:

1. HiddenWeights = Table [Table [Random [Real, {-0.1, 0.1}, {inN}], {hidN}]];
2. OutputWeights = Table [Table [Random [Real, {-0.1, 0.1}, {hidN}], {outN}]];

It is always desired to add a momentum function to the weight update process. The idea behind momentum is that once you start adjusting the weights in a certain direction, keep them moving generally in that direction. In more practical terms, after you adjust the weights during one training iteration, save the value of that adjustment (δ^{o}_{pk}, δ^{h}_{pj}); when calculating the adjustment for the next iteration, add a fraction, α, of the previous adjustment to the new one. In other words, this is equivalent to: (New weight adjustment) + α * (Last weight adjustment). In order to start the momentum function, we should initialize the adjustments of the hidden and output layer units as follows:

3. HiddenLastDelta = Table [Table [0, {inN}], {hidN}];
4. OutLastDelta = Table [Table [0, {hidN}], {outN}];

Now we can start the forward and back propagation paths. The error list will be built-up at the same time. For each cycle, an input-ouput pair, p, is imported from the IO-Matrix as follows:

5. ioP = IO-Matrix[[Random[Integer,{1,Length[IO-Matrix]}]]];

The input and output vectors of the ioP are assigned to the input and desired output of the network as follows:

6. Inputs = ioP[[1]];
7. OutDesired = ioP[[2]];

The forward propagation is then carried out as follows:

8. HiddenOuts=Sigmoid[HiddenWeights . Inputs];
9. Outputs=Sigmoid[OutputWeights . HiddenOuts];

At this point we evaluate the error and start the error back propagation process and the weights adjustments as follows:

10. Outerrors = OutDesired-Outputs;
11. OutDelta = Outerrors (Outputs (1-Outputs));
12. HidDelta = (HiddenOuts(1-HiddenOuts)) Transpose[OutWeights] . OutDelta;
13. OutLastDelta = Eta (OutDelta . HiddenOuts) + Alpha OutLastDelta;
14. OutWeights+ = OutLastDelta;
15. HiddenLastDelta = Eta (hidDelta . Inputs) + Alpha HiddenLastDelta;
16. HiddenWeights = HiddenLastDelta;

Next, we compute the square error (Outerrors * Outerrors) and post it in the error list, and then we move to the next cycle and repeat steps 8 through 16. Finally, the

learning function will return the hidden and output weights matrices and the error list as follows:

17. Return[{HiddenWeights, OutWeights, ErrorList];

We note here that these outputs are then posted in an empty matrix of three elements. The first two elements will be the hidden and output weights matrices, the third element will be the error list matrix.

The test function will be constructed in a similar manner. This function will only have a forward propagation path as shown in steps 7 and 8, and its inputs are the hidden and output weight matrices. The outputs of the test function will be the output values of step 8 and the deviation of these values from the desired ones.

The one major disadvantage to using Mathematica for simulating neural networks is lack of speed. Although you can compile functions in Mathematica to increase their execution speed, Mathematica is primarily an interpreted language, like the old Basic language, and, therefore, is not very fast. Neural networks algorithms such as back propagation or simulated annealing will execute quite slowly for all but the smallest of networks. Nevertheless, researchers generally use small networks to experiment with new architectures or familiarize themselves with the characteristics of neural networks which makes Mathematica an attractive alternative for learning and rapid prototyping.

A complete listing of the Mathematica program is shown in Appendix B. The structure of the program is illustrated in Figures 4.2 and 4.3 for both the learning and test functions.

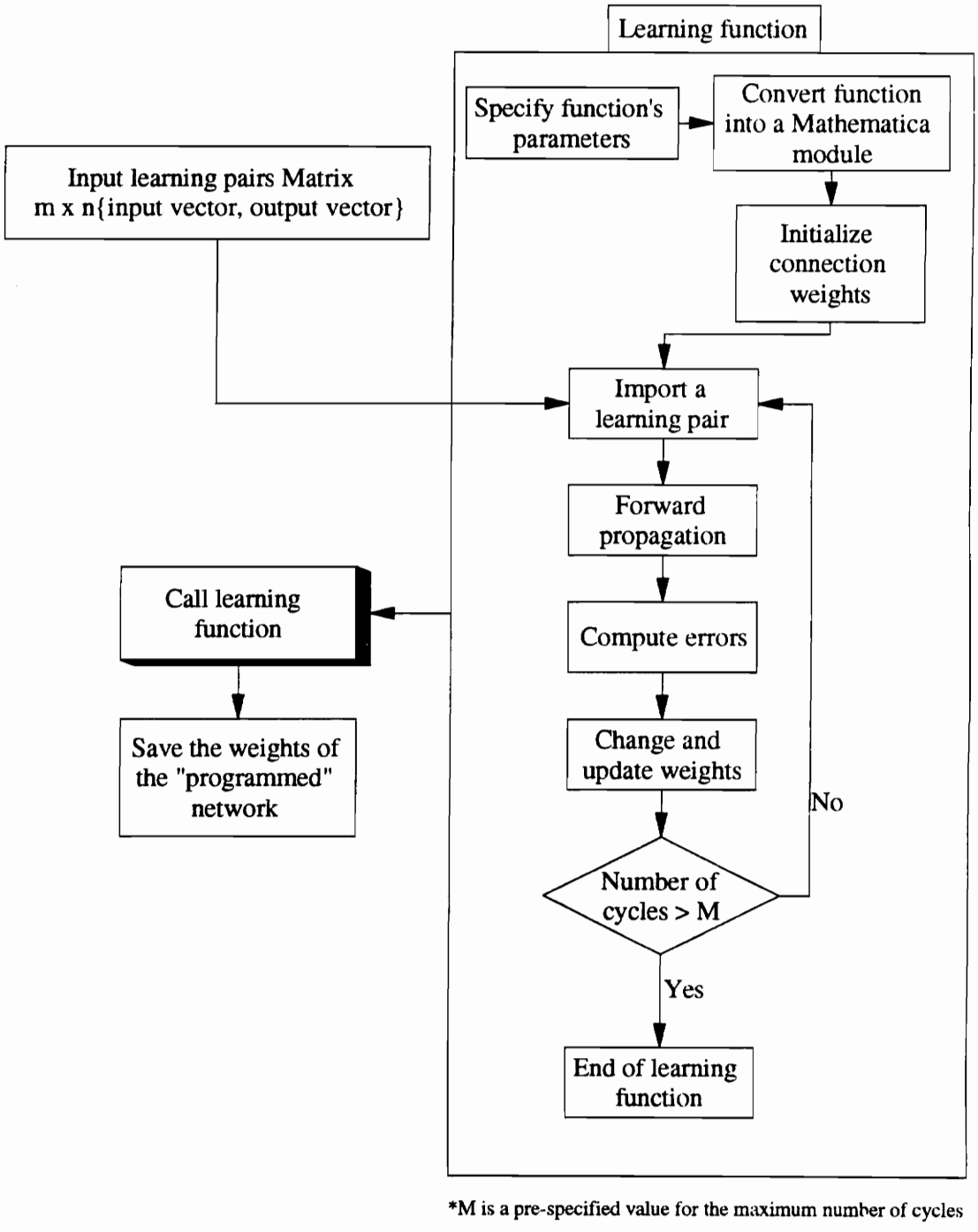


Figure 4.2 Network learning - Mathematica program flow chart

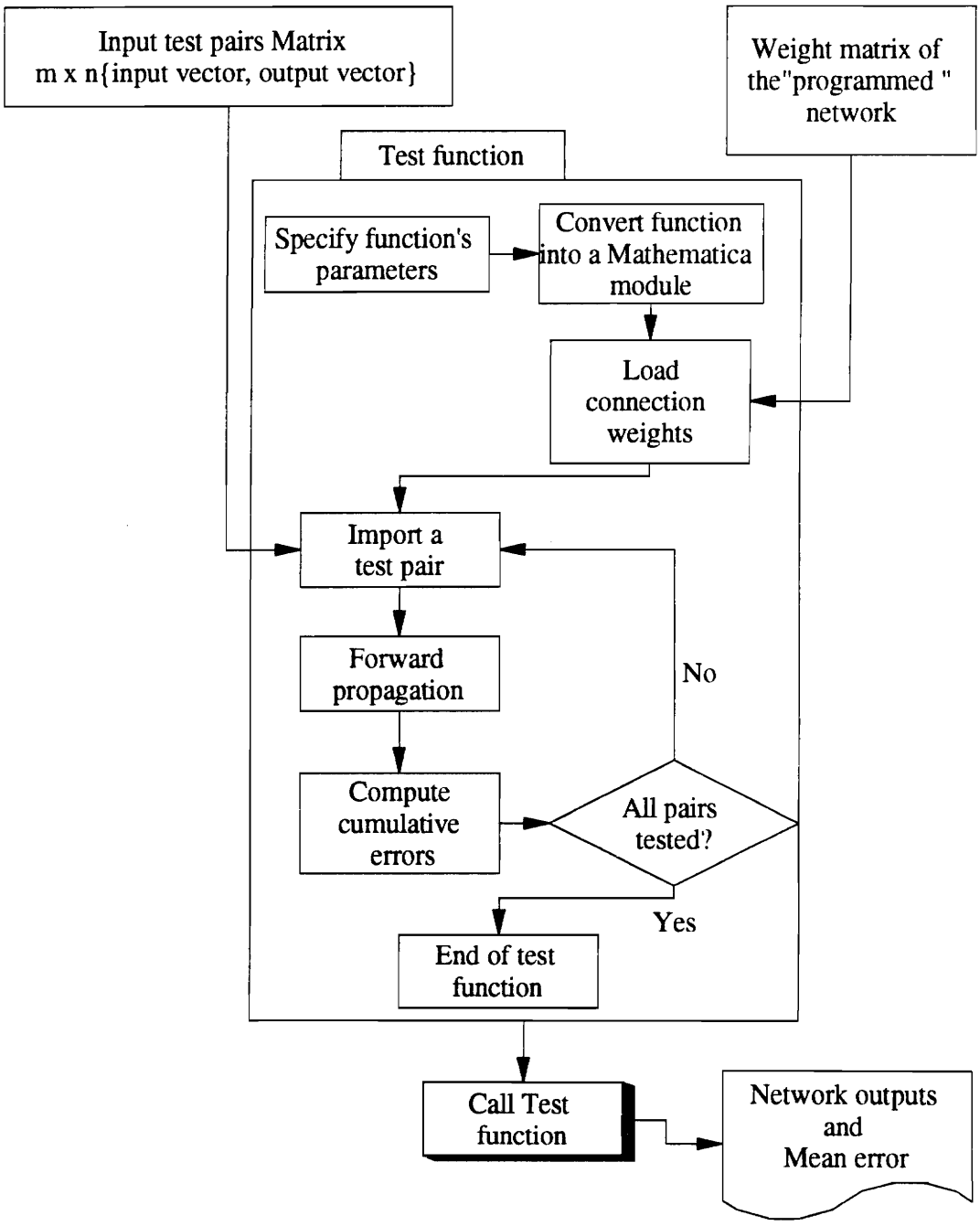


Figure 4.3 Testing the network - Mathematica program flow chart

4.3 PRELIMINARY WORK

Preliminary work consisted mainly of testing the effectiveness of applying the neural network technique to two samples of data:

1. Material cost for 16 carbon steel pipes jobs.
2. Bid estimates for 4 bridges.

The first sample comprised data of actual material cost of a piping section expressed per 100 ft of pipe length. Data for this problem has been adopted from [Sigurdsen, 1992]. Data for the second sample consisted of the detailed cost estimates for four bridges as bidded by an actual construction company located in North Carolina.

A neural network application was developed for each of the data samples. Each sample was split into two sets: a learning set and a test set. The network was trained for a number of cycles using the learning set, and was then asked to make cost estimates for the test set. In addition, the cost estimates of the network were compared to those produced by the multiple regression method which is one of three conventional parameter-based estimating techniques presented earlier in Chapter 2.

4.3.1 Estimating the material cost of carbon steel pipes

Data from 16 previous jobs giving the actual material cost of carbon steel pipes has been adopted from [Sigurdsen, 1992]. The data set comprised the actual cost of the pipe

per 100 ft of pipe length along with the diameter size, number of elbows and flange rating associated with each job as shown in Table 4.1.

The first step was to establish a cost estimating relationship that will enable us to predict the material cost for a pipe section of a particular job. Next, we developed a neural network application, trained the network, and then tested the network predictive capabilities in estimating the material cost of some of the pipes in Table 4.1. Finally, we compared the results of the two methodologies.

Table 4.1 Pipe example data sample [Sigurdsen, 1992]

Job #	X1-Pipe diameter(in)	X2-Number of elbows	X3-Flange rating	C-Actual cost
1	20	14	250	46.1
2	20	14	150	43.2
3	20	14	100	42.1
4	2	14	250	1.9
5	12	12	100	16.8
6	8	16	150	11.7
7	16	12	100	26.3
8	18	8	200	26.1
9	4	4	300	2.5
10	24	12	100	50.2
11	16	12	200	28.4
12	20	12	250	41.3
13	6	12	150	6.5
14	24	8	300	42.3
15	12	4	300	10.8
16	20	8	100	28.9

4.3.1.1 Parametric estimating: establishing the CER

The CER describing the pipe cost (C) as a function of three variables: Pipe diameter (X1), Number of elbows (X2), and Flange rating (X3) will be of the general form

C = F (X1, X2, X3). Multiple regression was used as the parametric estimating tool that will enables us to calculate the parameters or coefficients of the cost estimating relationship. Multiple regression is an expansion of the least-square regression method in that it involves more than one independent variable.

In [Sigurdsen, 1992] it was assumed that the CER has a linear structure of the form:

$$C = A + BX1 + CX2 + DX3 \qquad [4.13]$$

A multiple linear regression analysis yielded the values of the coefficients of [4.13] as follows:

$$C = -27.3 + 2.26\,X1 + 1.308\,X2 + 0.028\,X3 \text{ ; [Sigurdsen, 1992]} \qquad [4.14]$$

Where
- C = Cost per 100 feet of pipe
- X1 = Pipe diameter in inches
- X2 = Number of elbows
- X3 = Flange rating

This equation has a fairly good correlation (R-Square) coefficient of 95 % and a Mean Squared Error (MSE) of 9.99. The mean squared error was computed as follows:

$$MSE = \frac{\sum_{i=1}^{n} (x_i - x_{actual})^2}{n} \qquad [4.15]$$

Where
- x_i = Pipe cost for job i as given by [4.14]
- x_{actual} = Actual pipe cost for job i
- n = Total number of jobs

In our analysis, we investigated a non-linear form for [4.13] since some of the variables might have a non-linear relationship with the cost like the pipe diameter for example. The proposed non-linear CER is of the form:

$$C = A + BX_1^m + CX_2^n + DX_3^p \qquad [4.16]$$

The non-linear regression process is different from the linear analysis in that it involves a search routine for determining the best-fit exponent for each of the variables. The search technique is performed in four steps:

1. First we assume a value for the exponent of the variable that will cause the most variations in the cost value (In our case that is the pipe diameter X1 and the corresponding exponent is m).

2. A linear regression analysis is then applied to (C, X_1^m, X2, X3) and a value of the associated MSE is computed.

3. We repeat step 2 until we reach a value of m resulting in minimum MSE.

4. We fix the value of m, and repeat steps 1, 2 and 3 for the next most sensitive variable.

5. The final regression equation is obtained when all the best-fit exponents search resulting in minimum MSE for all the variables in the equation is completed.

The non-linear regression search for the exponents m, n, p corresponding to X1, X2 and X3 respectively is shown in Figure 4.4. The resulting CER is a non-linear function of the form:

$$C = 0.57X_1^{1.414} + 0.065X_2^2 + (1.5*10^{-8})*X_3^{1.8} - 11.79 \qquad [4.17]$$

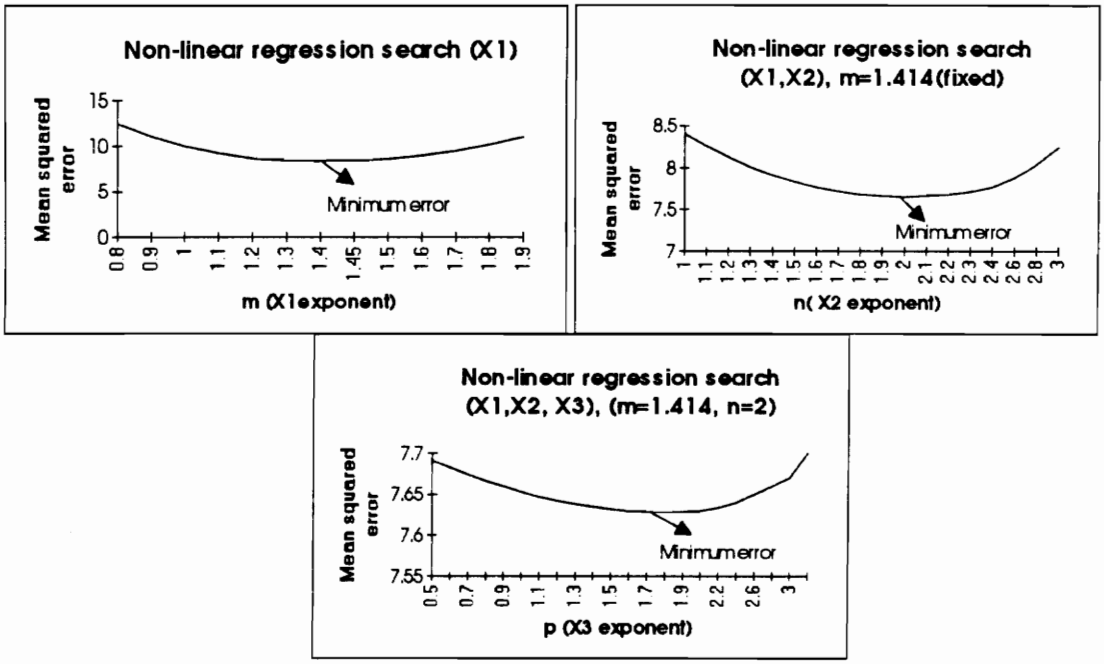


Figure 4.4 Non-linear multiple regression search

The resulting CER has an R-square (correlation coefficient) of 97 % and a MSE of 7.64. We note here that the non-linear cost function resulted in better correlation and is more accurate than [4.14] which has a higher MSE.

4.3.1.2 Neural network application

A three-layered neural network was used to solve the problem. The procedure used to reach a satisfactory solution of the problem was as follows:

1. Splitting the data into (1) a learning set and (2) a test set.
2. Normalizing inputs.
3. Network learning.
4. Testing the trained network

Using the data of Table 4.1, a neural network was trained to predict the material cost of carbon steel pipes using the three variables X1, X2, and X3 as inputs to the network. The data was randomized and split into two sets: a learning set and a test set. The learning set consisted of 10 jobs and was presented to a three-layered back-propagation network having a three units input layer, four units hidden layer and a single output. The network architecture is shown in Figure 4.3. The remaining six data items were isolated from the set and grouped together to constitute the experimental or test set that will be used to evaluate the performance of the trained network.

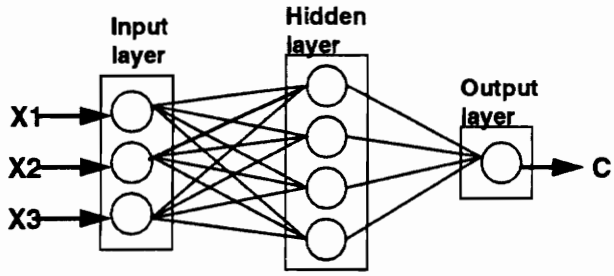


Figure 4.5 Network architecture- Pipe example

The three inputs to network are the pipe diameter, number of elbows and the flange rating. These inputs were normalized to values between 0 and 1 by dividing the actual values by their approximate range. For example, the normalized values for the pipe

of job #1 are obtained by dividing 20, 14, 250, and 46.1 by 25, 18, 350, and 65 respectively. Normalized values are listed in Table 4.2.

The network was programmed in Mathematica. The complete list of Mathematica code used to train and test the network is listed in Appendix C. The one major disadvantage for simulating neural networks using Mathematica is lack of speed with an average processing time of twenty-five minutes for a run of 20,000 cycles.

Table 4.2 Pipe example - Normalized Data

Job #	X1-Pipe diameter(in)	X2-Number of elbows	X3-Flange rating	C-Actual cost
1	0.80	0.7	0.78125	0.71
2	0.80	0.7	0.46875	0.66
3	0.80	0.7	0.31250	0.65
4	0.08	0.7	0.78125	0.03
5	0.48	0.6	0.31250	0.26
6	0.32	0.8	0.46875	0.18
7	0.64	0.6	0.31250	0.40
8	0.72	0.4	0.62500	0.40
9	0.16	0.2	0.93750	0.04
10	0.96	0.6	0.31250	0.77
11	0.64	0.6	0.62500	0.44
12	0.80	0.6	0.78125	0.64
13	0.24	0.6	0.46875	0.10
14	0.96	0.4	0.93750	0.65
15	0.48	0.2	0.93750	0.17
16	0.80	0.4	0.31250	0.44
Range	25	18	350	65

The network was trained for 20,000 and 30,000 learning cycles and was then instructed to predict the cost of both the learning and the test data sets. An interesting observation to note here is that the learning rule (the generalized delta rule) that we used in training the network was flexible enough to allow additional improvements by manipulating the value of the learning coefficient (η) after the 30,000 cycles run. The learning coefficient is nothing but the gradient that determines the amount of shifts in the

value of the interconnection weights at the end of each cycle. The higher the value of η the faster the change in the value of the connection weights and in some instances a high value of η may result in overshooting the solution space where the network exhibits its best performance. In our analysis, we reduced the value of η from 0.15 to 0.01 and performed an additional 1,000 cycles run. The extra run could be considered as a "refinement" process to the weights of the network that will result in a near-optimal solution.

The performance measure that was used to compare neural network results with those of the regression methods, was the value of the mean squared error (MSE). Percentage improvement was computed as follows:

$$PI = \frac{MSE(regression\ method) - MSE(neural\ network)}{MSE(regression\ method)} \text{ x } 100\ \% \qquad [4.18]$$

The results of the network's predictions for the two runs and the additional 1,000 cycles run are shown in Table 4.3. A summary of the results of the various methods is shown in Table 4.4. The table shows that the MSE was 2.47 for the extra run, 3.72 and 3.21 for the 20,000 and 30,000 learning cycles respectively. In all cases, the neural network resulted in less error than that occurring in using the two CERs of the multiple regression method. It can be seen that the neural network computing technology outperformed the parametric estimating techniques and presented strong predictive capabilities. The neural network resulted in substantial reduction in the value of the mean squared error for the example application presented in this paper. The improvements in the accuracy level accomplished by the network as compared to the multiple regression methods ranged from 62.7 to 72.6 percent.

Table 4.3 - Pipe example - Cost estimates summary

Job No.	Actual Cost	Multiple regression		Neural network		
		Linear [4.14]	Non-linear [4.17]	20,000 Cycles	30,000 Cycles	30000(0.15) + 1000(0.01)
Learning set						
2	43.20	39.63	40.91	43.39	43.95	42.96
4	1.90	1.23	4.85	3.23	3.22	3.08
6	11.70	15.13	16.08	12.20	12.00	11.44
8	26.10	28.40	27.87	28.23	28.62	27.49
10	50.20	44.92	48.55	47.51	48.20	47.35
11	28.40	29.12	27.76	29.75	30.63	29.49
13	6.50	5.38	5.33	6.52	6.47	6.17
14	42.30	44.24	47.17	43.43	42.07	42.71
15	10.80	11.89	12.21	8.52	8.92	8.40
16	28.90	30.64	31.84	31.02	31.40	30.31
Test set						
1	46.10	41.91	42.78	42.65	44.12	43.05
3	42.10	38.49	40.26	43.40	43.69	42.73
5	16.80	17.80	16.66	18.79	18.50	17.71
7	26.30	26.84	26.27	30.03	30.00	28.96
9	2.50	-6.19	-2.89	1.71	1.79	1.70
12	41.30	39.30	39.46	40.53	41.77	40.65

Table 4.4 Pipe example - Summary of results

Multiple regression	MSE	Improvement (%) over [4.14]
Linear [4.14]	9.99	-
Non-linear [4.17]	7.64	23.52
Neural network		
20,000 Cycles	3.72	62.76
30,000 Cycles	3.21	67.87
30,000 (eta =0.15) + 1000 (eta =0.01) Cycles	2.47	72.58

4.3.2 Bridge cost estimating

Cost data for the bid estimates of four bridges were obtained from an actual construction company. The total bid items and the associated unit prices of these items are listed in Table 4.5. Using this list, the individual cost of each bridge was calculated along with the quantities of each cost item for that bridge as shown in Table 4.5.

The first step was to perform a parametric cost modeling of the bridge costs. This step required the identification of a set of parameters that best describes the relationship between the final cost of the bridge and the values of these parameters. The methodology used to select these parameters was as follows:

1. Identify parameters that best describe the superstructure.
2. Identify parameters that best describe the substructure.
3. Identify parameters describing the relationship between the superstructure and substructure.
4. The location of the bridge: Overpass, river crossing, off-shore, etc.
5. Type and method of construction.

Since the bridge location and the type and method of construction parameters were the same for the four bridges, the parameter selection process was narrowed down to those describing the superstructure, the substructure or both as illustrated in Figure 4.6. Based on the drawings and bid quantities, a list of five parameters was compiled along the guidelines stated earlier.

Item #	Description	Unit price	Bridge 1		Bridge 2		Bridge 3		Bridge 4	
			Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost
224	Excavation	75	185	13875	262	19650	80	6000	105	7875
225	Reinf. Conc. deck slab	12	6166	73992	12360	148320	7026	84312	7967	95604
226	bridge floors	0.4	5578	2231.2	12121	4848.4	6664	2665.6	7647	3058.8
227	Class A conc.	350	148.2	51870	239	83650	144.8	50680	160.5	56175
228	Slab 19	11000	1	11000	0	0	0	0	0	0
229	Slab 59	18000	0	0	1	18000	0	0	0	0
230	Slab 134	14000	0	0	0	0	1	14000	0	0
231	Slab 134	15000	0	0	0	0	0	0	1	15000
232	Reinf. steel	0.4	24743	9897.2	39863	15945.2	21999	8799.6	24253	9701.2
233	Spiral col RS	1	2987	2987	5072	5072	2314	2314	2508	2508
234	36 " C girder	80	0	0	532.73	42618.4	0	0	0	0
235	45" C girder	90	738.54	66468.6	1023.17	92085.3	0	0	0	0
236	54" C girder	100	0	0	0	0	0	0	0	0
237	Str. steel	1.23	0	0	0	0	172700	212639.19	200900	247360.81
238	Steel piles	14	2210	30940	3950	55300	1130	15820	1480	20720
239	Barrier rail	32	372.8	11929.6	448.77	14360.64	342.04	10945.28	353.42	11309.44
240	4" slope prot.	35	680	23800	620	21700	627	21945	656	22960
241	Bearings	1200	3	3600	3	3600	2	2400	2	2400
242	Exp. Seals	2500	3	7500	3	7500	2	5000	2	5000
243	Perform. joint	1600	3	4800	3	4800	2	3200	2	3200
	Total Cost			**$314,891**		**$537,450**		**$440,721**		**$502,872**

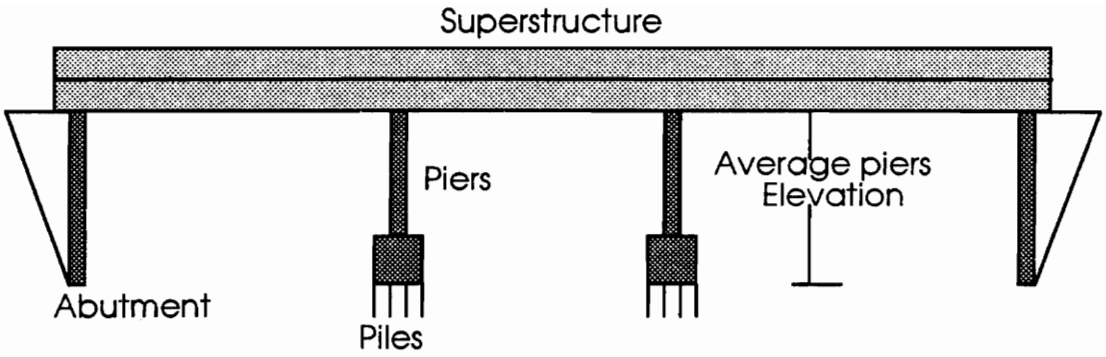


Figure 4.6 Schematic of a typical bridge section layout

The list comprised the following:

1. Average piers' elevation in feet.
2. Number of piers per section.
3. Concrete slab surface in square feet.
4. Type of superstructure : concrete girders (CG) or structural steel (SS).
5. Number of steel piles.

The first two parameters give us the height and number of piers, and therefore an indication of part of the cost of the substructure. In addition, the number of piers per cross section implies an average value for the span, and therefore establishes a relationship between the substructure and the superstructure since, in general, the cost of the superstructure increases when the span between the piers increases and vice versa.

The concrete slabs deck surface and the type of superstructure obviously describe the superstructure and the cost associated with it. The fifth parameter describes the other part of the substructure cost which is the foundations. The five parameters and their respective values are listed in Table 4.6.

Table 4.6 Cost parameters - Bridge example

Parametric modeling of bridge costs					
Input Parameters	**Bridge #1**	**Bridge #2**	**Bridge #3**	**Bridge #4**	**Unit**
1. Average piers height	19	17	17	17	Feet
2. Number of Piers per section	3	3	2	2	#
3. Concrete slabs surface	6166	12360	7026	7967	Sq.Ft.
4. Type of superstructure	1	1	2	2	CG,SS
5. No. of H-Steel piles	40	80	48	42	#
Output parameter					
1. Bridge direct cost	314891	537450	440721	502872	$
Note: CG (concrete girders), SS(structural steel)					

After the completion of the parametric cost modeling process, the next step was to split the four bridges into two sets: (1) a historical set composed of three bridges and (2) an experimental set consisting of the fourth bridge. First, multiple linear regression was applied to the historical set and the resulting correlation equation was used to predict the cost of the fourth bridge. Next, we developed a neural network application, used the historical set for the network's learning process, and then asked the network to predict the cost of the fourth bridge. Finally, we compared the results of the two methodologies.

4.3.2.1 Multiple regression analysis

The cost estimating relationship describing the cost of the bridge as a function of the five parameters listed earlier was assumed to be a linear function of the form:

$$C = A*X1 + B*X2 + C*X3 + D*X4 + E*X5 + F \qquad [4.19]$$

Where C is the total cost of the bridge, and X1, X2, X3, X4, X5 denote the values of the average piers elevation, number of piers per section, concrete slabs deck surface, type of the superstructure, and the number of steel piles respectively. The historical set comprised bridges 1, 2 and 4 as listed in Table 4.6. A multiple linear regression analysis yielded the following equation:

$$C = -4732*X1 + 6534*X2 + 41.83*X3 + 107427*X4 - 1508.5*X5 - 88455 \qquad [4.20]$$

This equation has an R-square coefficient of 1.0 which means a perfect correlation. Using this equation and the parameter values for bridge 3 the predicted cost was $475,583 as compared to a detailed estimate value of $440,721. The percent deviation of the predicted cost using [4.20] and the bid value was computed as follows:

$$Percent\ deviation(\%) = \frac{(475583 - 440721)*100}{440721} = 7.91\%$$

This analysis reveals that there is a strong correlation between the parameters and the total cost of each bridge due to the fact that [4.20] has a perfect correlation

coefficient of 1 for the historical set and a 7.9 % deviation from the bid estimate for the experimental set.

4.3.2.2 Neural network application

A three-layered neural network having a five-units input layer, six-units hidden layer, and a single output was used to solve the problem. The procedure used to reach a satisfactory solution of the problem was as follows:

1. Normalizing inputs for both the historical set and the test set.
3. Network learning.
4. Testing of the trained network

The network architecture is illustrated in Figure 4.7. The values of the five parameters as well as the associated bridge costs were normalized by dividing each value with the corresponding range as shown in Table 4.7.

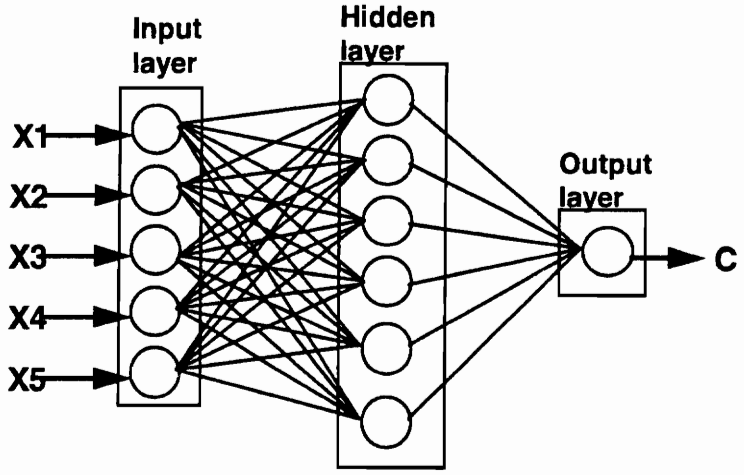


Figure 4.7 Network architecture-Bridge example

Table 4.7 Bridge example - Normalized data for learning and test sets

Bridge #	X1	X2	X3	X4	X5	C
	Learning set					
Bridge 1	0.76	0.75	0.4110667	0.4	0.4	0.314891
Bridge 2	0.68	0.75	0.824	0.4	0.8	0.53745
Bridge 4	0.68	0.5	0.5311333	0.8	0.48	0.502873
	Test set					
Bridge 3	0.68	0.5	0.4684	0.8	0.42	0.440721
Range	25	4	15000	2.5	100	1000000

The network was trained for 1,000 learning cycles and was then instructed to predict the cost of both the learning and the test data sets. The input vector for each bridge consisted of the five parameters {X1, X2, X3, X4, X5} and the output vector consisted of the bid estimate value {C} of the bridge. The trained network was then asked to predict the cost of bridge 3. The network output was $459,583 as compared to a detailed estimate value of $440,721 The percent deviation of the predicted cost using the trained network and the bid value was computed as follows:

$$Percent\ deviation(\%) = \frac{(459583 - 440721) * 100}{440721} = 4.21\%$$

The results show that the neural network methodology resulted in less deviation from the detailed estimated value and therefore was more consistent in predicting the cost of bridge 3 based on the historical data of bridges 1, 2, and 4. The improvement in accuracy achieved by the network as compared to the linear regression method was 46.8%. A final note to mention here is that the neural network developed for this application was tested using a new set of bridges that had different types of construction, different location and larger spans than those of the bridges on which the network was

trained. The test results revealed that the network was inaccurate in estimating the cost of these bridges. Nevertheless, this poor performance by the neural net was expected since the test was simply outside the input range of the trained network and had features that the network has not been exposed to during training.

4.3.3 Preliminary work summary

The results clearly showed the strong predictive capabilities of neural networks and the high accuracy level that could be achieved using this technology. Using the values of Table 4.3, a quick calculation reveals that using a three-layered back propagation neural network, the percent error between actual and estimated costs ranged between -8 and +9 % for the pipe data for the extra run. Similarly, the values listed in section C.1.4 of appendix C show that the percent error between actual and estimated costs ranged between -1 and +4.2 % for the bridge data.

In summary, the preliminary work perfomed using the pipe and bridge examples was intended to show the applicability of the neural network computing technology to cost estimating problems and the great potential that neural networks have in analyzing cost data and making predictions. Moreover, it was a prototype application on how parameter-based cost estimating applications can benefit a great deal from the use of this new technology.

4.4 CASE STUDIES PRESENTATION

This thesis examines the application of neural network computing technology to four case studies where preliminary, and sometimes detailed cost estimates, are carried out using traditional parametric estimating methods, like scaling and regression analysis. The four case studies to be used as applications for both, the neural network methodology and the parameter-based estimating techniques, are extracted from actual construction projects and the cost reports of these projects as documented by either the contractor or the owner.

The four studies are compiled from the records of two construction companies: (1) a major industrial company and (2) an international engineering and contracting comapny, and focus mainly on two areas: (1) Industrial projects and (2) Bridge construction. The case studies are the following:

I. Estimating the cost of major industrial equipment of process plants based on preliminary equipment specifications.

II. Estimating the capital cost of process plants from the equipment list.

III. Estimating the material and installation costs of pressure vessels.

IV. Estimating the cost of bridges in roadway projects.

For each study, cost estimates were performed by using one or more parametric cost estimating technique, as shown in Table 4.8.

Table 4.8 - Parametric estimating techniques used in case studies

	Case study #			
Technique	1	2	3	4
Equipment factor method.		x		
Power law and sizing method.	x		x	
Multiple regression analysis method.	x	x	x	x
Neural Networks	x	x	x	x

4.4.1 Estimating the purchase cost of major industrial equipment of process plants based on preliminary equipment specifications.

The first case study deals with estimating the purchase cost of major industrial equipment based on preliminary information on the specifications of the equipment. The cost of purchased equipment is the basis of several predesign methods for estimating capital investments. Sources of equipment prices, methods of adjusting equipment prices for capacity, and methods of estimating auxiliary process equipment are therefore essential to the estimator in making reliable cost estimates [Peters et al., 1980].

Data for the study was compiled from the close-outs reports of the industrial company for five of its process plants projects.

The compiled information included equipment catalogs, equipment lists, the size and weight of volumetric equipment, types and capacities of motor-driven equipment, and the total purchase cost of the major equipment as recorded for each plant as well as the date of purchase.

After collecting and compiling the data, we divided it into five sets. Each set is comprised of a number of parameters describing the most important characteristics of the major industrial equipment that are believed to have the most impact on the total purchase cost of that equipment and the auxiliary equipment associated with it.

The five data sets are then split into two groups: (1) a historical group and (2) a test group. Regression analysis, equipment scaling, and neural networks were used as the parametric estimating techniques in establishing the cost estimating relationship based on the historical group data.

4.4.2 Estimating the capital cost of process plants from the equipment list.

Various methods could be employed for estimating capital investment. The choice of any method depends upon the amount of detailed information available and the accuracy level desired. A detailed-item estimate has a maximum accuracy of ±5 % of the actual capital investment. This method requires that the equipment and material *individual* costs be determined from *completed* drawings and specifications as well as accurate labor rates, efficiencies, labor-hour calculations, and accurate and detailed indirect costs estimates.

In situations where a detailed method is not feasible because of lack of information or insufficient time to prepare the estimate, alternate, less accurate, methods could be employed, like pre-flowsheet estimates [Cran, 1981] or equipment list estimates [Viola, 1981]. Capital cost estimates based on equipment lists are generally more accurate than pre-flowsheet estimates [Sinha, 1988].

The equipment factor method, regression analysis, and a neural network nested with the neural net of the first study were used as the parametric methods to produce capital estimates. Data for this study is also compiled from the records of the industrial company for the same five projects mentioned earlier. Again, we will have five data sets split into two groups, a historical group and a test group, as in the equipment purchase cost study. We note here that the estimates of the equipment costs obtained in the first study will be treated as one of the input parameters of this second case study. The reason behind this is to show that the accuracy of the capital cost estimate is really a function of the accuracy of the cost correlations available for the equipment and of the factors used for translating the equipment cost to the total capital cost.

4.4.3 Estimating the material and installation cost of pressure vessels.

Purchased equipment costs for vessels, tanks, and process and materials handling equipment can be estimated on the basis of weight [Peters et al., 1980]. However, the cost data generated by this method are only reliable for an order-of-magnitude estimate with an accuracy range of ±35 %. In addition, the installation cost of equipment is a function of labor, supports, platforms, construction expenses, and other factors directly related to the erection of purchased equipment.

This case study deals with finding a set of parameters that could be closely correlated to the purchase and installation costs of pressure vessels in addition to the erection, foundations, insulation and fireproofing costs. Multiple linear regression analysis and the power law and sizing model were used to establish a cost estimating relationship between the total cost of the vessel and a number of independent parameters. On the other hand, a neural network was trained using historical vessel construction data with the same set of parameters and was then asked to estimate the total cost for a number of test vessels.

Data for this case study was compiled from three of the five process plants projects mentioned earlier, and included data for twenty-seven pressure vessels. The collected information for each vessel included the vessel's specifications, costs of labor, insulation, fireproofing, ladders and platforms, and installation. We note here that the three projects are located in approximately the same geographic location and were executed during the same time span. These factors suggest that no cost or labor indices were required to adjust the total cost of the vessels for location or time considerations. Moreover, the three plants under consideration share the same type of processing and functionality.

4.4.4 Estimating the cost of bridges in roadway projects.

A bridge could be described as a system made up of two basic structures : (1) a superstructure and (2) a substructure. In general, the elements of both structures are related together and changes in the elements of one structure affect the elements of the other structure as well as the total cost of the system which is in this case the bridge.

The objective of this fourth case study is to estimate the total cost of a given bridge by modeling it as a function of a small number of superstructure and substructure elements using two methodologies: (1) Correlation analysis and (2) Neural networks.

Data for this case study were obtained from a highway construction project in Northern Virginia. Cost data and section layouts of seventeen bridges were analyzed and arranged into a total set of seventeen data items, i.e., a data item for each bridge. The data set was split into two groups of data items, one historical and another experimental. Correlation analysis was used to establish a cost estimating relationship that was then used to estimate the costs of the bridges of the experimental group. On the other hand, a neural network was trained using the historical group and was then asked to make similar estimates for the experimental group. It should be noted here that the bid estimates prepared using a detailed estimate method which, in general, results in an accuracy level of ±5 % of the actual cost, were considered as the actual costs in this study. Finally, the cost estimates produced by the correlation method and the neural network approach were compared to these bid estimates.

4.5 PARAMETRIC COST MODELING AND ANALYSIS OF CASE STUDIES

In general, the cost of a given system could be modeled as a function of a number of independent variables [Blanchard, 1990]. In parameter-based estimating, the main task is to identify a set of variables that have a logical or theoretical relationship to cost, a statistical significance of the variables' contribution, and independence of the variables in the explanation of the cost [Ostwald, 1980].

Parametric cost modeling could be defined mathematically as a functional model, F, describing a cost, C, as follows:

$$C=F(X1,X2,..,Xn) \qquad [4.21]$$

where X1, X2, .. , Xn are the independent cost variables.

Using this cost model as a reference, parametric estimating techniques often lead to a mathematically fitted function or a cost estimating relationship (CER) where each variable is associated with a given factor or parameter. Regression analysis, power law and sizing model, and the equipment factor method are typical CERs. Although, CERs and neural networks use a parameter-based approach in modeling cost, the computational techniques used by the two methodologies to analyze cost data and produce results are significantly different. For example, in parametric estimating techniques, we assume that the cost can always be defined *mathematically,* whereas the neural computing methodology takes a different approach in describing the cost model. The application of neural networks in cost estimating requires that an input vector comprising the variables {X1, X2, .. , Xn}, be mapped into an output vector describing the cost C.

Back propagation neural networks are called mapping networks if they are able to compute some functional relationship between their input and their output. These networks are very useful where we don't know how to describe the functional relationship in advance, but we do know examples of the correct mapping, like cost estimating for example. Therefore, a neural network computes the cost of a given system by performing the mapping: {X1,X2, ..,Xn}→C. The mapping process is performed in two steps:

1. During network learning, and for each historical mapping example, p, a weight vector, W_p, is set up in the network to match a similar future example.
2. When the input vector, In, of a new example is supplied to the network, this input is mapped into a cost C_n, by associating it with the closest weight vector.

Figure 4.8 illustrates a typical mapping function of a neural network.

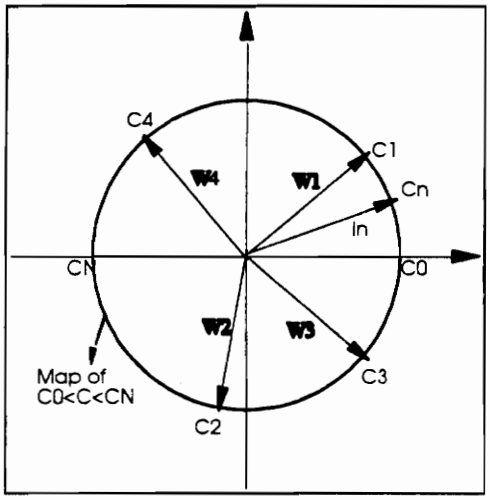


Figure 4.8 Neural network cost mapping

4.5.1 Parametric cost modeling and analysis of major equipment purchase cost

In general, a process plant equipment list is composed of the following major categories:

1- Columns.

2- Vessels and drums.

3- Reactors.

4- Exchangers.

5- Pumps and drivers.

6- Specialty.

In modeling the cost of major process equipment, one should carefully consider the characteristics of the different types of equipment and their functions. Many studies suggest that, for the same metallurgy, the cost of vessels, columns, and reactors could be modeled as a function of the weight or the volume of these pieces of equipment. On the other hand, heat exchangers cost is proportional to the bare tube surface area of the exchanger [Peters and Timmerhaus, 1980]. Many variables could be used to predict the cost of pumps, among these are the weight, flow rate, head and the driving HP (horse power) of the pump motor. However, it is sometimes feasible to express the cost of pumps as a function of the driving HP, especially when we have a large number of them as in the case of process plants for example. Auxiliary and specialty equipment cost could be computed as a fraction of the process equipment cost and it is generally proportional to the number of pieces of these equipment.

Based on the preceding considerations, the purchase cost of major equipment was modeled as a function of the following cost variables:

1- The weight of volumetric equipment, W

2- The volume of volumetric equipment, V

3- Number of volumetric equipment, Nv

4- The bare tube surface area of exchangers, S

5- The number of pieces of equipment, N

6- Cumulative driving power, HP

We note here that the volumetric equipment category includes any vessel of any size, towers, columns, reactors, and drums.

Using these notation, the purchase cost of major equipment, PC, could be modeled, as a function of the six cost variables stated earlier, as follows:

$$PC = F(W, V, Nv, S, N, HP) \qquad [4.22]$$

4.5.2 Parametric cost modeling of the capital costs of process plants

The main task in estimating the capital cost of a process plant from an equipment list, after obtaining the equipment costs, is to develop a reasonable estimate of the direct cost. The direct cost, is usually broken down into the following cost elements [Peters et al., 1980]:

1- Process equipment

2- Non-process equipment

3- Piping

4- Insulation

5- Painting

6- Site development

7- Substructures and superstructure

8- Process buildings

9- Instrumentation and electrical

In addition to these, there are pollution control, water supply treatment, fire protection, demolition and alterations.

[Sinha,1988] suggests that the capital cost of a process plant could be correlated with a number of independent variables derived from the equipment list of that plant. For example, piping for a vessel generally runs the width and height of that vessel; its cost should, therefore, be correlated with the volume of the equipment. The total process piping cost would then depend on the total volume of equipment and the number of pieces of equipment. Similarly, instrumentation would be proportional to the number of pieces of equipment, since the cost of control devices, sensors and their installation is not very sensitive to the size and cost of equipment. Insulation and painting would depend on the external surface of the piping and the vessels; therefore, they could be correlated with the total equipment volume. Site development, substructures, superstructures and process buildings depend on the area and the height occupied by the equipment, and these elements could be correlated to the volume and weight of equipment. Pollution control,

water supply treatment, fire protection, and demolitions and alterations could be directly correlated with the total cost of the direct cost elements mentioned earlier.

Engineering, supervision, start-up, indirect costs, and contingencies could be also correlated to the direct cost. Based on these observations, the capital cost of a process plant, C, could be modeled as a function of the following cost variables:

1- Purchase cost of major equipment, PC
2- The weight of volumetric equipment, W
3- The volume of volumetric equipment, V
4- The number of pieces of equipment, N

Hence, the capital cost function is of the general form:

$$C = F(PC, W, V, N) \qquad [4.23]$$

We note here that PC is obtained from the first case study and is used here as a cost variable.

4.5.3 Parametric cost modeling and analysis of the installation and material cost of vessels

The Purchase cost of vessels is usually estimated based on a cost per unit weight for vessels of the same metallurgy, like carbon steel for example [Phadke, 1977]. Other methods suggest that the purchase cost of a given vessel could be correlated with its

capacity in gallons [Pikulik, 1977]. Nevertheless, the most reliable method for estimating the costs for tanks and pressure vessels is to obtain the assistance of a representative of a vessel fabricator.

On the other hand, installation costs of vessels are more difficult to estimate because of the number of cost variables involved in them. For example, installation labor costs modeled as a function of equipment size showed wide variations when scaled from previous installation estimates [Peters and Timmerhaus, 1980]. Moreover, installation costs of vessels computed as a percentage of the purchased-equipment cost vary greatly [Guthrie, 1974]. In analyzing the cost data for the twenty seven vessels of this study, we hypothetized that the purchase and installation cost of any vessel, including labor, insulation, foundations, fireproofing and platforms could be modeled as a function of three variables:

1- The weight of the vessel, W

2- The volume of the vessel, V

3- A weight assigned to the type of construction , T

Thus, the total cost of a vessel, VC, could be modeled as follows:

$$VC = F(W, V, T) \qquad [4.24]$$

The different types of vessel construction outlined in this study and their corresponding factors are as follows:

Type	Description	Factor, T
1	Vessel not insulated, No foundation and ladders required	2
2	Ladders and platforms required, Little or no insulation	6
3	Foundation required, little or no insulation	8
4	Vessel insulated, Foundation or platforms required	12
5	Vessel Highly insulated, Foundation required	15
6	Vessel insulated, Foundation and platforms required	20
7	Vessel highly insulated, Foundation and platforms required	25

The choice of these factors is based on scaling the costs of six vessels, chosen randomely from the data set and having types of construction of 2, 3, 4, 5, 6, and 7 to the cost of a vessel of type 1 which was assigned a factor of 2.

4.5.4 Parametric cost modeling and analysis of bridges costs

The main task in modeling the cost of a bridge is to find a function that describes the relationship among the elements of the superstructure and substructure and between all these elements and the total cost of the bridge. If we denote by X the set of superstructure elements,{x1,x2,..,xn}, and Y the set of substructure elements, {y1,y2,..,ym}, such that:

$X = \phi(x1,x2,..,xn)$ and $Y = \varphi(y1,y2,..,ym)$. [4.25]

where ϕ and φ are the substructure and superstructure cost functions of the bridge.

Let C be the total cost of a given bridge B. Let's assume that it exists a relationship between C and all the substructure and superstructure elements such that C = F(X,Y). Then C could also be defined as: C = F[ϕ(x1,x2,..,xm),φ(y1,y2,..ym)]. In more general terms C could be modeled as a function of all the elements of the system as follows:

$$C = F[\phi(x1,x2,..,xm),\varphi(y1,y2,..ym)] = G[\, x1,x2,..,xn,y1,y2,..,ym] \qquad [4.26]$$

Where G represents the final bridge cost function.

The parametric cost modeling of bridges cost required the identification of a set of variables that best describes the relationship between the final cost of the bridge and the values of these variables according to Eq. [4.26]. The methodology used to select these variables was as follows:

1. Identify variables that best describe the superstructure.
2. Identify variables that best describe the substructure.
3. Identify variables describing the relationship between the superstructure and substructure.
4. The location of the bridge: Overpass, river crossing, off-shore, etc.
5. Type and method of construction.

Since the location and the method of construction were the same for all the bridges considered in this study, the variables selection process was narrowed down to those describing the bridge type, the superstructure, the substructure or both. Based on the drawings and bid quantities, a list of seven parameters was compiled along the guidelines stated earlier.

The list comprised the following:

1- Deck surface in Sq.Ft., DS

2- Average span in Ft., S

3- Number of piers, Np

4- Average pier width in Ft., Pw

5- Pier type (1= solid pier, 2= columns pier), Pt

6- Pounds of structural steel needed for the construction of the bridge, SS

7- Bridge type (1= Cross-over, 2=Ramp), Bt

Using these variables, the total bridge cost, BC, could be modeled as follows:

$$BC = F(S, DS, Np, Pw, Pt, SS, Bt) \quad [4.27]$$

4.5.5 Summary

The parametric cost modeling process resulted in four cost models to be used by the parametric estimating in establishing CERs for each of the four case studies. The general cost model established for each case study and the corresponding cost associated with it are listed in Table 4.9.

Table 4.9 - Parametric cost models of case studies

Case study #	Cost model	Cost Type
1	PC = F(W, V, Nv, S, N, HP)	Purchase cost of major industrial equipment
2	C = F (PC, W, V, N)	Capital costs of industrial plants
3	VC = F (V, W, T)	Material and installation costs of pressure vessels
4	BC = F(S, DS, Np, Pw, Pt, SS, Bt)	Total cost of bridges

CHAPTER 5

IMPLEMENTATION OF PARAMETER-BASED TECHNIQUES AND NEURAL NETWORKS TO THE CASE STUDIES

5.1 INTRODUCTION

This chapter presents the analysis and experimentation process performed on the four case studies presented earlier in chapter four. The analysis tools used to test the case studies as well as the efficiency of these tools in cost estimating applications are the following:

1. Neural networks applications.

2. Parameter-based techniques:
    - i- Equipment factor method.
    - ii- Power law and sizing model.
    - iii- Regression and correlation analysis.

First, the general guidelines used in developing a neural network application for a given case study, starting with data preparation, normalization, network sizing, convergence, learning and testing, are presented. Next, the analysis of each case study using neural networks and parametric estimating techniques is presented.

1. Case study I : Estimating the cost of major industrial equipment of process plants based on preliminary equipment specifications.

    a- Power law and sizing model

    b- Multiple linear regression analysis

    c- A neural network application

2. Case study II : Estimating the capital cost of process plants from the equipment list

    a- Equipment factor method.

    b- Multiple linear regression analysis.

    c- A neural network application.

3. Case study III: Estimating the material and installation cost of pressure vessels.

    a- Power law and sizing model

    b- Multiple linear regression analysis

    c- A neural network application

4. Case study IV: Estimating the cost of bridges in roadway projects.

    b- Multiple linear regression analysis.

    c- A neural network application.

Finally, a summary of the results obtained by neural networks applications and parametric estimating techniques will be outlined for each one of the four case studies. These results include the cost estimates obtained by each method, the mean percent deviation of these estimates from the actual costs, and the mean absolute error of the test set.

5.2 DATA PREPARATION AND DEVELOPMENT OF NEURAL NETWORK APPLICATIONS IN COST ESTIMATING

The major steps and guidelines used in developing and implementing a neural network to a cost estimating application are the following:

1. Data preparation and normalization.
2. Network sizing and convergence.
3. Network training and testing.

5.2.1 Data preparation and normalization

In preparing the implementation of a neural network, it is important to assess the available data properly before presenting it to the network. The data need to be converted into another form to be meaningful to the network. In general, the data will be either continuous-valued or binary.

All of the cost data collected for the four case studies of this thesis, are continuous numbers. In this case, it is a mistake to break the data into artificial groups, because it is difficult for the network to learn examples on or near the borders of each group. In general, the operating range of the neuron or computing unit of the neural net will be dictated by the transfer function of that unit. Since all the neural networks developed in this thesis use a sigmoidal transfer function, and thus have an operating range between 0 and 1 for the computing units, we had to normalize the data.

For example, if the input data could be anything from 49 to 174, the total range is 125. An input value of 92 is 43/125 of the range or .344 on a scale from 0 to 1. As a rule of thumb, the data range of a given set is computed as 1.2 x (Max-Min), where Max and Min are the maximum and minimum values in the data set respectively [Lawrence, 1992]. In our analysis, the range was assumed to be:

$(1+n) * Max$ [5.1]

Where $0<n<0.5$

This equation presents a slight deviation from the one proposed by Lawrence because of the form of the sigmoidal function. This function will result in the fact that network outputs can never reach 0 or 1. Therefore, it is always desirable to use values such as 0.1 and 0.9 to represent the smallest and largest output values. This could be done by changing the slope of the linear portion of the sigmoid by including a multiplicative constant in the exponential.

5.2.2 Network sizing and convergence

The size of both the input and output layers is usually dictated by the nature of the application. There are no strict rules that dictate the best or optimal number of layers in a neural net. However, the basic rule in determining the number of units to use in the hidden layer is to use as few units as possible. The general approach in sizing the hidden layer of a neural network is to start with a few number of hidden units, and if the network fails to converge to a solution, it may be that more hidden nodes are needed. If it does converge,

we might try fewer hidden units and settle on a size on the basis of the overall system performance, i.e., for both the learning and testing sets.

5.2.3 Network training and testing

Back propagation neural networks are good at generalization. What we mean by generalization is that given several different input vectors, all belonging to the same class, a back propagation neural net will learn to make significant correlations between these input vectors and their corresponding output vectors. The noise introduced to the learning process by irrelevant data will be ignored or minimized.

In contrast to generalization, back propagation neural networks will not extrapolate well. If the network is insufficiently trained on a particular class of input vectors, subsequent testing of members of that class may be unreliable, like in the case of the test performed on the network developed for the bridge example presented earlier in Chapter 4. In order to avoid that problem, one should make sure that the training data covers the entire expected input space. During the testing process, if a particular test case is not handled properly by the trained network, it is most likely that this an indication of some untrained feature or a gap in the coverage of the input space during learning and that case is then added to the learning set.

5.3 CASE STUDY I : ESTIMATING THE COST OF MAJOR INDUSTRIAL EQUIPMENT OF PROCESS PLANTS BASED ON PRELIMINARY EQUIPMENT SPECIFICATIONS.

This study deals with estimating the purchase cost of the major equipment of an industrial plant based on preliminary equipment specification. As stated earlier in chapter 4, the cost of major industrial equipment could be modeled as follows:

$$PC = F(W, V, Nv, S, N, HP) \qquad [5.2]$$

Where PC = The purchase cost of major equipment for a base year Y.

W = The weight of volumetric equipment

V = The volume of volumetric equipment

Nv = Number of volumetric equipment

S = The bare tube surface area of exchangers

N = The number of pieces of equipment

HP = Cumulative drivers power

PC includes the cost of columns, vessels and drums, reactors, exchangers, pumps and drivers, specialty, auxiliary, and non-process equipment.Using these notations, cost estimates were prepared using two cost estimating relationships: (1) multiple linear regression; and (2) the power law and sizing method. In addition, a neural network application was used in making similar cost estimates.

As mentioned in chapter 4, five data sets pertaining to five process plants projects were compiled from the files of a large chemical and industrial company. Since the

projects were completed in different years, it was necessary to scale the cost of equipment to a base year Y. The Marshall & Swift Cost Equipment Cost Index was used as the basis of scaling. Using this index (Table 5.1) the cost of major equipment for the five data sets was scaled to 1988 dollars. This is done by multiplying the cost of equipment by the cost index of the base year and then dividing it by the cost index corresponding to the year in which the project was completed.

Table 5.1 Marshall & Swift Annual Equipment Cost Index

YEAR	ANNUAL INDEX
1988	852.0
1989	895.1
1990	915.1
1991	930.6
1992	943.1
1993	964.2

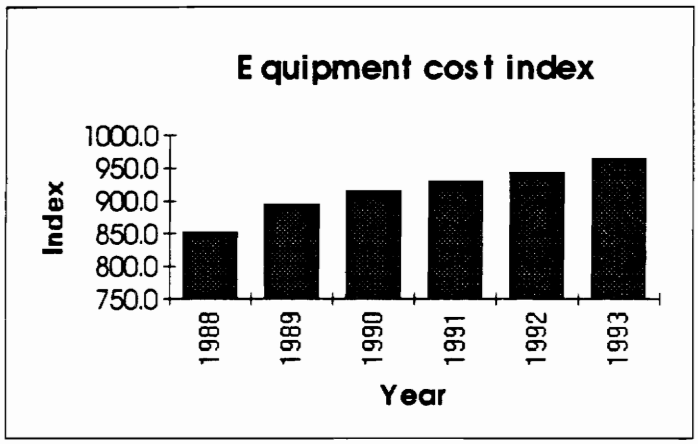


Figure 5.1 Marshall & Swift Annual Equipment Cost Index

The purchase cost of major equipment and the corresponding values for the cost variables associated with it are listed in Table 5.2

PROJECT	P1-DCP	P2-OLE	P3-DCU	P4-HTU	P5-SBU
YEAR	1993	1988	1991	1991	1991
List of cost variables					
Weight of volumetric equipment (x1000	1235	1507	2654	1740	1051
Volume of volumetric equipment (Cu.Ft	95000	39494	209529	27378	86500
Number of volumetric equipment	57	31	41	33	27
Exchangers total surface area (Sq. Ft.)	132634	58800	181748	63375	60400
Total pieces of equipment	183	115	224	133	97
Drivers cumulative horsepower (HP)	6223	3030	4761	8330	2170
Major equipment purchase cost	$12,684,000	$3,743,000	$36,813,000	$19,112,000	$8,283,000
Major equipment purchase cost (1988 dollars)	$11,208,015	$3,743,000	$33,703,714	$17,497,769	$7,583,404

5.3.1 Power law and sizing method: Establishing the CER.

Purchased equipment costs for vessels, tanks, and process and materials handling equipment can often be estimated on the basis of weight [Peters et al., 1980]. The cost of non-process, auxiliary, specialty and motor driven equipment could also be correlated with that of process equipment. Based on these assumptions, the cost function of [5.2] will be reduced to:

$$PC = F(W) \qquad [5.3]$$

Using the power sizing method, [5.3] translates into the following cost estimating relationship:

$$PC2 = PC1 * \left(\frac{W2}{W1}\right)^{m}, \quad 0 < m < 1 \qquad [5.4]$$

Where PC1, and W1 are the purchase cost and weight for a base equipment list. PC2 and W2 are the estimated purchase cost and weight for a new equipment list. Values for m of 0.6 and 1 were used which resulted in the following CERs:

$$PC2 = PC1\left(\frac{W2}{W1}\right)^{0.6} \quad m=0.6 \qquad [5.5]$$

$$PC2 = PC1\left(\frac{W2}{W1}\right) \quad m=1.0 \qquad [5.6]$$

These values were used because they correspond to the most common value of the exponent m and the extreme value of 1 for the m range. The results of these two

exponents will determine the range of the most appropriate exponent for this study. Project 5 was chosen as the test case with projects 1 through 4 as different alternatives for the base equipment purchase cost. Using these values and equations [5.5] and [5.6], estimates of the purchase cost of major equipment for project 5 were prepared as shown in Tables 5.3 and 5.4. The percent deviation of the estimates from actual equipment costs ranged between -60.2% and 154.94% for m=0.6, and -65.6% and +76% for m=1.0.

Table 5.3 - Power law and sizing method - Major equipment purchase cost estimates (m=0.6).

			Test project - P5					
Base project	W	PC	W5	(W5/W)^ 0.6	Estimate PC5	Actual cost	Error	% deviation
P1	1507	$3,743,000	1051	0.81	$3,015,182	$7,583,404	($4,568,222)	60.24%
P2	2654	$33,703,714	1051	0.57	$19,332,982	$7,583,404	$11,749,577	154.94%
P3	1740	$17,497,769	1051	0.74	$12,930,483	$7,583,404	$5,347,079	70.51%
P4	1235	$11,208,015	1051	0.91	$10,173,963	$7,583,404	$2,590,559	34.16%

Table 5.4 - Power law and sizing method - Major equipment purchase cost estimates (m=1.0).

			Test project - P5					
Base project	W	PC	W5	(W5/W)^1 .0	Estimate PC5	Actual cost	Error	% deviation
P1	1507	$3,743,000	1051	0.70	$2,610,413	$7,583,404	($4,972,991)	65.58%
P2	2654	$33,703,714	1051	0.40	$13,346,874	$7,583,404	$5,763,470	76.00%
P3	1740	$17,497,769	1051	0.60	$10,569,055	$7,583,404	$2,985,651	39.37%
P4	1235	$11,208,015	1051	0.85	$9,538,157	$7,583,404	$1,954,753	25.78%

The results clearly show that a value of 1.0 for the exponent results in better estimates, since the range of errors and percent deviations are reduced from those resulting from the use of the six-tenth factor.

5.3.2 Multiple linear regression analysis: Establishing the CER

In this analysis, we assumed that the cost model of the purchase equipment cost could be written as a linear function of the form:

$$PC = a*W + b*V + c*Nv + d*S + e*N + f*HP + g \qquad [5.7]$$

Where a, b, c, d, e, f, and g are the linear regression coefficients of the cost function,

and W, V, Nv, S, N, HP are the cost variables,

and PC is the purchase cost of major equipment.

Multiple linear regression analysis was used to determine the coefficients of equation [5.7] using the cost data of projects 1,2,3 and 4. Microsoft Excel© was used to compute the coefficients of the linear regression equation as follows:

1. For a linear equation of the form : $y = m_1x_1 + m_2x_2 + \cdots + b$ where the dependent y-value is a function of the independent x-values, the m-values are coefficients corresponding to each x-value, and b is a constant value, then the values of the m coefficients can be calculated by using the function "LINEST" in Microsoft Excel.

2. The "LINEST" function returns the array $\{m_n, m_{n-1}, \ldots, m_1, b\}$ for a set of *known_y's* and *Known_x's* by using the syntax LINEST (*known_y's*,*Known_x's*, const, stats), where *const* is logical value specifying whether to force the constant b to equal 0, and *stats* is a logical value specifying whether to return additional statistics. A "True" value for both *const* and *stats* will return values for b and the statistical data respectively.

3. For this study, we grouped the y's (PC) and x's (W, V, Nv, S, N, HP) as shown below and we then applied the LINEST function as follows:

LINEST(G13:G16, A132:F16, True, False);

	Column						
Row #	A	B	C	D	E	F	G
13	1507	39500	31	58800	115	3030	3743000
14	2654	209500	41	181800	224	4761	33703714
15	1740	27400	33	63800	133	8330	17497769
16	1235	95000	57	132600	183	6223	11208015
	W	V	Nv	S	N	HP	PC

This function returned the following output:

Row #	A	B	C	D	E	F	G
17	3435.5239	-38750.54	247.57709	-1024561	98.131468	-7542.364	22483681
Coefficient	m6	m5	m4	m3	m2	m1	b

The multiple linear regression equation can now be obtained by using the values from row 17:

$$PC = -7542.36 * W + 98.13 * V - 1024561 * Nv + 247.58 * S - 38750.54 * N + 3435.52 * HP + 22483681 \quad [5.8]$$

This equation has a perfect correlation coefficient of 1.0 and a mean error and deviation of 0. The input vector of Project 5 corresponding to the values {W=1051, V=86500, Nv=27, S=60400, N=97, HP=2170} was supplied to equation [5.8] to obtain an estimate of the purchase cost of the major equipment for that project. This estimate is then compared to the actual cost of $7,853,404. The estimate produced by equation [5.8] was $14,031,823 resulting in an error of $6,448,419 and a percent deviation of 85%. We note here that even though the regression equation had a correlation coefficient of 1, it performed very poorly when exposed to a new set of data.

5.3.3 Neural network application

A neural network application was developed to map the values of the cost variables (or the input vector) into a value of the purchase cost of the equipment as follows:

$$\{ W, V, Nv, S, N, HP \} \rightarrow \{PC\} \qquad [5.9]$$

A three-layered feed forward back propagation neural network having a six-unit input vector, three-unit hidden layer, and a single unit output layer was used in training and testing the cost data. The network architecture is illustrated in Figure 5.2.

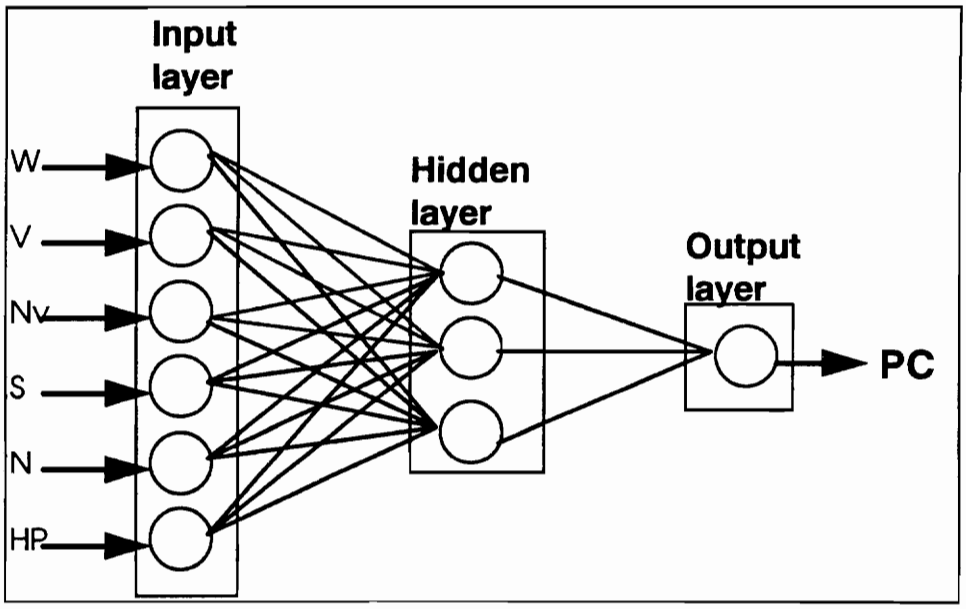


Figure 5.2 - Case Study I - Network Architecture

The training set comprised the cost data of projects 1, 2, 3 and 4, and the test set comprised the cost data of project 5. The input and output vectors of both sets were normalized as shown in Table 5.5. In normalizing the value of the cost, a constant of $5 million was added to the actual equipment cost of each project in order to limit the normalized values of the desired output vector to values between 0.1 and 0.9.

Table 5.5 - Case Study I - Normalized data

	Input						Output
	W	V	Nv	S	N	HP	PC
Test set							
P5	0.350777	0.346	0.385714	0.302	0.388	0.217	0.251668085
Learning set							
P1	0.50233333	0.158	0.442857	0.294	0.46	0.303	0.17486
P2	0.88466667	0.838	0.585714	0.909	0.896	0.4761	0.774074275
P3	0.58	0.1096	0.471429	0.319	0.532	0.833	0.449955384
P4	0.41166667	0.38	0.814286	0.663	0.732	0.6223	0.324160299
Range	3000	250000	70	200000	250	10000	(PC +5000000)/ 50000000

The network was trained for 2500 cycles using a learning rate of 0.15, a momentum coefficient of 0.9, and a sigmoidal transfer function for all the processing units. Then we plotted the Mean Squared Error vs. the total number of cycles as shown in Figure 5.3. The error plot shows that no more training cycles are required since the MSE converged assymptotically over the cycles axis. It should be noted here that sometimes it is better to stop training before we reach a very low MSE, in order not to cause what is

referred to as "over training." This is why we chose to plot the error instead of specifying a threshhold value for it when we performed the network training phase.

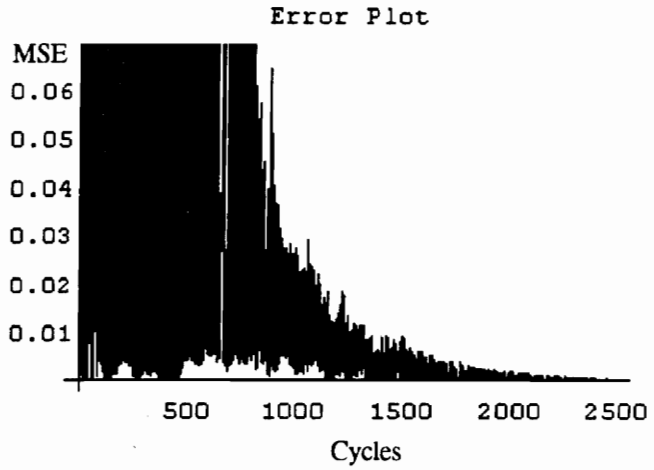


Figure 5.3 - Case study I - Neural Network Training Error Plot

The resulting hidden and output connection weights of the trained network are the following:

Hidden weights ={

{-0.342282, -2.31226, 2.78473, -0.954718, 0.367379, -1.85039},

(*First hidden unit*)

{0.475612, 0.661695, -0.237559, 0.727532,0.0660697, 0.866314},

(*Second hidden unit*)

{-0.267698, -0.556785, -0.123483, -0.654603,-0.353348, -0.592483}

(*Third hidden unit*) };

Output weights ={{-5.29296, 2.28468, -1.33046}};

The mean error and mean percent deviation of the training set were $109,812 and 0.38% respectively. The input vector of Project 5 corresponding to the values {W=1051, V=86500, Nv=27, S=60400, N=97, HP=2170} was supplied to the "trained" network to obtain an estimate of the purchase cost of the major equipment for that project. The estimated cost value of the test set was $8,440,340 as compared to an actual equipment cost of $7,583,404. The resulting error between the estimated value and the actual cost is $856,938 and the percent deviation 11.3 %. A listing of the computer run in Mathematica and the results of network training and testing are shown in Appendix C.

5.3.4 Case study I - Summary of results

A summary of the results of the different methods used in the analysis of this study (the power law and sizing method, the multiple linear regression method, and the neural network method), are listed in Table 5.6. Those results comprise the estimated equipment cost of the test project, the error and the percent deviation of that estimate from the actual cost.

Table 5.6 - Case study I - Summary of results

Method	Estimate	Actual cost	Error	Percent deviation
Power law and sizing method				
m=0.6, P4 base	$10,173,963	$7,583,404	$2,590,559	34.16%
m=1.0, P4 base	$9,538,157	$7,583,404	$1,954,753	25.78%
Multiple linear regression	$14,031,823	$7,583,404	$6,448,419	85.03%
Neural network	$8,440,340	$7,583,404	$856,936	11.30%

5.4 CASE STUDY II :ESTIMATING THE CAPITAL COST OF PROCESS PLANTS FROM THE EQUIPMENT LIST

This case study deals with estimating the capital cost of a process plant, once the equipment list has been prepared. As stated earlier in chapter 4, the capital cost of process plants could be modeled as follows:

$$C = F(W, V, PC) \qquad [5.10]$$

Where C = Capital cost of a process plant for a base year Y

W = The weight of volumetric equipment

V = The volume of volumetric equipment

PC = Estimated purchase cost of major equipment from Case Study I

C includes the cost of equipment, materials and labor, all indirect costs and contingencies such as engineering, supervision, and start-up costs.

Using these notations, cost estimates were prepared using two cost estimating relationships: (1) the equipment factor method; and (2) the multiple linear regression analysis. In addition, a neural network application was used in making similar cost estimates.

As mentioned earlier, five data sets pertaining to five process plants projects were compiled from the files of a large chemical and industrial company. Since the projects were completed in different years, it was necessary to scale the capital costs of these

plants to a base year Y. The Chemical Engineering Plant Cost Index was used as the basis of scaling. Using this index (Table 5.7) the capital costs of the five data sets were scaled to 1988 dollars. This is done by multiplying the capital cost of each plant by the cost index of the base year and then dividing it by the cost index corresponding to the year in which the project was completed.

Table 5.7 - Chemical Engineering Plant Cost Index

YEAR	ANNUAL INDEX
1988	342.5
1989	355.4
1990	357.6
1991	361.3
1992	358.2
1993	359.2

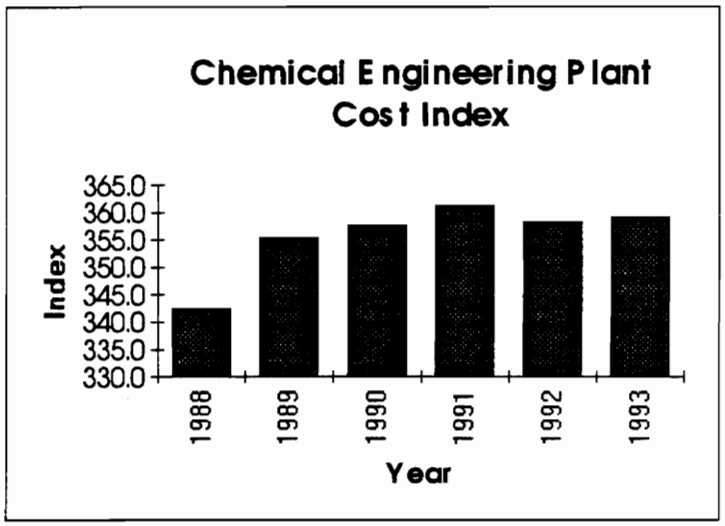


Figure 5.4 - Chemical Engineering Plant Cost Index

The capital cost of each process plant and the corresponding values for the cost variables associated with it are listed in Table 5.8.

PROJECT	P1-DCP	P2-OLE	P3-DCU	P4-HTU	P5-SBU*
Year	1993	1988	1991	1991	1991
	Learning set				Test set
List of parameters					
Weight of volumetric equipment (1000lbs)	1235	1507	2654	1740	1051
Volume of volumetric equipment (Cu.Ft)	95000	39494	209529	27378	86500
Major equipment purchase cost (1988 dollars)	$11,208,015	$3,743,000	$33,703,714	$17,497,769	$6,825,064
Total plant cost	$80,325,000	$27,725,000	$189,044,000	$87,900,000	$76,341,000
Total plant cost (1988 dollars)	$76,590,514	$27,725,000	$179,207,224	$83,326,183	$72,368,648

5.4.1 The Equipment factor method

The equipment factor method suggests that the capital cost of a plant could be obtained by multiplying the purchase costs of different types of equipment by a number of cost factor. As mentioned earlier in chapter 2, the Cran method of factor estimating will be used to estimate the capital costs of each of the five projects using the following cost estimating relationship:

$$C = \left(\sum EF_D + IF_I\right) * (1 + F_O) \qquad [5.11]$$

Where C = Total plant cost

E = The purchase cost of a particular type of equipment.

F_D = The direct cost factor that varies with the different type of equipment and its material of construction.

I = The sum of the costs of all the instruments.

F_I = The direct cost factor of the instruments

F_O = The indirect cost factor

The cost factors suggested by Cran to be used in equation [5.11] for estimating the total cost of a process plant from the purchase cost of the different types of equipment and the instrumentation cost of that project are listed in Table 5.9.

Table 5.9 Cost factors for the equipment factor method [Cran, 1981]

Equipment type	Cost Factor
Pumps, centrifugal, carbon steel	2.8
Instruments, all types	2.5
Columns, carbon steel	3
Vessels and drums, carbon steel	2.8
Tanks, storage, carbon steel	2.3
Heat Exchangers	2
Reactors	2.2
Filters, all types	1.4
Specialty*	1.5
Indirect cost factor	0.315

* Estimated as an average for all the types of specialty equipment.

These factors are then used to estimate the capital costs of two of the five projects of this study using the actual purchase costs of the different types of equipment listed in Table 5.9, in addition to the cost of instruments. Only two projects were analyzed because the individual cost item for instruments was not available for projects 3, 4, and 5. The computations leading to capital cost estimates for the two projects of this study according to the Cran method are shown in Table 5.10.

Table 5.10 - Capital cost estimates: The Equipment Factor Method

PROJECT/YEAR		P1(1993)		P2(1988)	
Equipment type	Cost Factor	Purchase cost	Cost	Purchase cost	Cost
Pumps, centrifugal, carbon steel	2.8	$2,557,938	$7,162,226	$590,000	$1,652,000
Instruments, all types	2.5	$4,913,421	$12,283,553	$1,285,000	$3,212,500
Columns, carbon steel	3	$3,101,386	$9,304,158	$504,000	$1,512,000
Vessels and drums, carbon steel	2.8	$1,156,351	$3,237,783	$326,000	$912,800
Tanks, storage, carbon steel	2.3	$313,834	$721,818	$417,000	$959,100
Heat Exchangers	2	$4,218,085	$8,436,170	$578,000	$1,156,000
Reactors	2.2	$549,922	$1,209,828	$630,000	$1,386,000
Filters, all types	1.4	$84,463	$118,248	$24,000	$33,600
Specialty*	1.5	$522,865	$784,298	$371,000	$556,500
Subtotal (Direct cost)			$43,258,082		$11,380,500
Indirect cost factor	0.315		$13,626,296		$3,584,858
Estimated Plant Cost			$56,884,378		$14,965,358
Estimated Plant Cost (1988 dollars)			$54,239,698		$14,965,358
Actual Plant cost (1988 dollars)			$76,590,514		$27,725,000
Error			($22,350,816)		($12,759,643)
Percent deviation			-29.18		-46.02

The accuracy of this method ranged between -29.18% and -46.02% for projects 1 and 2. The major reason why this method was presented, is to give an example on the performance of preliminary estimating techniques used to produce capital cost estimates following a parameter-based approach.

5.4.2 Multiple Linear Regression : Establishing the CER

In this analysis, we assumed that the cost model of the capital cost of a process plant could be written as a linear function of the form:

$$C = a*W + b*V + c*PC + d \qquad [5.12]$$

Where a, b, c, d are the linear regression coefficients of the cost function, and W, V, and PC are the cost variables.

Multiple linear regression analysis was used to determine the coefficients of equation [5.12] using the cost data of projects 1, 2, 3 and 4.

The resulting linear regression equation is the following:

$$C = -15136.53 * W + 203.9 * V + 4.478 * PC + 25720602.95 \qquad [5.13]$$

This equation has a perfect correlation coefficient of 1.0 and a mean error and deviation of 0. The input vector of project 5 corresponding to the values {W=1051, V=86500, PC=14031823} was supplied to equation [5.13] to obtain an estimate of the capital cost for that project. The estimate produced by equation [5.13] was $90,288,452 resulting in an error of $17,919,805 and a percent deviation of 24.76 %.

5.4.3 Neural network application

A neural network application was developed to map the values of the cost variables (or the input vector) into a value of the capital cost of each plant as follows:

$$\{ W, V, PC \} \rightarrow \{C\} \qquad [5.14]$$

A three-layered feed forward back propagation neural network having a three-unit input vector, four-unit hidden layer, and a single unit output layer was nested to the neural network of case study I. The purpose of the nesting of the two networks is to link the output (PC) of the first network to one of the inputs of the second network and two of the

inputs of the first network (W,V) to the remaining inputs of the second network. The network architecture is illustrated in Figure 5.4.

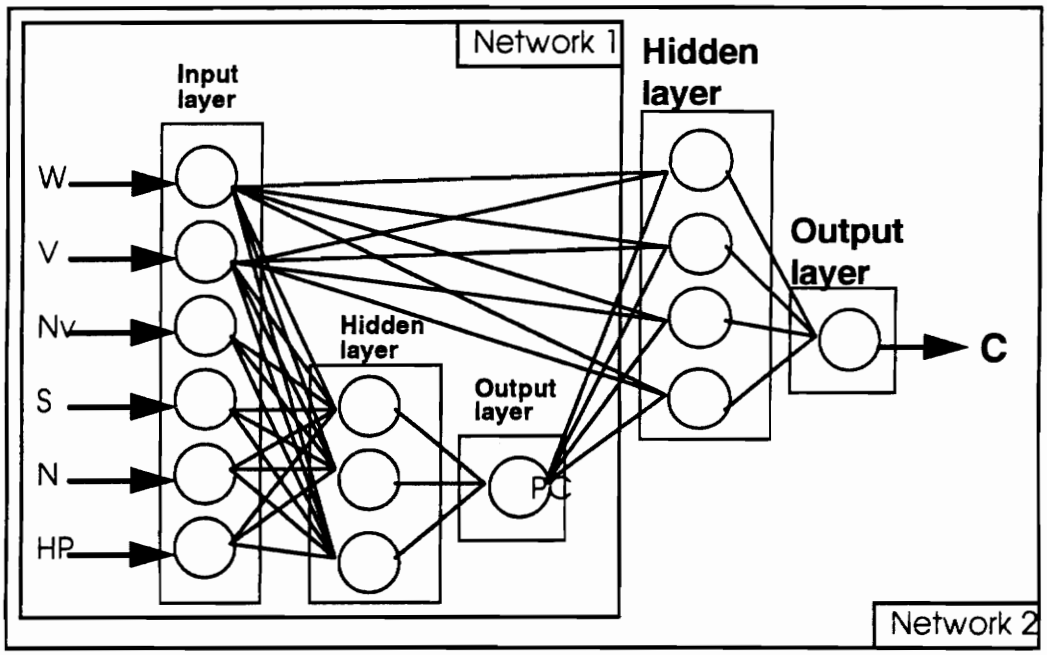


Figure 5.5 Case Study II - Network Architecture

The training set comprised the cost data of projects 1, 2, 3 and 4, and the test set comprised the cost data of project 5. The input and output vectors of both sets were normalized as shown in Table 5.11.

Table 5.11- Case Study II - Normalized data

	Input			Output
	W	V	PC	C
Test set				
P5	0.350777	0.346	Output NN1	0.382952568
Learning set				
P1	0.50233333	0.158	Output NN1	0.138625
P2	0.88466667	0.838	Output NN1	0.89603612
P3	0.58	0.1096	Output NN1	0.416630916
P4	0.41166667	0.38	Output NN1	0.36184324
Range	3000	250000	N/A	200000000

The network was trained for 2000 cycles using a learning rate of 0.3, a momentum coefficient of 0.9, and a sigmoidal transfer function for all the processing units. The hidden and output connection weights of the trained network are the following:

Hidden weights={

{0.0948155, -1.98772, -1.11391}, (* First Hidden Unit*)

{0.4741, 1.06357, 0.47164}, (* Second Hidden Unit*)

{-0.56876, -1.33477, -0.886306}, (* Third Hidden Unit*)

{0.614657, 1.45424, 0.467622} } (* Fourth Hidden Unit*)

Output weights ={{-6.46473, 1.08067, -4.88583, 2.57272}};

The mean error and mean percent deviation of the training set were $3,442,118 and 3.33% respectively. The estimated capital cost value of the test set was $66,415,200 as compared to an actual equipment cost of $72,236,860. The resulting error between the

estimated value and the actual cost is $5,953,400 and the percent deviation 8.22%. A listing of the computer run in Mathematica and the results of network training and testing are shown in Appendix C.

5.4.4 Case Study II - Summary of results

A summary of the results of the different methods used in the analysis of this study (the equipment factor method, the multiple linear regression method, and the neural network method), are listed in Table 5.12. Those results comprise the estimated capital cost of the test projects (P1 and P2 for the equipment factor method; P5 for both the neural net method and the multiple regression method), the error, and the percent deviation of that estimate from the actual cost.

Table 5.12 - Case study II - Summary of results

Method	Estimate	Actual cost	Error	Percent deviation
Equipment factor method				
P1	$54,239,698	$76,590,514	($22,350,816)	-29.18%
P2	$14,965,358	$27,725,000	($12,759,642)	-46.02%
Multiple linear regression- P5	$90,288,452	$72,368,648	$17,919,804	24.76%
Neural network-P5	$66,415,200	$72,368,648	($5,953,448)	-8.23%

5.5 CASE STUDY III: ESTIMATING THE MATERIAL AND INSTALLATION COSTS OF PRESSURE VESSELS

This case study deals with estimating the material and installation costs of pressure vessels, based on their weight, volume, and type of construction. As stated earlier in chapter 4, the material and installation costs of pressure vessels could be modeled as follows:

$$VC = F(W, V, T) \qquad [5.14]$$

Where VC = Total cost of the vessel

W = The weight of the vessel

V = The volume of the vessel

T = A weight assigned to the type of construction

Using these notations, cost estimates were prepared using two cost estimating relationships: (1) the power law and sizing model; and (2) the multiple linear regression analysis. In addition, a neural network application was used in making similar cost estimates. As mentioned earlier, twenty-seven data sets pertaining to three process plants projects were compiled from the files of a large chemical and industrial company. All of the three projects were completed in 1991; Hence, the cost vessels will be expressed in terms of 1991 dollars. The volume, weight, and the weighted value corresponding to a particular type of construction, as well as the material and installation cost of each of the twenty-seven vessels are listed in Table 5.13. The data is split into two sets: a learning set

and a test set. The learning set is comprised of eighteen vessels and the test set is comprised of nine vessels.

Table 5.13 - Case Study III - Vessels cost data

	Test set			
Vessel No.	**Volume, V**	**Weight, W**	**Type factor, T**	**Total Cost**
160VES35310	481	15000	12	$83,216
160VES35312	223	5300	12	$65,214
160VES35316	332	7800	8	$39,019
160VES35330	4411	86800	8	$253,686
160VES35332	126	5900	25	$156,031
161VES35216	126	17300	12	$87,512
161VES35217	1104	28740	6	$77,981
161VES35220	157	16900	8	$53,169
161VES35221	697	17500	12	$63,987
	Learning set			
160VES35305	3428	56700	2	$141,744
160VES35306	851	31800	12	$122,924
160VES35307	1325	19900	15	$132,064
160VES35308	1590	45100	6	$123,119
160VES35317	6465	55500	20	$305,892
160VES35334	236	7700	8	$38,154
160VES35337	71	3400	8	$31,722
160VES35338	3849	40500	12	$189,755
160VES35339	96	4200	8	$29,820
160VES35344	236	7200	8	$39,386
160VES35435	107	4200	6	$20,364
161VES35210	42	8000	12	$65,285
161VES35211	1781	26100	12	$124,108
161VES35214	495	68000	12	$207,033
161VES35226	47	3100	8	$24,255
161VES35228	2827	34500	12	$131,011
161VES35233	96	3060	6	$25,930
161VES35401	126	6000	8	$30,941

5.5.1 Power law and sizing model: Establishing the CER

Peters suggests that the purchased equipment costs for vessels are often modeled as a function of their weight. The cost of installation could also be correlated with the purchase cost of a given vessel. Based on this assumptions the cost function of [5.14] was reduced to:

$$VC = F(W) \qquad [5.15]$$

Using the power sizing method [5.15] translates into the following cost estimating relationship:

$$VC2 = VC1 * \left(\frac{W2}{W1}\right)^{m}, \quad 0 < m < 1 \qquad [5.16]$$

Where VC1, and W1 are the total cost and weight for a base vessel respectively. PC2 and W2 are the estimated total cost and weight for a new vessel of the test set. A value for m of 0.57 recommended by [Peters an Timmerhaus, 1980] for scaling the costs of pressure vessels, was used. It resulted in the following CER:

$$PC2 = PC1\left(\frac{W2}{W1}\right)^{0.57} \quad m=0.57 \qquad [5.17]$$

Another CER, found in [Bielefeld et al., 1992], was used in preparing estimates for the pressure vessels of the test set. This CER has the following form:

$$VC2 = VC1\frac{36908 + 0.67 * W2}{36908 + 0.67 * W1} \qquad [5.18]$$

Where VC1, and W1 are the total cost and weight for a base vessel respectively. PC2 and W2 are the estimated total cost and weight for a new vessel of the test set.

Table 5.14 - Case study III - Power law method cost estimates - Eq. [5.17]

Vessel No.	Weight	Total Cost	Estimated cost Eq. [5.17]	Error	Percent deviation
160VES35310	15000	$83,216	$64,167	$19,049	22.89
160VES35312	5300	$65,214	$35,463	$29,751	45.62
160VES35316	7800	$39,019	$44,201	($5,182)	13.28
160VES35330	86800	$253,686	$174,541	$79,145	31.20
160VES35332	5900	$156,031	$37,699	$118,332	75.84
161VES35216	17300	$87,512	$69,603	$17,909	20.46
161VES35217	28740	$77,981	$92,956	($14,975)	19.20
161VES35220	16900	$53,169	$68,681	($15,512)	29.17
161VES35221	17500	$63,987	$70,060	($6,073)	9.49
Mean values				$33,992	29.68
Base vessel					
161VES35233	3060	$25,930			

Table 5.15 - Case study III - Power law method cost estimates - Eq. [5.18]

Vessel No.	Weight	Total Cost	Estimated cost Eq. [5.18]	Error	Percent deviation
160VES35310	15000	$83,216	$31,255	$51,961	62.44
160VES35312	5300	$65,214	$26,929	$38,285	58.71
160VES35316	7800	$39,019	$28,044	$10,975	28.13
160VES35330	86800	$253,686	$63,273	$190,413	75.06
160VES35332	5900	$156,031	$27,196	$128,835	82.57
161VES35216	17300	$87,512	$32,280	$55,232	63.11
161VES35217	28740	$77,981	$37,382	$40,599	52.06
161VES35220	16900	$53,169	$32,102	$21,067	39.62
161VES35221	17500	$63,987	$32,369	$31,618	49.41
Mean values				$63,221	56.79
Base vessel					
161VES35233	3060	$25,930			

The base vessel was chosen to be the one with the smallest weight among all the vessels of this case study as shown in Tables 5.14 & 5.15. The total costs of the nine test vessels listed in Table 5.13 were estimated using the CERs of Eqs. [5.17] and [5.18] as shown in Tables 5.14 and 5.15 respectively. The mean errors and mean percent deviations of the estimates from actual vessel costs were $33,992 and 29.68 % for Eq [5.17] and $63,221 and 56.79% for Eq. [5.18].

5.5.2 Multiple linear regression: Establishing the CER

In this analysis, we assumed that the cost model of the total cost of a pressure vessel could be written as a linear function of the form:

$$V = a*W + b*V + c*T + d \qquad [5.19]$$

Where a, b, c, d are the linear regression coefficients of the cost function, and W, V, and T are the cost variables. Multiple linear regression analysis was used to determine the coefficients of equation [5.19] using the cost data of the vessels pertaining to the learning set.The resulting linear regression equation is the following:

$$VC = 2.183061*W + 11.62253*V + 2.183061*T - 35227 \qquad [5.20]$$

This equation has a correlation coefficient of 0.99, a mean error of $4736 and a mean percent deviation of 8.09 % for the learning set. The input values of the three cost variables of the nine vessels of the test set were supplied to Eq. [5.20] that produced corresponding cost estimates for each of the vessels as shown in Table 5.16. The cost

estimates produced by equation [5.20] for the test set resulted in a mean error of $6220 and a mean percent deviation of 9.22 %.

Table 5.16 - Case study III - Multiple linear regression method cost estimates Eq. [5.20]

Vessel No.	v	w	t	Actual Cost	Cost Estimated by [5.20]	Error	Percent deviation
				Test Set			
160VES35310	481	15000	12	$83,216	$86,406	3190	3.83
160VES35312	223	5300	12	$65,214	$62,232	2982	4.57
160VES35316	332	7800	8	$39,019	$41,196	2177	5.58
160VES35330	4411	86800	8	$253,686	$261,061	7375	2.91
160VES35332	126	5900	25	$156,031	$152,657	3374	2.16
161VES35216	126	17300	12	$87,512	$87,301	211	0.24
161VES35217	1104	28740	6	$77,981	$82,001	4020	5.16
161VES35220	157	16900	8	$53,169	$59,026	5857	11.02
161VES35221	697	17500	12	$63,987	$94,378	30391	47.50
Mean values						6620	9.22
				Learning set			
160VES35305	3428	56700	2	$141,744	$142,274	530	0.37
160VES35306	851	31800	12	$122,924	$127,388	4464	3.63
160VES35307	1325	19900	15	$132,064	$127,746	4318	3.27
160VES35308	1590	45100	6	$123,119	$123,364	245	0.20
160VES35317	6465	55500	20	$305,892	$299,912	5980	1.95
160VES35334	236	7700	8	$38,154	$39,855	1701	4.46
160VES35337	71	3400	8	$31,722	$28,551	3171	10.00
160VES35338	3849	40500	12	$189,755	$181,221	8534	4.50
160VES35339	96	4200	8	$29,820	$30,594	774	2.59
160VES35344	236	7200	8	$39,386	$38,763	623	1.58
160VES35435	107	4200	6	$20,364	$16,834	3530	17.34
161VES35210	42	8000	12	$65,285	$66,023	738	1.13
161VES35211	1781	26100	12	$124,108	$125,755	1647	1.33
161VES35214	495	68000	12	$207,033	$202,273	4760	2.30
161VES35226	47	3100	8	$24,255	$27,616	3361	13.86
161VES35228	2827	34500	12	$131,011	$156,251	25240	19.27
161VES35233	96	3060	6	$25,930	$14,222	11708	45.15
161VES35401	126	6000	8	$30,941	$34,866	3925	12.68
Mean Values						4736	8.09

5.5.3 Neural network application

A neural network application was developed to map the values of the cost variables (or the input vector) into a value of the purchase and installation costs of a given pressure vessel as follows:

$$\{W, V, T\} \rightarrow \{VC\} \qquad [5.21]$$

A three-layered feed forward back propagation neural network having a three-unit input vector, four-unit hidden layer, and a single unit output layer was used in training and testing the cost data. The network architecture is illustrated in Figure 5.5.

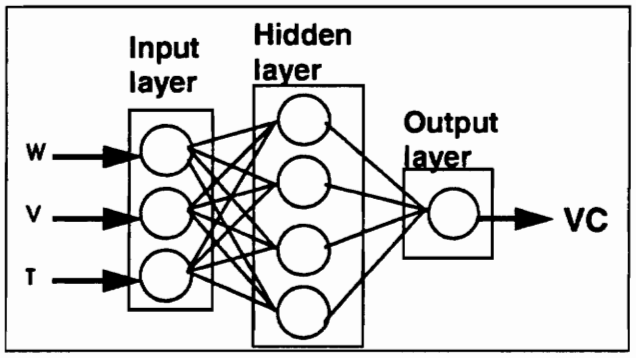


Figure 5.6 - Case Study III - Network Architecture

The training set comprised the cost data of eighteen vessels, and the test set comprised the cost data of nine vessels, as shown in Table 5.14. The input and output vectors of both sets were normalized as shown in Table 5.17. In normalizing the value of the cost a constant of $25,000 was added to the actual cost of each vessel in order to limit the normalized values of the desired output vector to values between 0.1 and 0.9.

Table 5.17 - Case Study III - Normalized data

	Test set			
Vessel No.	**Volume v**	**Weight w**	**Type weight t**	**Total Cost**
160VES35310	0.04807	0.15000	0.40000	0.27054
160VES35312	0.02227	0.05300	0.40000	0.22554
160VES35316	0.03322	0.07800	0.26667	0.16005
160VES35330	0.44108	0.86800	0.26667	0.69672
160VES35332	0.01257	0.05900	0.83333	0.45258
161VES35216	0.01257	0.17300	0.40000	0.28128
161VES35217	0.11045	0.28740	0.20000	0.25745
161VES35220	0.01571	0.16900	0.26667	0.19542
161VES35221	0.06970	0.17500	0.40000	0.22247
	Training Set			
160VES35305	0.34277	0.56700	0.06667	0.41686
160VES35306	0.08512	0.31800	0.40000	0.36981
160VES35307	0.13254	0.19900	0.50000	0.39266
160VES35308	0.15904	0.45100	0.20000	0.37030
160VES35317	0.64654	0.55500	0.66667	0.82723
160VES35334	0.02356	0.07700	0.26667	0.15789
160VES35337	0.00707	0.03400	0.26667	0.14181
160VES35338	0.38489	0.40500	0.40000	0.53689
160VES35339	0.00962	0.04200	0.26667	0.13705
160VES35344	0.02356	0.07200	0.26667	0.16097
160VES35435	0.01068	0.04200	0.20000	0.11341
161VES35210	0.00417	0.08000	0.40000	0.22571
161VES35211	0.17813	0.26100	0.40000	0.37277
161VES35214	0.04948	0.68000	0.40000	0.58008
161VES35226	0.00466	0.03100	0.26667	0.12314
161VES35228	0.28274	0.34500	0.40000	0.39003
161VES35233	0.00962	0.03060	0.20000	0.12733
161VES35401	0.01257	0.06000	0.26667	0.13985
Range	10000	100000	30	(VC+25000)/400000

The network was trained for 10,000 cycles using a learning rate of 0.2, a momentum coefficient of 0.9 and a sigmoidal transfer function for all the processing units. The hidden and output connection weights of the trained network are the following:

Hidden Weights ={

{-0.796648, -1.28414, -1.09229}, (*First hidden unit*)

{0.569639, 0.479947, 1.41274}, (*Second hidden unit*)

{-0.844284, -1.22041, -1.17892}, (*Third hidden unit*)

{-0.826069, -1.4214, -1.09667}}; (* Fourth hidden unit*)

Output Weights={{-2.49122, 2.82731, -2.8042, -3.27715}};

The mean error and mean percent deviation were $3,570 and 5.67% for the training set and $4,012 and 4.35% for the test set. The estimated cost values of the test set and training sets using the neural network technique, as well as the resulting error and percent deviation between the estimated value and the actual cost are listed in Table 5.18.

Table 5.18 - Case study III - Neural Network Cost Estimates

Vessel No.	V	W	T	Actual Cost	Neural network estimate	Error	Percent deviation
				Test Set			
160VES35310	481	15000	12	$83,216	$82,209	1007	1.21
160VES35312	223	5300	12	$65,214	$58,346	6868	10.53
160VES35316	332	7800	8	$39,019	$38,631	388	0.99
160VES35330	4411	86800	8	$253,686	$257,604	3918	1.54
160VES35332	126	5900	25	$156,031	$166,239	10208	6.54
161VES35216	126	17300	12	$87,512	$81,906	5606	6.41
161VES35217	1104	28740	6	$77,981	$74,295	3686	4.73
161VES35220	157	16900	8	$53,169	$52,275	894	1.68
161VES35221	697	17500	12	$63,987	$60,451	3536	5.53
Mean values						4012	4.35
				Learning set			
160VES35305	3428	56700	2	$141,744	$141,682	62	0.04
160VES35306	851	31800	12	$122,924	$127,670	4746	3.86
160VES35307	1325	19900	15	$132,064	$133,136	1072	0.81
160VES35308	1590	45100	6	$123,119	$119,083	4036	3.28
160VES35317	6465	55500	20	$305,892	$291,358	14535	4.75
160VES35334	236	7700	8	$38,154	$37,408	746	1.96
160VES35337	71	3400	8	$31,722	$29,189	2533	7.98
160VES35338	3849	40500	12	$189,755	$196,611	6856	3.61
160VES35339	96	4200	8	$29,820	$30,601	781	2.62
160VES35344	236	7200	8	$39,386	$36,615	2771	7.03
160VES35435	107	4200	6	$20,364	$20,945	581	2.85
161VES35210	42	8000	12	$65,285	$61,203	4082	6.25
161VES35211	1781	26100	12	$124,108	$129,413	5305	4.27
161VES35214	495	68000	12	$207,033	$205,103	1930	0.93
161VES35226	47	3100	8	$24,255	$28,525	4270	17.61
161VES35228	2827	34500	12	$131,011	$131,833	822	0.63
161VES35233	96	3060	6	$25,930	$19,440	6490	25.03
161VES35401	126	6000	8	$30,941	$33,588	2647	8.55
Mean Values						3570	5.67

A listing of the computer run in Mathematica and the results of network training and testing are shown in Appendix C.

5.5.4 Case Study III - Summary of results

A summary of the results of the different methods used in the analysis of this study (the power law and sizing method, the multiple linear regression method, and the neural network method), are listed in Table 5.19. Those results comprise the mean error and the mean percent deviation of the estimated vessel costs from actual costs.

Table 5.19 Case study III - Summary of results

	Learning set		Test set	
Method	**Mean error**	**Mean percent deviation**	**Mean error**	**Mean percent deviation**
Power law and sizing method				
Equation [5.17]	N/A	N/A	$33,992	29.68%
Equation [5.18]	N/A	N/A	$63,221	56.79%
Multiple linear regression	$4,732	8.09%	$6,620	9.22%
Neural network	$3,570	5.67%	$4,012	4.35%

5.6 CASE STUDY IV: ESTIMATING THE COST OF BRIDGES IN ROADWAY PROJECTS

This case study deals with estimating the costs of bridges in highway projects. The estimate is based on a preliminary section layout of a given bridge and the take-off quantities for some of the materials needed to build that bridge. As mentioned before in chapter 4, the total cost of a bridge pertaining to the bridges of this study could be modeled as a function of a number of cost variables as follows:

$$BC = F(S, DS, Np, Pw, Pt, SS, Bt) \quad [5.22]$$

Where BC = Total bridge cost

DS = Deck surface in Sq. Ft.

S = Average span in Ft.

Np = Number of piers

Pw = Average pier width in Ft.

Pt = Pier type (1= solid pier, 2= columns pier)

SS = Pounds of structural steel needed for the construction of the bridge

Bt = Bridge type (1= Cross-over, 2=Ramp)

Using these notations, cost estimates were prepared using a cost estimating relationship established by multiple linear regression analysis. In addition, a neural network application was used in making similar cost estimates.

As mentioned earlier, seventeen data sets pertaining to a highway project in Northern Virginia were compiled from the files of a large construction company. All of the bridges are currently under construction; Moreover, the actual cost of each bridge is assumed to be equal to the bid estimate obtained by a detailed estimating method as prepared and revised by the construction company. The cost variables of equation [5.22] as well as the bid estimate value of each of the seventeen bridges are listed in Table 5.20. The data is split into two sets: a learning set and a test set. The learning set is comprised of ten bridges and the test set is comprised of seven bridges.

Table 5.20 - Case Study IV - Bridges cost data

Bridge code	Bridge No.	Average span, S	Deck surface, DS	# of piers, Np	Average Pier Width, Pw	Piers Type, Pt	Structural steel, SS (lbs)	Bridge Type, Bt	Total Bridge Cost*, BC
Learning Set									
BN-1	Bridge 1	50.83	6664	2	47.25	1	170853	1	$562,383
BN-5	Bridge 3	105.833	13335	2	45.5	1	351936	1	$806,141
D-2	Bridge 5	104.25	7089	1	28.5	1	200146	1	$350,193
A-5	Bridge 7	139	11398	1	44.5	1	583878	2	$755,613
A-3	Bridge 8	228.17	28749	2	45.5	1	1786300	1	$2,547,744
CS-1	Bridge 10	158	17696	3	31.5	2	714500	1	$1,003,993
BS-1	Bridge 11	147	8820	1	47.75	2	377750	1	$510,736
BS-3	Bridge 12	69.2	8715	2	45.5	2	300000	1	$523,975
BR-1	Bridge 15	141	24592	2	56	1	933600	1	$1,642,759
CN-2	Bridge 17	122	10248	1	50.75	2	349500	1	$739,749
Test Set									
BN-2	Bridge 2	50.83	6456	2	45.1	1	162147	1	$516,711
D-1	Bridge 4	110	13210	1	65.5	2	493480	2	$832,565
A-7	Bridge 6	53.52	6423	2	43.5	1	133800	1	$479,018
A-6	Bridge 9	111	13320	2	43.5	2	410300	2	$627,737
BS-4	Bridge 13	69.2	8715	2	45.5	2	300000	1	$562,817
BR-2	Bridge 14	141	17808	2	45.5	1	759300	1	$1,378,773
CN-4	Bridge 16	187	19604	1	55.5	2	952600	1	$1,460,347

*Bid estimate

5.6.1 Multiple linear regression: Establishing the CER

In this analysis, we assume that the cost model of the total cost of a bridge could be written as a linear function of the form:

$$BC = a*S + b*DS + c*Np + d*Pw + e*Pt + f*SS + g*Bt + h \qquad [5.23]$$

Where a, b, c, d, e, f, g, and h are the linear regression coefficients of the cost function,

and S, DS, Np, Pw, Pt, SS, and Bt are the cost variables,

and BC is the total bridge cost.

Multiple linear regression analysis was used to determine the coefficients of equation [5.23] using the cost data of the bridges pertaining to the learning set. The resulting linear regression equation is the following:

$$BC = -956.77*S + 16.64*DS + 10163.41*Np + 8427*Pw - 120905*Pt + 1.1200347*SS - 209974.9*Bt + 200604.3 \qquad [5.24]$$

This equation has a correlation coefficient of 0.99, a mean error of $34,993 and a mean percent deviation of 5.231 % for the learning set. The input values of the seven cost variables of the seven bridges of the test set were supplied to Eq. [5.24] that produced corresponding cost estimates for each of the bridges as shown in Table 5.21. The cost estimates produced by equation [5.24] for the test set resulted in a mean error of $47,346 and a mean percent deviation of 5.94%.

Bridge code	Bridge No.	Average span, S	Deck surface, DS	# of piers, Np	Average Pier Width, Pw	Piers Type, Pt	Structural steel, SS (lbs)	Bridge Type, Bt	Total Bridge Cost, BC	Cost Estimate Eq. [5.22]	Error	Percent Deviation
Learning Set												
BN-1	Bridge 1	50.83	6664	2	47.25	1	170853	1	$562,383	$541,892	20490.57	3.64
BN-5	Bridge 3	105.833	13335	2	45.5	1	351936	1	$806,141	$788,369	17771.89	2.20
D-2	Bridge 5	104.25	7089	1	28.5	1	200146	1	$350,193	$362,485	12291.94	3.51
A-5	Bridge 7	139	11398	1	44.5	1	583878	2	$755,613	$755,613	0.00	0.00
A-3	Bridge 8	228.17	28749	2	45.5	1	1786300	1	$2,547,744	$2,534,415	13329.45	0.52
CS-1	Bridge 10	158	17696	3	31.5	2	714500	1	$1,003,993	$988,403	15589.58	1.55
BS-1	Bridge 11	147	8820	1	47.75	2	377750	1	$510,736	$590,637	79901.48	15.64
BS-3	Bridge 12	69.2	8715	2	45.5	2	300000	1	$523,975	$567,446	43471.11	8.30
BR-1	Bridge 15	141	24592	2	56	1	933600	1	$1,642,759	$1,682,059	39299.97	2.39
CN-2	Bridge 17	122	10248	1	50.75	2	349500	1	$739,749	$631,966	107783.00	14.57
Mean Values											34992.90	5.23
Test Set												
BN-2	Bridge 2	50.83	6456	2	45.1	1	162147	1	$516,711	$510,553	6158.114	1.19
D-1	Bridge 4	110	13210	1	65.5	2	493480	2	$832,565	$768,339	64226.45	7.71
A-7	Bridge 6	53.52	6423	2	43.5	1	133800	1	$479,018	$462,200	16817.82	3.51
A-6	Bridge 9	111	13320	2	43.5	2	410300	2	$627,737	$500,811	126926.27	20.22
BS-4	Bridge 13	69.2	8715	2	45.5	2	300000	1	$562,817	$567,446	4629.11	0.82
BR-2	Bridge 14	141	17808	2	45.5	1	759300	1	$1,378,773	$1,285,434	93338.52	6.77
CN-4	Bridge 16	187	19604	1	55.5	2	952600	1	$1,460,347	$1,441,023	19323.59	1.32
Mean values											47345.70	5.94

5.6.2 Neural network application

A neural network application was developed to map the values of the cost variables (or the input vector) into a value of the total cost of a given bridge as follows:

$$\{S, DS, Np, Pw, Pt, SS, Bt\} \rightarrow \{BC\} \qquad [5.25]$$

A three-layered feed forward back propagation neural network having a seven-unit input layer, a six-unit hidden layer, and a single unit output layer was used in training and testing the cost data. The network architecture is illustrated in Figure 5.6.

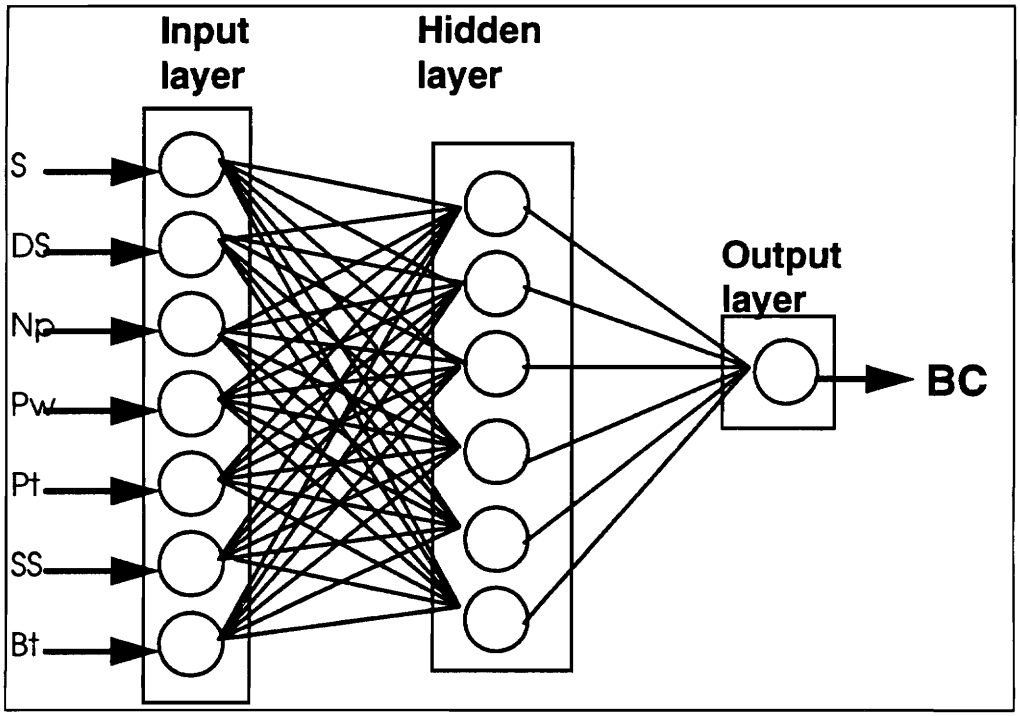


Figure 5.7 - Case Study IV - Network Architecture

The input and output vectors of the learning and test sets were normalized as shown in Table 5.22. In normalizing the value of the cost a constant of $500,000 was added to the actual cost of each bridge in order to limit the normalized values of the desired output vector to values between 0.1 and 0.9.

Table 5.22 - Case Study IV - Normalized data

		Input							Output
Bridge code	**Bridge No.**	**Average span, S**	**Deck surface, DS**	**# of piers, Np**	**Average Pier Width, Pw**	**Piers Type, Pt**	**Structural steel, SS (lbs)**	**Bridge Type, Bt**	**Total Bridge Cost*, BC**
					Learning Set				
BN-1	Bridge 1	0.2033	0.2221	0.5000	0.6300	0.4000	0.0854	0.4000	0.2291
BN-5	Bridge 3	0.4233	0.4445	0.5000	0.6067	0.4000	0.1760	0.4000	0.3104
D-2	Bridge 5	0.4170	0.2363	0.2500	0.3800	0.4000	0.1001	0.4000	0.1584
A-5	Bridge 7	0.5560	0.3799	0.2500	0.5933	0.4000	0.2919	0.8000	0.2935
A-3	Bridge 8	0.9127	0.9583	0.5000	0.6067	0.4000	0.8932	0.4000	0.8909
CS-1	Bridge 10	0.6320	0.5899	0.7500	0.4200	0.8000	0.3573	0.4000	0.3763
BS-1	Bridge 11	0.5880	0.2940	0.2500	0.6367	0.8000	0.1889	0.4000	0.2119
BS-3	Bridge 12	0.2768	0.2905	0.5000	0.6067	0.8000	0.1500	0.4000	0.2163
BR-1	Bridge 15	0.5640	0.8197	0.5000	0.7467	0.4000	0.4668	0.4000	0.5893
CN-2	Bridge 17	0.4880	0.3416	0.2500	0.6767	0.8000	0.1748	0.4000	0.2882
Mean Values									
					Test Set				
BN-2	Bridge 2	0.2033	0.2152	0.5000	0.6013	0.4000	0.0811	0.4000	0.2139
D-1	Bridge 4	0.4400	0.4403	0.2500	0.8733	0.8000	0.2467	0.8000	0.3192
A-7	Bridge 6	0.2141	0.2141	0.5000	0.5800	0.4000	0.0669	0.4000	0.2013
A-6	Bridge 9	0.4440	0.4440	0.5000	0.5800	0.8000	0.2052	0.8000	0.2509
BS-4	Bridge 13	0.2768	0.2905	0.5000	0.6067	0.8000	0.1500	0.4000	0.2293
BR-2	Bridge 14	0.5640	0.5936	0.5000	0.6067	0.4000	0.3797	0.4000	0.5013
CN-4	Bridge 16	0.7480	0.6535	0.2500	0.7400	0.8000	0.4763	0.4000	0.5284
Range		250	30000	4	75	2.5	2000000	2.5	(C+500000) /3000000

*Bid Estimate

The network was trained for 8,000 cycles using a learning rate of 0.3, a momentum coefficient of 0.9, and a sigmoidal transfer function for all the processing units. The resulting hidden and output connection weights of the trained network are the following:

Hidden Weights ={

{{0.327668, -0.426516, 0.29873, 1.29571, 0.11342, 1.55933, -0.0870809},

(*First hidden unit*)

{-0.555733, -0.336815, -0.393273, -0.43195, -0.139424, -0.721752, -0.207788},

(*Second hidden unit*)

{-0.753077, -0.1738, -0.4109, -0.608726, 0.00505423, -1.02603, 0.0258896},

(*Third hidden unit*)

{-0.762064, -0.139216, -0.398818, -0.739388, 0.0833498, -1.05328, 0.0672192},

(*Fourth hidden unit*)

{-0.685056, -0.219926, -0.351675, -0.615592, 0.054432, -0.956148, -0.127042},

(*Fifth hidden unit*)

{-0.253802, -0.156687, 0.843217, 1.34814, 1.57577, -2.97151, 1.56144}};

(*Sixth hidden unit*)

Output Weights= {3.94483, -0.580749, -1.32537, -1.44981, -1.04605, -3.26669};

The mean error and mean percent deviation were $42962 and 6.32% for the training set and $26626 and 2.66% for the test set. The estimated cost values of the test set and training sets using the neural network technique, as well as the resulting error and percent deviation between the estimated value and the actual cost are listed in Table 5.23.

Bridge code	Bridge No.	Average span, S	Deck surface, DS	# of piers, Np	Average Pier Width, Pw	Piers Type, Pt	Structural steel, SS (lbs)	Bridge Type, Bt	Total Bridge Cost, BC	Cost Estimate Eq. [5.22]	Error	Percent Deviation
Learning set												
BN-1	Bridge 1	50.83	6664	2	47.25	1	170853	1	$562,383	$530,558	31825.00	5.66
BN-5	Bridge 3	105.833	13335	2	45.5	1	351936	1	$806,141	$781,749	24392.00	3.03
D-2	Bridge 5	104.25	7089	1	28.5	1	200146	1	$350,193	$381,337	31144.00	8.89
A-5	Bridge 7	139	11398	1	44.5	1	583878	2	$755,613	$758,623	3010.00	0.40
A-3	Bridge 8	228.17	28749	2	45.5	1	1786300	1	$2,547,744	$2,510,310	37434.00	1.47
CS-1	Bridge 10	158	17696	3	31.5	2	714500	1	$1,003,993	$986,315	17678.00	1.76
BS-1	Bridge 11	147	8820	1	47.75	2	377750	1	$510,736	$582,375	71639.00	14.03
BS-3	Bridge 12	69.2	8715	2	45.5	2	300000	1	$523,975	$568,058	44083.00	8.41
BR-1	Bridge 15	141	24592	2	56	1	933600	1	$1,642,759	$1,686,010	43251.00	2.63
CN-2	Bridge 17	122	10248	1	50.75	2	349500	1	$739,749	$614,588	125161.00	16.92
Mean Values											42961.70	6.32
Test Set												
BN-2	Bridge 2	50.83	6456	2	45.1	1	162147	1	$516,711	$504,061	12650.00	2.45
D-1	Bridge 4	110	13210	1	65.5	2	493480	2	$832,565	$809,602	22963.00	2.76
A-7	Bridge 6	53.52	6423	2	43.5	1	133800	1	$479,018	$470,100	8918.00	1.86
A-6	Bridge 9	111	13320	2	43.5	2	410300	2	$627,737	$616,371	11366.00	1.81
BS-4	Bridge 13	69.2	8715	2	45.5	2	300000	1	$562,817	$568,058	5241.00	0.93
BR-2	Bridge 14	141	17808	2	45.5	1	759300	1	$1,378,773	$1,329,430	49343.00	3.58
CN-4	Bridge 16	187	19604	1	55.5	2	952600	1	$1,460,347	$1,384,446	75901.00	5.20
Mean values											26626	2.66

5.6.3 Case Study IV - Summary of results

A summary of the results of the different methods used in the analysis of this study (the multiple linear regression method, and the neural network method), are listed in Table 5.24. Those results comprise the mean error and the mean percent deviation of the estimated vessel costs from actual costs.

Table 5.24 Case study IV - Summary of results

	Learning set		Test set	
Method	**Mean error**	**Mean percent deviation**	**Mean error**	**Mean percent deviation**
Multiple linear regression	$34,993	5.23%	$47,346	5.94%
Neural network	$42,962	6.32%	$26,626	2.66%

CHAPTER 6

SUMMARY AND DISCUSSION OF RESULTS, CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

6.1 INTRODUCTION

This chapter presents a complete review of the results of the analysis and experimentation process performed on the four case studies, a discussion of those results and their significance, as well as the contribution of this thesis to the body of knowledge of construction engineering and management. Suggestions for further work and future research opportunities will be also outlined.

The main objective of this thesis is to explore the possibility of applying neural networks to cost estimating problems and the efficiency and accuracy of the neural network computing technology in producing good estimates. The process of developing neural networks applications to cost estimating problems required a parameter-based approach in modeling the cost of a specific problem as a function of a number of cost variables. Due to this requirement, the results of the neural network approach, for the four case studies mentioned earlier, were compared to the results achieved by parameter-based cost estimating techniques commonly used in parametric cost estimating applications.

Both methodologies, parametric estimating techniques and neural networks used the historical data collected for each case study in order to establish a cost estimating relationship and a trained neural network respectively. It should be noted

here that in the case of parametric estimating techniques, the CER of the equipment factor method was not derived from the historical cost data of the study in which it was used (capital cost estimating of process plants), but it was obtained from an extensive study performed by [Cran, 1981]. In addition, the CERs of the power law and sizing model were adopted from [Peters et al., 1980] and a study in [Bielefeld, 1992], with the base cost item of these CERs taken from the actual historical sets of the case studies in which these CERs were tested. The analysis performed using multiple linear regression analysis in establishing a linear cost estimating relationship and the training of a neural network able of producing future cost estimates was performed using a historical set of cost items selected randomly from the complete set of the cost data of each study. Table 6.1 presents the number of data items for each study, the number of data items used in training and testing the estimating method for a particular study, and the table where the results of the described method are listed.

Table 6.1 - Estimating methods, data sets and tabulated results

Study	Method	Total No. of data items	No. of training data items	No. of test data items	Tabulated Results
Case Study I					
	Power law	5	4	1	Tables 5.3 & 5.4
	Multiple linear regression	5	4	1	Tables 5.6
	Neural network	5	4	1	Tables 5.6
	Summary				Tables 5.6
Case Study II					
	Equipment factor method	5	None	2	Tables 5.10
	Multiple linear regression	5	4	1	Tables 5.12
	Neural network	5	4	1	Tables 5.12
	Summary				Tables 5.12
Case Study III					
	Power law	27	None	9	Tables 5.14 & 5.15
	Multiple linear regression	27	18	9	Table 5.16
	Neural network	27	18	9	Table 5.18
	Summary				Table 5.19
Case Study IV					
	Multiple linear regression	17	10	7	Table 5.21
	Neural network	17	10	7	Table 5.23
	Summary				Table 5.24

6.2 SUMMARY AND DISCUSSION OF RESULTS

The discussion and evaluation of the results of neural networks and parametric estimating techniques were based on two basic criteria:

1. A mean value of the absolute error between the cost estimates of a given estimating technique and the actual costs of the test set of each study.

2. A mean value expressed as a percentage of the deviation of the cost estimates of a given estimating technique from the actual costs of the test set of each case study.

The mean absolute error is computed as follows:

$$\text{Mean absolute error} = \sum_{i=1}^{n} \frac{|x_i - x_{io}|}{\text{n}} \qquad [6.1]$$

Where i = 1..n, for n data items

x_i = Cost estimate of data item i

x_{io} = Actual cost of data item I

The mean percent deviation is computed as follows:

$$\text{Mean percent deviation} = \sum_{i=1}^{n} \frac{\left(\frac{|x_i - x_{io}| * 100}{x_{io}}\right)}{n} \qquad [6.2]$$

Where i = 1..n, for n data items

x_i = Cost estimate of data item i

x_{io} = Actual cost of data item i

We note here that for the first two case studies, only one data item was tested. Due to this fact, there was no need to compute a mean value for the absolute error and the percent deviation, instead, the error and percent deviation of the cost estimate from the actual cost of the test data item were computed.

The methodology and the process of producing the cost estimates of the different methods used in each of the four case studies were fully described and presented in chapters 4 & 5. In addition, a table summarizing the results achieved in each study was presented at the end of the section dealing with the analysis of that study. Based on these tables, the complete results, in terms of the mean absolute errors and percent deviations of all the cost estimates produced by the different methods for all the case studies are presented in Table 6.2.

Table 6.2 - Summary of results - Case studies I, II, III, and IV

Study	Method	Data set evaluated	Mean absolute error/error	Mean percent deviation
Case Study I	**Estimating the purchase cost of major equipment for process plants**			
	Power law			
	m=0.6, Eq. [5.5]	{P5}	$6,063,859	79.96%
	m=1.0, Eq. [5.6]	{P5}	$3,919,216	51.68%
	Multiple linear regression	{P5}	$6,448,419	85.03%
	Neural network	{P5}	$856,936	11.30%
Case Study II	**Estimating the capital costs of process plants**			
	Equipment factor method	{P1,P2}	$17,555,230	37.60%
	Multiple linear regression	{P5}	$17,919,804	24.76%
	Neural network	{P5}	$5,953,448	8.23%
Case Study III	**Estimating the material and installation costs of pressure vessels**			
	Power law			
	m=0.57, Eq. [5.17]	{9 vessels}, test set	$33,992	29.68%
	Eq. [5.18]	{9 vessels}, test set	$63,221	56.79%
	Multiple linear regression	{9 vessels}, test set	$6,620	9.22%
		{18 vessels}, learning set	$4,735	8.09%
	Neural network	{9 vessels}, test set	$4,012	4.35%
		{18 vessels}, learning set	$3,570	5.67%
Case Study IV	**Estimating bridge costs in roadway projects**			
	Multiple linear regression	{7 bridges}, test set	$47,346	5.94%
		{10 bridges}, learning set	$34,992	5.23%
	Neural network	{7 bridges}, test set	$26,626	2.66%
		{10 bridges}, learning set	$42,962	6.32%

The results presented in Table 6.2 allow us to start the comparison process between the performance of neural networks and the parametric estimating techniques used in the different studies. First, we will define two percent improvement indices, PIE and PIP. PIE indicates the amount of decrease or increase in the value of the mean absolute error produced by the neural network method as compared to that the parametric estimating technique used in the same study for a similar data set. On the

other hand, PIP indicates the amount of decrease or increase in the value of the mean percent deviation produced by the neural network method as compared to that of the parametric estimating technique used in the same study for a similar set.

$$PIE = \frac{(\text{Mean absolute error}_{\text{parametric method}} - \text{Mean absolute error}_{\text{neural network}}) * 100}{\text{Mean absolute error}_{\text{parametric method}}}$$

$$PIP = \frac{(\text{Mean percent deviation}_{\text{parametric method}} - \text{Mean percent deviation}_{\text{neural network}}) * 100}{\text{Mean percent devaition}_{\text{parametric method}}}$$

The values of PIE and PIP for the neural networks applications of the four case studies were computed for the four case studies and are listed in Table 6.3.

Table 6.3 - Improvement indices, PIE & PIP of the neural network method for Case studies I, II, III, and IV.

Study	Method	Data set evaluated	PIE	PIP
Case Study I	**Estimating the purchase cost of major equipment for process plants**			
	Power law			
	m=0.6, Eq. [5.5]	{P5}	85.87%	85.87%
	m=1.0, Eq. [5.6]	{P5}	78.14%	78.14%
	Multiple linear regression	{P5}	86.71%	86.71%
Case Study II	**Estimating the capital costs of process plants**			
	Equipment factor method	{P1,P2}	66.09%	78.11%
	Multiple linear regression	{P5}	66.78%	66.76%
Case Study III	**Estimating the material and installation costs of pressure vessels**			
	Power law	{9 vessels}, test set	88.20%	85.34%
	m=0.57, Eq. [5.17]	{9 vessels}, test set	93.65%	92.34%
	Eq. [5.18]	{9 vessels}, test set	39.40%	52.82%
	Multiple linear regression	{18 vessels}, learning set	24.60%	29.91%
Case Study IV	**Estimating bridge costs in roadway projects**			
	Multiple linear regression	{10 bridges}, learning set	-22.78%	-20.84%
		{7 bridges}, test set	43.76%	55.18%

It should be noted here, that the negative values PIE and PIP indeces for the learning set are not a definite indication of a poor performance exhibited by the network. In using the neural network technique, there is always a possibility in reducing the amount of error and deviation either by increasing the number of cycles or reducing the value of the learning parameter, but this approach may lead to "over-training" the network, the fact that will have negative effects on the results produced in the testing phase.

In the first case study, it can be clearly seen that the neural network outperformed the power law and multiple linear regression methods and presented strong predictive capabilities. Both parametric estimating techniques, the power law and multiple regression and using the same type of historical data (process plants) failed to establish a CER capable of predicting the cost of major equipment of a process plant within reasonable accuracy. The mean percent deviation for the power law and sizing method ranged between 51.68% and 79.78% and that of the multiple linear regression method was 85.03%. Those values suggest the general form of the CER used in each method does not match the cost data of this study. On the other hand, the neural network method, which is not restricted to a specific mathematical model for the cost function, was capable of providing a reasonably accurate estimate for the purchase cost of the major equipment of the test project of this study. The percent deviation of the neural network estimate from the actual cost of equipment of the test project was 11.3%. The mean absolute errors and percent deviations for this study are illustrated in Figure 6.1.

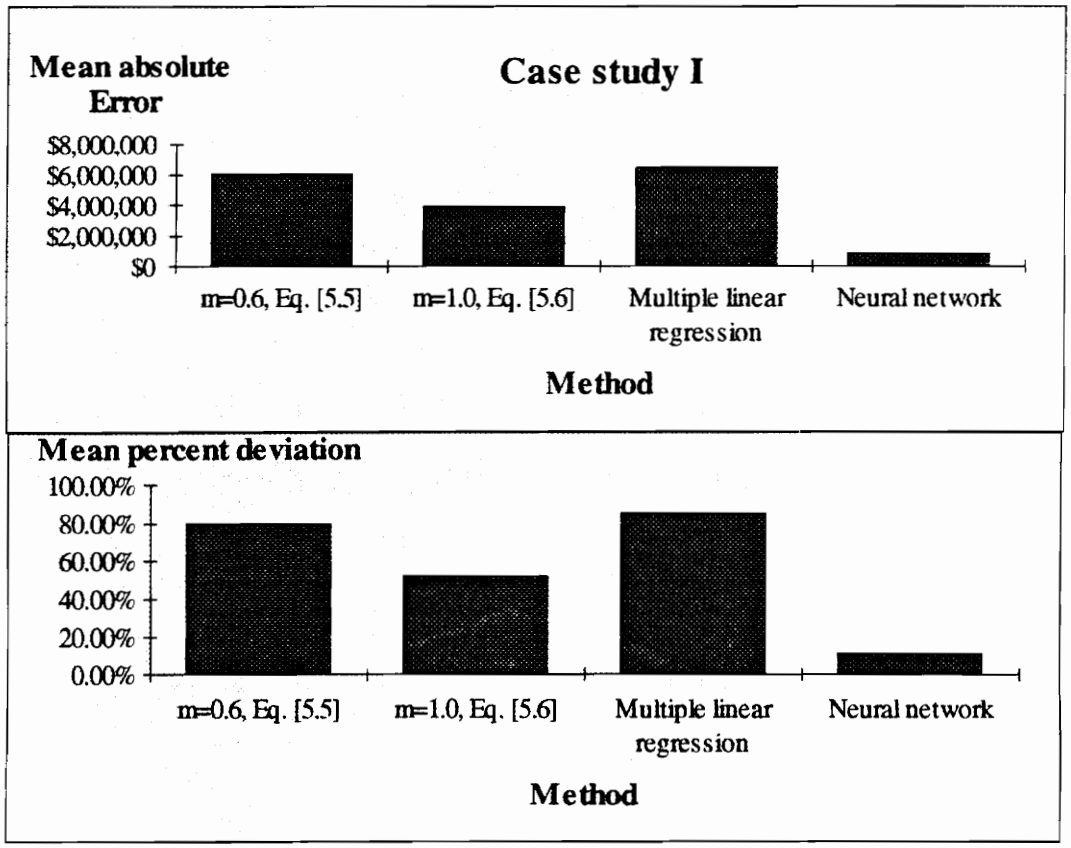


Figure 6.1 - Mean absolute error and mean percent deviation - Case study I

In the second case study, the equipment factor method which is traditionally used in preparing preliminary capital cost estimates for industrial plants, performed exactly as expected, by having a mean percent deviation of 37.6%. This figure represents a slight deviation from what [Peters et al., 1980] suggests as an accuracy range (at ±30 %) for using this method. The multiple linear regression method performed better than the equipment factor method, and the linear CER resulting from this method was not a very bad assumption for the general form of the proposed cost model. The mean percent deviation of the regression method was 24.6%. The neural network technique succeeded, once again, in providing the best capital cost estimate for the test project of this study. The accuracy of the neural network

estimate as compared to the actual cost was 8.3%. This accuracy is the result of the good mapping capabilities of the neural network. During training, the network learns to match closely a given set of cost variables with the corresponding actual cost of that set. When we present the network with a new set of cost variables, the network maps those variables into a cost value according to the matching process established in the learning phase.

The mean absolute errors and percent deviations for this study are illustrated in Figure 6.2.

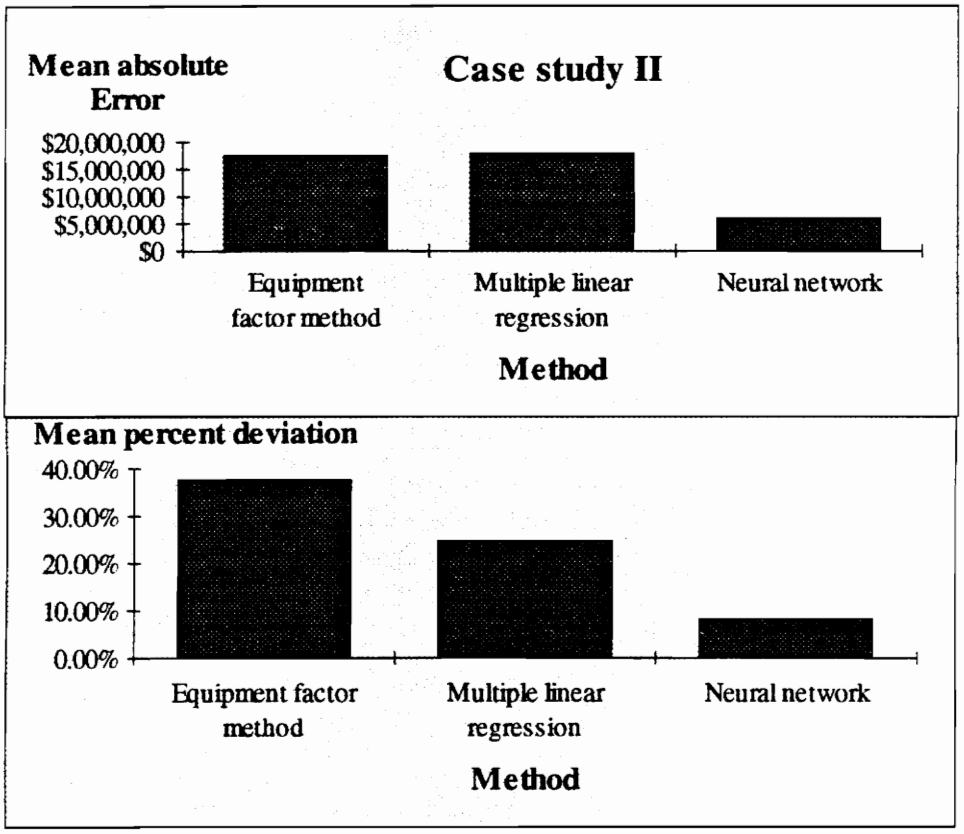


Figure 6.2 - Mean absolute error and mean percent deviation - Case study II

The third case study which deals with estimating the material and installation costs of pressure vessels, showed a better performance by the parametric estimating techniques that produced more accurate cost estimates than in the two previous case studies.

The power law method, and the corresponding power coefficient (m=0.57) proposed by [Peters et al.], resulted in a mean percent deviation of 29.68%, which is typical for this type of estimating techniques having a typical accuracy range of ±30% [Jelen et al., 1983]. Previous studies showed that the cost of pressure vessels could be correlated with the weight of the vessel when no enough information is available to produce an accurate estimate of its cost [Ostwald, 1980]. A CER proposed by a study found in [Bielefeld et al., 1992], was used as a different form of the CERs usually used in the power law and sizing model. This model suggests that the cost of two vessels is proportional to the ratio of two linear functions expressed in terms of their weight (Eq. [5.18]). This cost relationship didn't perform well in this study, and resulted in a mean percent deviation of 56.79%. This poor performance could be explained by the fact that linear scaling does not fit the data of this study. The multiple regression methods exhibited very good performance in correlating the cost data of the learning resulting in mean percent deviation of 8.09% and a mean absolute error of $4735, but failed to produce similar results for the testing set with the mean percent deviation increasing to 9.22% and the mean absolute error to $6620; but in general, one can say that, for this study, the regression method resulted in good correlations and produced good estimates. Once again, the neural network technique, outperformed all the other estimating methods used in this study and resulted in good

predictions for both sets: the training set and the test set. The neural network resulted in a mean absolute error of $3570 and $4012 and a mean percent deviation of 5.67% and 4.35% for the learning and test sets respectively. This superior performance could also be linked to the fact that the network is not bound to a specific mathematical form for the cost function modeling the material and installation costs of the vessels of this study.

The mean absolute errors and percent deviations for this study are illustrated in Figure 6.3.

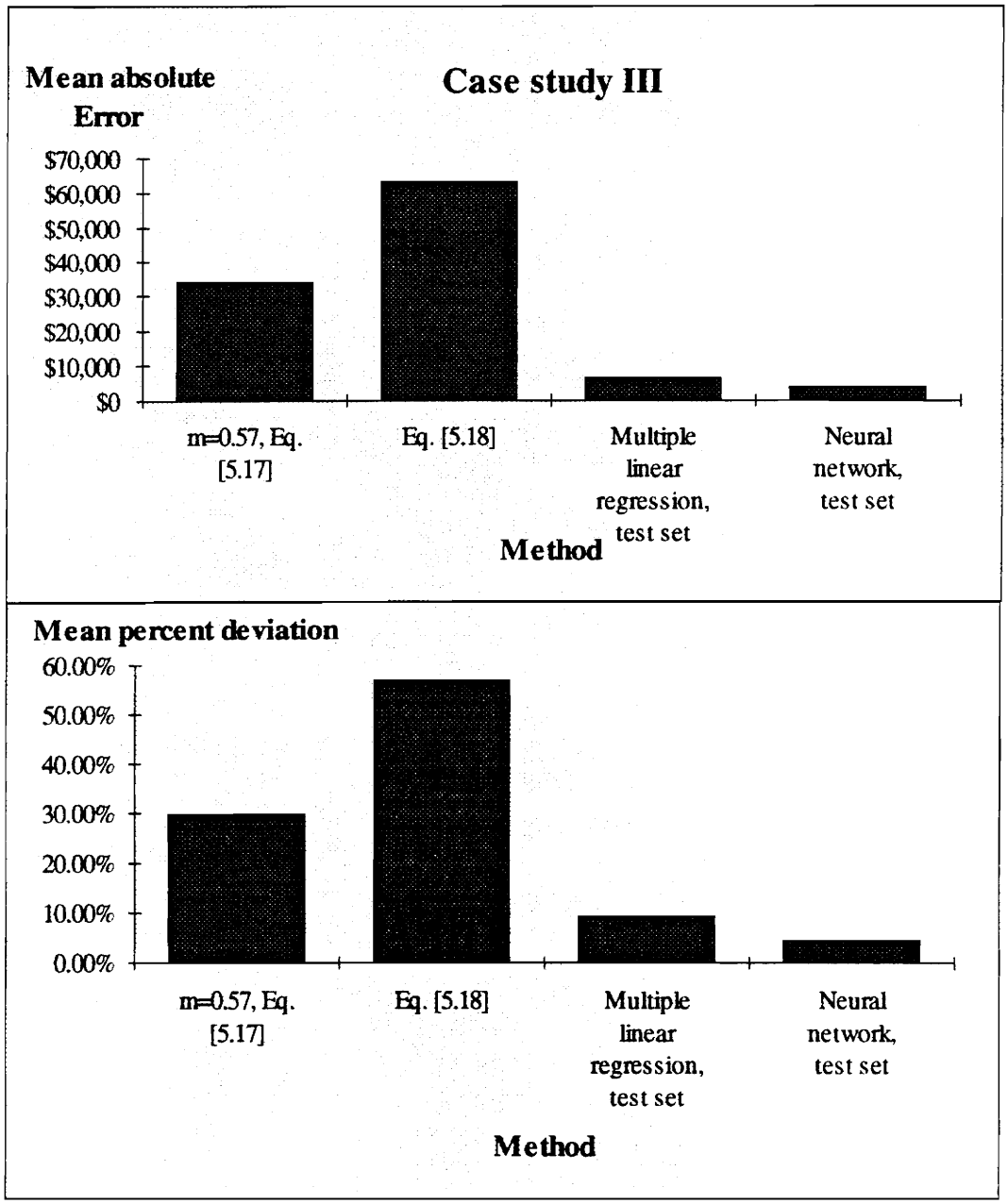


Figure 6.3 - Mean absolute error and mean percent deviation - Case study III

The fourth case study could be classified into a separate type of projects since it deals with the cost of bridges in roadway projects, whereas the first three case studies dealt with industrial projects. Multiple linear regression analysis was used as

the parametric estimating technique in establishing the CER of the cost function proposed for the parametric modeling of the total cost of bridges in this study. This method resulted in excellent correlations for the training set, but failed to produce similar results for the test set with the mean percent deviation increasing from 5.23% for the learning set to 5.94% for the training set, and the mean error increasing from $34,992 for the learning set to $47,236 for the test set. Even though the neural network had more error and percent deviation than the multiple regression method during the training phase, it resulted in substantial error reductions in the testing phase coupled with a mean percent deviation of 2.26%.

The mean absolute errors and percent deviations for this study are illustrated in Figure 6.4.

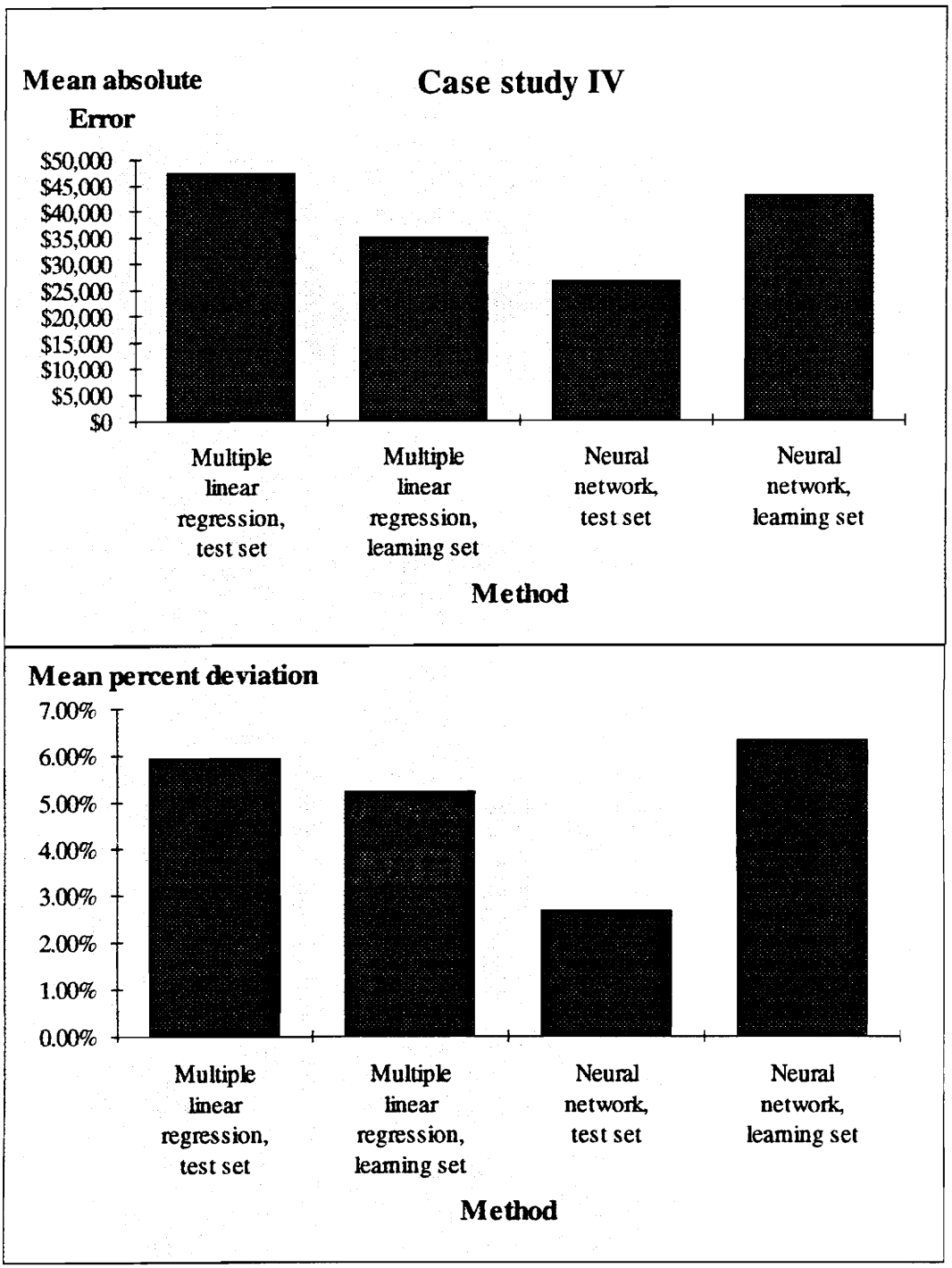


Figure 6.4 - Mean absolute error and mean percent deviation - Case study IV

6.3 FINDINGS AND CONTRIBUTIONS OF THIS THESIS TO THE FIELD OF CONSTRUCTION ENGINEERING AND MANAGEMENT

The parametric method of cost estimating has been used in the past, and is still used today for a limited types of cost estimates and when there is not enough time to use an alternate estimating method. The parametric estimating methodology has been used in applications involving the determination of order of magnitude or preliminary cost estimates in a short period of time and with no or little specific information about the system that is being evaluated. Parametric estimating techniques use historical cost data of previous projects in order to establish a cost estimating relationship (CER) capable of generating cost estimates for future projects of the same type. It is always required that the general mathematical form of the CER be defined before any parametric analysis or correlation be performed to determine the cost function that best fits the historical cost data. However, modeling the cost of a system or a structure or even a module as a function of a number of independent variables is not an easy task. This is due to the large number of variables present in the system under evaluation, and the numerous interactions among these variables. Another consideration is that cost estimating relationships are often modeled according to a single mathematical form that is used for all the cost variables of the cost function, like linear or non-linear forms for example. In most systems, the cost variables that are used in modeling the cost, often have a different mathematical correlation with the cost of that system. For example, the change in the system's cost may be a linear function of changes in a particular cost variable, and non-linear function for another variable, or maybe a hyperbolic, step, impulse or discontinuous function for other variables. Moreover, sometimes it is not possible to express the cost model in

mathematical terms or it is very difficult to find a mathematical function capable of describing the cost models. The major disadvantage of using parametric estimating technique is that there are is no specific or a clearly defined approach that will help us make the correct choice of the cost model that best fits the historical data of a given cost estimating application and the efficiency of this model in producing future cost estimates that are reasonably accurate. These considerations might explain the limited used of parametric estimating techniques in the construction industry and their relatively low accuracy range that most reviews and publications rate in the order of ± 30%.

The neural network technique presents itself as a new paradigm of computation that has the potential of resolving some of the major drawbacks of the parametric estimating methodology and holds great promise for rendering the parametric method of cost estimating a reliable and reasonably accurate method in preparing cost estimates. The neural network computing technology eliminates the need for finding a *good* cost estimating relationship that mathematically describes the cost of a system as a function of the variables that have the most impact on the cost of that system. In addition, the neural network technique has no restrictions on the number of cost variables or the relevance of choosing these variables in modeling the cost function, this is due to the ability of the neural network of self-organization and learning. A neural network internally organizes itself to be able to generalize rules from the cases on which it was trained and apply those rules to new stimuli. Those rules are generally constructed by assigning different weights to the connections between the processing elements of a neural net. The significance of these weights is that the effect of irrelevant inputs is minimized by assigning small weights to the

connections closest to these inputs and the effect of the inputs having an immediate impact on the output of the network is maximized by assigning higher weights to their connections. The assignment of connection weights creates an internal balance within the network in such a way that noise and high deviations in the values of the network's outputs from the actual outputs are minimized. The application of neural networks to parametric cost estimating problems was a natural choice due to the features and characteristics of the neural network computing technology that holds a great promise for a successful implementation to cost estimating applications.

This thesis examined the potential applications of neural networks to cost estimating problems by developing neural networks applications for a number of case studies constructed from the historical cost data of actual construction projects. The findings showed that the neural network technique presented strong predictive capabilities and a high accuracy level in producing cost estimates for cases belonging to the same class of the cases on which the network has been trained. The major findings of this thesis could be summarized as follows:

- Neural networks outperformed traditional parametric estimating techniques, like the power law and sizing method, regression methods, and the equipment factor method, by producing more accurate estimates and lesser deviations from actual costs.

- The neural network technique was consistent throughout the analysis of the four case studies with an accuracy ranging between -9 and +11 % and an average accuracy of 6.42 % for all the studies.

- In the case where the network is presented with test cases possessing features that were not present in any of the training cases, the results were discouraging.

- High inconsistency was observed in the implementation of regression methods and scaling techniques, which presented high fluctuations in their accuracy level from one case study to the other, in particular the multiple linear regression method.

- The possibilities of further improvement in the accuracy of neural networks in producing cost estimates is always an option by changing the network architecture or the training parameters; Whereas, once a CER is established by a parametric estimating technique, no further improvements could be achieved to give a better accuracy level for that CER.

- The neural network was more accurate in producing estimates for test cases than the parametric techniques that resulted in very good correlations for the historical cases but failed to maintain the same level of accuracy and correlation for the cost data of the test cases.

- Large improvements in the accuracy level have been achieved through the use of neural networks by reducing the deviations of cost estimates produced by using a parametric estimating methodology from ±30% to less than ±10%.

These findings highly suggest that the implementation of neural network to cost estimating applications should be seriously considered when the prediction and analysis of construction costs is required. The neural network technique eliminates the need of finding a good cost estimating relationship, and eventually produces more accurate results than the existing parameter-based estimating methods.

The contribution of this thesis to the body of knowledge of construction engineering and management is present in the fact that it examined the implementation of a new paradigm of computation to construction cost estimating problems. In these times, where artificial intelligence applications are gaining more appreciation in the construction industry, the neural network computing technology, which is a new form of artificial intelligence, presents itself, through this thesis, as a viable alternative that should be given serious attention in the field of construction cost estimating. The neural network computing technology presented a relatively accurate and reliable method in analyzing cost data and making predictions. This technology holds great promise for applications in cost estimating and should be carefully considered when analysis and predictions are required.

6.4 THE DARK SIDE OF NEURAL NETWORKS

The process of developing and implementing neural networks to parametric cost estimating applications has a number of problems and drawbacks associated with it. First, designing the network architecture and setting its parameters is not a straight forward approach, but on the contrary, it's a trial and error approach. A good amount of time must be spent to determine the best network architecture and the network parameters that best fit the application under consideration. Second, the learning algorithm, such as back propagation, may be suitable for some applications but not for many others. There is no explicit set of rules that determine whether a given learning algorithm is suitable for a particular application or not. In addition, applying the neural network approach to every cost estimating application without examining before hand if the application itself fits into a pattern association framework, or in our case, a parametric cost model, may lead to very discouraging results. The major drawback of applying the neural network technique to cost estimating applications is that the risk factor associated with preparing an estimate is higher for the neural net approach than with other traditional estimating techniques. Neural networks perform very poorly on new cases that do not belong entirely to the input space of the cases on which the network was trained. A fuzzy logic analysis on new cases may be required to determine how good these cases correlate with the set of old cases that were presented to the network in the learning phase. In addition, it is necessary to know what is the amount of risk involved in presenting a new case which has different features than those of the cases of the learning set to the network. Sometimes, this risk factor added to that present inherently in the estimating process tend to make the neural network approach less reliable for preparing cost estimates.

6.5 SUGGESTIONS FOR FUTURE WORK AND FURTHER RESEARCH OPPORTUNITIES

Future research opportunities, refinements, improvements, and extensions of this work are tremendous. The first opportunity that should be investigated is the refinement of the proposed neural network approach of this thesis by developing a more robust programming shell capable of executing quick and efficient network training and testing. In addition, the shell could be improved to include automatic adjustments for the network architectures, the number of cycles, and training parameters throughout the learning phase. Other opportunities are present in the wide variety of cost estimating applications that are encountered in the construction industry and the investigation of developing neural networks for these applications.

Every cost estimating application presents a challenge by itself in terms of producing the most accurate estimate for that particular application. The neural network technique presents itself as a strong candidate to be used as an effective tool capable of producing, from preliminary information, relatively accurate estimates, inexpensively, and in a short period of time. Like any new paradigm of computation and thinking, the neural network methodology deserves real attention from practitioners and researchers in the construction industry and their commitment of investigating further applications and real testing of this new technology in the field of cost estimating.

APPENDIX A: BIBLIOGRAPHY

[1] Alexander, I., *Neural Computing Architectures* , MIT Press, 1989.

[2] Ansaklis, P.J., "Neural Networks for Control Systems," *IEEE Transactions on neural Networks,* vol.1 , no.1, Jan. 1990, pp 93-100.

[3] Bielfel, J.R., et al., " Cost Scaling Factors: How Accurate Are They ?," *Cost Engineering,* Octobre, 1992.

[4] Blanchard, B.S.et al., *Systems Engineering and Analysis,* Prentice Hall, Inc., NJ, 1990.

[5] Caudill, M., "What is a Neural Network," *AI Expert, Neural Network Special Report,*1992.

[6] Clerck, J., "Multiplying Factors Give Installed Costs of Process Equipment," *Chemical Engineering*, 182, February 18, 1963.

[7] Cran, J., "Improved Factor Method Gives Better Preliminary Capital Cost Estimates," *Chemical Engineering,* April (65); 1981.

[8] Dayhoff, J. E., *Neural Networks Architectures*, Van Nostrand Reinhold, New York, 1990.

[9] Fabrycky, W.J. et al., *Life-Cycle Cost and Economic Analysis,* Prentice Hall, Inc., NJ, 1991.

[10] Freeman, J.A., "Neural Networks in Mathematica ," *AI Expert,* November 1992.

[11] Freeman, J.A., "Back Propagation in a Neural Network ," *AI Expert, Neural Network Special Report,* 1992.

[12] Garrett, J.H., Jr. et al., "Neural Networks," *Expert Systems for Civil Engineers: Knowledge Representation,* ASCE, 1992. pp 104-143.

[13] Grossberg, S., The Adaptive Brain, Volumes I & II, North-Holland, New York, 1987.

[14] Guthrie, K.M., "Data and Techniques for Preliminary Capital Cost Estimating," *Chemical Engineering*, 114, March 24, 1969.

[15] Hand, W. E., "From Flow Sheet to Cost Estimate," Petroleum Refiner, 331, September, 1958.

[16] IEEE, *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, CA, June 1987.

[17] IEEE, *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, CA, July 1988.

[18] Jelen C. F., and Black J. H., *Cost and Optimization Engineering*, McGraw-Hill, Inc., 1983.

[19] Kohonen, T., *Self-Organization and Associative Memory*, Series in information sciences, Vol. 8, Berlin: Springer Verlag.

[20] Lang, H. J., "A Simplified Approach To Preliminary Cost Estimates," *Chemical Engineering*, 112, June, 1948.

[21] Lawrence, J., "Data Preparation For a Neural Network," *AI Expert, Neural Network Special Report, 1992.*

[22] Lehay, M.B. et al., "Neural Network Payload Estimation for Adaptive Robot Control," *IEEE Transactions on neural Networks,* vol.2 , no.1, Jan. 1991, pp 93-100.

[23] McKim, R.A., "Neural Network Applications For Project Management: Three Case Studies," *Project Management Journal,* No. 4, volume 24, Dec. 1993.

[24] Oglesby, C.H. et al, *Productivity Improvement in Construction,* McGraw-Hill, New York, 1989.

[25] Ostwald, P.F., *Cost Estimating For Engineering and Management,* Prentice Hall, 1980.

[26] Peters et al., *Plant Design and Economics for Chemical Engineers,* McGraw-Hill, New York, 1980.

[27] Phadke, P. S., et al., "Estimating the Costs and Weighs of Process Vessels," *Chemical Engineering*, 84(21): 1977.

[28] Pikulik, A. et al., "Cost Estimating for Major Process Equipment," *Chemical Engineering*, 84(21), 1977.

[29] Ritz, G.J., *Total Engineering Project Management,* McGraw-Hill, New York, 1986.

[30] Rosenblatt, E., Principles of Neurodynamics, Spartan Books, New York, 1962.

[31] Rumelhart, D.E. et al, *Parallel Distributed Processing- Volume 1: Foundations,* The MIT Press, Cambridge , Mass, 1986.

[32] Sigurdsen, A. , "CERA: An Integrated Cost Estimating Program," *Cost Engineering,* vol.34, no.6, June 1992 .

[33] Sinha, V, "Estimating Capital Cost of a Plant from the Equipment List: A Case Study," *Engineering Costs and Production Economics,* 14(1988).

[34] Taylor, J. M., "The *Process Step Scoring* Method for Making Quick Capital Estimates," *Engineering Costs and Production Economics*, 259, February, 1977.

[35] Theusen, G.J. et al., *Engineering Economy,* Prentice Hall, Inc., NJ, 1989.

[36] Viola, J. L., "Estimating Capital Costs Via a New, Shortcut Method," *Chemical Engineering*, 80, April 6, 1969.

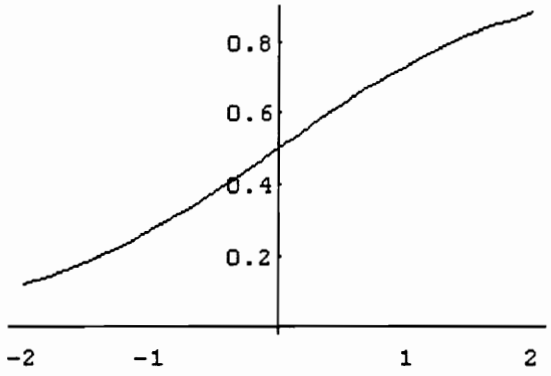
[37] Wasserman, P.D., *Neural computing: Theory and Practice,* Van Nostrand Reinhold, 1989.

[38] Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer,* Reading, Mass.: Addison Wesley, 1991.

APPENDIX B - SOFTWARE INFORMATION

A.Transfer function

```
Sigmoid[x_]:= 1/(1+E^(-x));
Plot[Sigmoid[x],{x,-2,2}]
```



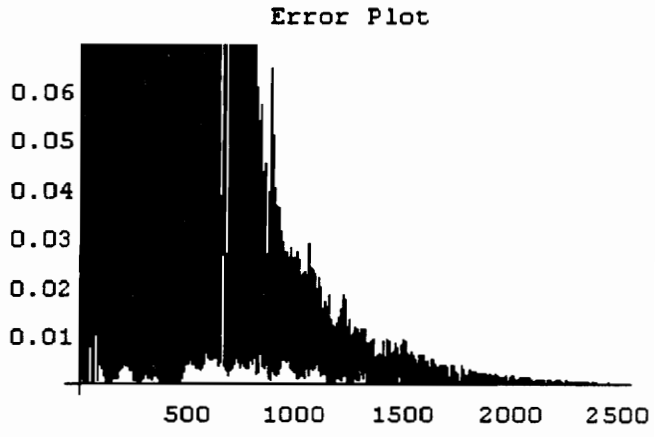
B- Learning function

```
Neurontum[inN_,hidN_,outN_,ioPairs_,eta_,alpha_,
cycles_]:= Module[{errorlist,hidwts,outwts,ioP,inputs,
        oexp,hidouts,outputs,outerrors,outlastdelta,
        hidlastdelta,outdelta,hiddelta},
hidwts= Table[Table[Random[Real,{-2,2}],{inN}],{hidN}];
outwts= Table[Table[Random[Real,{-2,2}],{hidN}],{outN}];
hidlastdelta= Table[Table[0,{inN}],{hidN}];
outlastdelta= Table[Table[0,{hidN}],{outN}];
errorlist= Table[
ioP= ioPairs[[Random[Integer,{1,Length[ioPairs]}]]];
inputs=ioP[[1]];
oexp=ioP[[2]];
```

```
hidouts = Sigmoid[hidwts.inputs]; (*hidden layer outs*)
outputs = Sigmoid[outwts.hidouts];(*output layer outs*)
outerrors = oexp-outputs;
outdelta= outerrors(outputs(1-outputs));
hiddelta= (hidouts(1-hidouts)) Transpose[outwts].outdelta;
outlastdelta = eta Outer[Times,outdelta,hidouts]+
        alpha outlastdelta;
outwts += outlastdelta;
hidlastdelta = eta Outer[Times,hiddelta,inputs]+
        alpha hidlastdelta;
hidwts += hidlastdelta;
outerrors.outerrors,{cycles}];
Return[{hidwts,outwts,errorlist}];];
```

C- Error plot

```
ListPlot[outs[[3]],PlotJoined->True, PlotLabel->Error Plot];
```



D- Test Function

```
Neurotest[hidwts_,outwts_,Pairs_] := Module[{inputs,hidden,outputs,desired,deviation,
    i,len,errorsum},
      inputs= Map[First,Pairs];
      desired= Map[Last,Pairs];
      hidden= Sigmoid[inputs.Transpose[hidwts]];
      outputs= Sigmoid[hidden.Transpose[outwts]];
      deviation= desired-outputs;
      errorsum=0;
      len= Length[inputs];
      For [i=1,i<=len,i++,
            Print["input",i,"=",inputs[[i]]];
            Print["output",i,"=",outputs[[i]],
            "desired=",desired[[i]],"Error=",
            deviation[[i]]];Print[];errorsum=errorsum+deviation[[i]].
            deviation[[i]];];
     Print["Mean Squared error=",errorsum/len];];
```

APPENDIX C- MATHEMATICA LISTINGS

C.1 Preliminary work

C.1.1 Pipe example

```
ioPairs=
{{{0.8,0.7,0.46875},{0.664615}},    (* job2 *)
{{0.08,0.7,0.78125},{0.029231}},    (* job4 *)
{{0.32,0.8,0.46875},{0.18}},        (* job6 *)
{{0.72,0.4,0.625},{0.401538}},      (* job8 *)
{{0.96,0.6,0.3125},{0.772308}},     (* job10 *)
{{0.64,0.6,0.625},{0.436923}},      (* job11 *)
{{0.24,0.6,0.46875},{0.1}},         (* job13 *)
{{0.96,0.4,0.9375},{0.650769}},     (* job14 *)
{{0.48,0.2,0.9375},{0.166154}},     (* job15 *)
{{0.8,0.4,0.3125},{0.444165}}};     (* job16 *)

Allpairs=
{{{0.8,0.7,0.78125},{0.709231}},    (* job1 *)
{{0.8,0.7,0.46875},{0.664615}},     (* job2 *)
{{0.8,0.7,0.3125},{0.647692}},      (* job3 *)
{{0.08,0.7,0.78125},{0.029231}},    (* job4 *)
{{0.48,0.6,0.3125},{0.258462}},     (* job5 *)
{{0.32,0.8,0.46875},{0.18}},        (* job6 *)
{{0.64,0.6,0.3125},{0.404615}},     (* job7 *)
{{0.72,0.4,0.625},{0.401538}},      (* job8 *)
{{0.16,0.2,0.9375},{0.038462}},     (* job9 *)
{{0.96,0.6,0.3125},{0.772308}},     (* job10 *)
{{0.64,0.6,0.625},{0.436923}},      (* job11 *)
{{0.8,0.6,0.78125},{0.635385}},     (* job12 *)
{{0.24,0.6,0.46875},{0.1}},         (* job13 *)
{{0.96,0.4,0.9375},{0.650769}},     (* job14 *)
{{0.48,0.2,0.9375},{0.166154}},     (* job15 *)
{{0.8,0.4,0.3125},{0.444165}}};     (* job16 *)
```

1. 20,000 cycles run

```
outs= Neurontum[3,4,1,ioPairs,0.15,0.9,20000];

outs[[1]]
{{-0.441239, -1.71508, -0.374553},
 {-2.41356, 0.685686, 0.659614},
 {0.711298, -3.08072, -1.08653},
 {1.26085, 0.703921, 0.315697}}

outs[[2]]
{{-5.32036, -6.92006, -3.241, 4.20628}}

Neurotest[hidwts_,outwts_,Pairs_] := Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum},
        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= desired-outputs;
        errorsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]],
                "desired=",desired[[i]],"Error=",
                deviation[[i]]];Print[];errorsum=errorsum+deviation[[i]].
                deviation[[i]];];
        Print["Mean Squared error=",errorsum/len];];
```

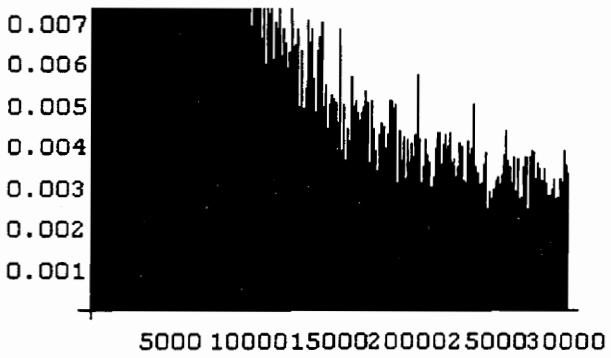
Neurotest[outs[[1]],outs[[2]],Allpairs]

input1={0.8, 0.7, 0.78125}
output1={0.656204}desired={0.709231}Error={0.0530273}
input2={0.8, 0.7, 0.46875}
output2={0.667554}desired={0.664615}Error= {-0.00293856}
input3={0.8, 0.7, 0.3125}
output3={0.667745}desired={0.647692}Error={-0.0200535}
input4={0.08, 0.7, 0.78125}
output4={0.0497648}desired={0.029231}Error= {-0.0205338}
input5={0.48, 0.6, 0.3125}
output5={0.289023}desired={0.258462}Error={-0.0305606}
input6={0.32, 0.8, 0.46875}
output6={0.187634}desired={0.18}Error={-0.0076337}
input7={0.64, 0.6, 0.3125}
output7={0.461988}desired={0.404615}Error={-0.0573734}
input8={0.72, 0.4, 0.625}
output8={0.434245}desired={0.401538}Error={-0.0327068}
input9={0.16, 0.2, 0.9375}
output9={0.0263336}desired={0.038462}Error={0.0121284}
input10={0.96, 0.6, 0.3125}
output10={0.730859}desired={0.772308}Error={0.0414494}
input11={0.64, 0.6, 0.625}
output11={0.457704}desired={0.436923}Error= {-0.0207809}
input12={0.8, 0.6, 0.78125}
output12={0.623476}desired={0.635385}Error={0.0119089}
input13={0.24, 0.6, 0.46875}
output13={0.100257}desired={0.1}Error={-0.000256969}
input14={0.96, 0.4, 0.9375}
output14={0.66817}desired={0.650769}Error={-0.0174006}
input15={0.48, 0.2, 0.9375}
output15={0.131033}desired={0.166154}Error={0.0351208}
input16={0.8, 0.4, 0.3125}
output16={0.477256}desired={0.444165}Error= {-0.0330909}

Mean Squared error = 0.000879257

2. 30,000 Cycles run

outs= Neurontum[3,4,1,ioPairs,0.13,0.9,30000];




```
outs[[1]]
{{0.12277, 1.98546, 0.0556459},
 {-1.52392, -0.235504, 0.0664981},
{0.117964, -2.56629, -0.900585},
 {-2.30239, 0.77575, 0.255704}}

outs[[2]]
{{4.57118, -3.89393, -4.93178, -7.15899}}

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
       i,len,errorsum},
         inputs= Map[First,Pairs];
         desired= Map[Last,Pairs];
         hidden= Sigmoid[inputs.Transpose[hidwts]];
         outputs= Sigmoid[hidden.Transpose[outwts]];
         deviation= desired-outputs;

         errorsum=0;
```

```
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]],
                "desired=",desired[[i]],"Error=",
                deviation[[i]]];Print[];
                errorsum=errorsum+deviation[[i]].
                deviation[[i]];];

        Print["Mean Squared error=",errorsum/len];];


Neurotest[outs[[1]],outs[[2]],Allpairs]

input1={0.8, 0.7, 0.78125}
output1={0.678798}desired={0.709231}Error={0.0304329}
input2={0.8, 0.7, 0.46875}
output2={0.676221}desired={0.664615}Error={-0.0116061}
input3={0.8, 0.7, 0.3125}
output3={0.672193}desired={0.647692}Error={-0.0245014}
input4={0.08, 0.7, 0.78125}
output4={0.0495916}desired={0.029231}Error= {-0.0203606}
input5={0.48, 0.6, 0.3125}
output5={0.284673}desired={0.258462}Error={-0.0262112}
input6={0.32, 0.8, 0.46875}
output6={0.1846}desired={0.18}Error={-0.00460043}
input7={0.64, 0.6, 0.3125}
output7={0.461551}desired={0.404615}Error={-0.0569364}
input8={0.72, 0.4, 0.625}
output8={0.440334}desired={0.401538}Error={-0.0387961}
input9={0.16, 0.2, 0.9375}
output9={0.0275201}desired={0.038462}Error={0.0109419}
input10={0.96, 0.6, 0.3125}
output10={0.74147}desired={0.772308}Error={0.030838}
```

input11={0.64, 0.6, 0.625}

output11={0.471217}desired={0.436923}Error= {-0.0342943}

input12={0.8, 0.6, 0.78125}

output12={0.642657}desired={0.635385}Error= {-0.00727165}

input13={0.24, 0.6, 0.46875}

output13={0.0994975}desired={0.1}Error={0.000502545}

input14={0.96, 0.4, 0.9375}

output14={0.674166}desired={0.650769}Error= {-0.0233974}

input15={0.48, 0.2, 0.9375}

output15={0.137227}desired={0.166154}Error={0.0289266}

input16={0.8, 0.4, 0.3125}

output16={0.483047}desired={0.444165}Error= {-0.0388821}

Mean Squared error = 0.000795422

3. 30,000 + 1000 cycles run

hidwts={{0.103505, 1.97702, 0.0489214},

{-1.51061, -0.228504, 0.0730713},

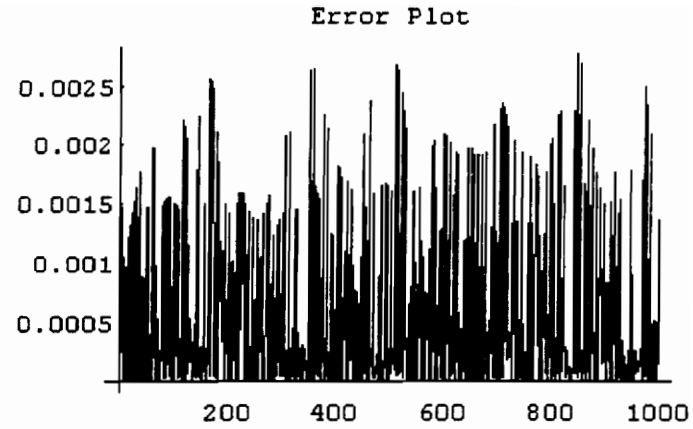
{0.135058, -2.56206, -0.898953},

{-2.27967, 0.786807, 0.265158}} ;

outwts={{4.56393, -3.90021, -4.93953, -7.1646}} ;

outs={0,0,0};

outs= Neurontum[3,4,1,ioPairs,0.1,0.99,1000];




```
outs[[1]]
{{0.10612, 1.97889, 0.0500647},

 {-1.5126, -0.22995, 0.0721745},

 {0.133014, -2.56356, -0.899847},

 {-2.28329, 0.784164, 0.263489}}

outs[[2]]
{{4.56715, -3.89955, -4.9391, -7.16391}}

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum},

        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= (desired-outputs)*65;
        errorsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
              Print["input",i,"=",inputs[[i]]];
```

```
            Print["output",i,"=",outputs[[i]]*65,
            "desired=",desired[[i]]*65,"Error=",
            deviation[[i]]];Print[];
            errorsum=errorsum+deviation[[i]].
            deviation[[i]];];

      Print["Mean Squared error=",errorsum/len];];

Neurotest[outs[[1]],outs[[2]],Allpairs]
```

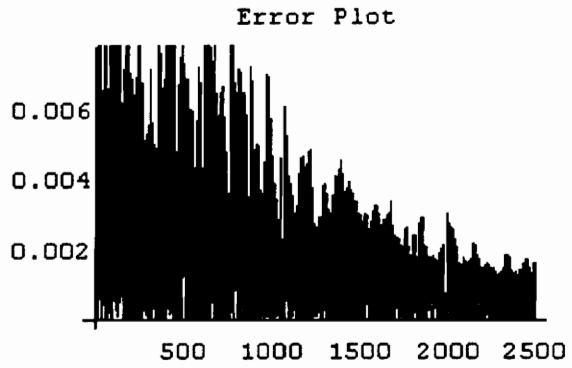
input1={0.8, 0.7, 0.78125}
output1={43.0467}desired={46.1}Error={3.05329}
input2={0.8, 0.7, 0.46875}
output2={42.9563}desired={43.2}Error={0.243633}
input3={0.8, 0.7, 0.3125}
output3={42.7267}desired={42.1}Error={-0.62671}
input4={0.08, 0.7, 0.78125}
output4={3.07593}desired={1.90001}Error={-1.17591}
input5={0.48, 0.6, 0.3125}
output5={17.7154}desired={16.8}Error={-0.915327}
input6={0.32, 0.8, 0.46875}
output6={11.4399}desired={11.7}Error={0.260147}
input7={0.64, 0.6, 0.3125}
output7={28.9565}desired={26.3}Error={-2.65652}
input8={0.72, 0.4, 0.625}
output8={27.488}desired={26.1}Error={-1.38807}
input9={0.16, 0.2, 0.9375}
output9={1.70152}desired={2.50003}Error={0.79851}
input10={0.96, 0.6, 0.3125}
output10={47.3487}desired={50.2}Error={2.85136}
input11={0.64, 0.6, 0.625}
output11={29.4856}desired={28.4}Error={-1.08562}
input12={0.8, 0.6, 0.78125}
output12={40.6461}desired={41.3}Error={0.653881}

input13={0.24, 0.6, 0.46875}
output13={6.17093}desired={6.5}Error={0.329068}
input14={0.96, 0.4, 0.9375}
output14={42.7103}desired={42.3}Error={-0.410354}
input15={0.48, 0.2, 0.9375}
output15={8.4002}desired={10.8}Error={2.39981}
input16={0.8, 0.4, 0.3125}
output16={30.3123}desired={28.8707}Error={-1.44157}
Mean Squared error=2.47092

C.1.2 Bridge example

ioPairs=
{{{0.76,0.5,0.411,0.5,0.5},{0.31489}},
{{0.68,0.5,0.824,0.5,0.8},{0.537450}},
{{0.68,0.33333,0.5311,0.75,0.48},{0.502873}}};
Tspairs =
{{0.68,0.33333,0.4684,0.75,0.42},{0.440721}};

outs= Neurontum[5,6,1,ioPairs,0.3,0.9,2500];



outs[[1]]
{{-2.08208, 1.59417, 0.485469, 0.490886, 0.217322},

{1.95609, 0.435613, -1.29355, -1.2627, -0.32351},

{-1.6995, 0.170013, 0.297572, 1.48, 0.997315},

{0.415211, -1.06923, 0.136103, -0.0627828,

0.250554}, {-1.43877, 0.591343, 0.315036, -1.41007,

-1.80571}, {-0.105552, -2.02725, 0.745988,

0.317182, 0.960698}};

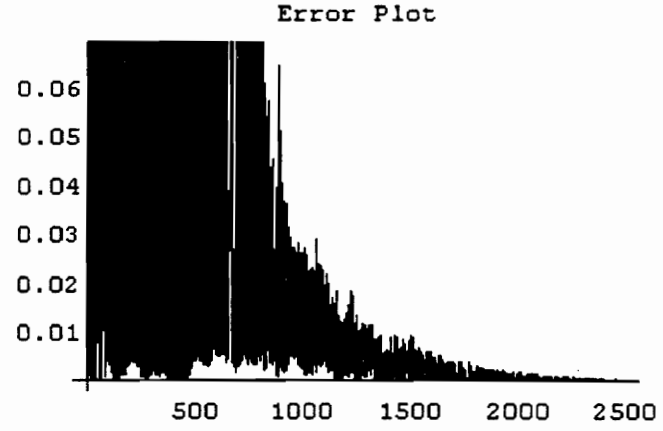
outs [[2]] ={{0.759164, -2.81027, 0.462779, -0.428147, -0.297864, 1.24587}};

```
Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum,PD,PDsum},

        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= (desired-outputs)*1000000;
        PD= Abs[deviation]*100/(desired*1000000);

        errorsum=0;
        PDsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
              Print["input",i,"=",inputs[[i]]];
              Print["output",i,"=",outputs[[i]]*1000000,
              "desired=",desired[[i]]*1000000];
              Print["Error=",deviation[[i]]," % Deviation =",PD[[i]]];
              Print["                    "];
              errorsum=errorsum+Abs[deviation[[i]]];
              PDsum=PDsum+PD[[i]];];

       Print["Mean Squared error=",errorsum/len];
       Print["Mean percent deviation=", PDsum/len];];



Neurotest1[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,Delta,PD
      },
```

```
        inputs= Pairs[[1]];
        desired= Pairs[[2]];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        Delta= (desired-outputs)*1000000;
        PD = Abs[Delta]/(desired*10000);
                Print["input","=",inputs];
                Print["output","=",outputs*1000000,
                "desired=",desired*1000000];
                Print["Error=",Delta,"%deviation=",PD];
                ];

Neurotest[outs[[1]],outs[[2]],ioPairs]
input1={0.76, 0.5, 0.411, 0.5, 0.5}
output1={322724.}desired={314890.}
Error={-7834.33} % Deviation ={2.48796}

input2={0.68, 0.5, 0.824, 0.5, 0.8}
output2={532473.}desired={537450.}
Error={4976.76} % Deviation ={0.925995}

input3={0.68, 0.33333, 0.5311, 0.75, 0.48}
output3={488625.}desired={502873.}
Error={14247.7} % Deviation ={2.83326}

Mean error={9019.6}
Mean percent deviation={2.08241}

Neurotest1[outs[[1]],outs[[2]],Tspairs]
input={0.68, 0.33333, 0.4684, 0.75, 0.42}
output={459456.}desired={440721.}
Error={-18735.4}%deviation={4.25109}
```

C.2 CASE STUDY I

```
ioPairs={
{{0.411667,0.38,0.814286,0.663,0.732,0.622},      (*P1*)
{0.32416}},
{{0.502333,0.158,0.442857,0.294,0.46,0.303},      (*P2*)
{0.17486}},
{{0.884667,0.838,0.585714,0.909,0.896,0.4761},    (*P4*)
{0.774074}},
{{0.58,0.1096,0.471429,0.319,0.532,0.833},        (*P3*)
{0.449955}}};

Tspairs ={{0.350333,0.346,0.385714,0.302,0.388,0.217}, (*P5*)
{0.251668}};

outs={0,0,0};
outs= Neurontum[6,3,1,ioPairs,0.15,0.9,2500];
```




```
outs[[1]]
{-0.342282, -2.31226, 2.78473, -0.954718, 0.367379,

 -1.85039}, {0.475612, 0.661695, -0.237559,

 0.727532, 0.0660697, 0.866314},
```

```
 {-0.267698, -0.556785, -0.123483, -0.654603,

  -0.353348, -0.592483}};
outs[[2]]
{{-5.29296, 2.28468, -1.33046}};

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum,PD,PDsum},

        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= -(desired-outputs)*50000000;
        PD= Abs[deviation]*100/(desired*50000000-5000000);

        errorsum=0;
        PDsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]]*50000000-5000000,
                "desired=",desired[[i]]*50000000-5000000];
                Print["Error=",deviation[[i]]," % Deviation =",PD[[i]]];
                Print["                    "];
                errorsum=errorsum+Abs[deviation[[i]]];
                PDsum=PDsum+PD[[i]];];

        Print["Mean error=",errorsum/len];
        Print["Mean percent deviation=", PDsum/len];];
```

```
Neurotest1[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,Delta,PD
       },

         inputs= Tspairs[[1]];
         desired= Tspairs[[2]];
         hidden= Sigmoid[inputs.Transpose[hidwts]];
         outputs= Sigmoid[hidden.Transpose[outwts]];
         Delta= -(desired-outputs)*50000;
         PD = Abs[Delta]/(desired*500-50);
                  Print["input","=",inputs];
                  Print["output","=",outputs*50000000-5000000,
                  "desired=",desired*50000000-5000000];
                  Print["Error=",Delta,"%deviation=",PD];
                  ];

Neurotest[outs[[1]],outs[[2]],ioPairs]
input1={0.411667, 0.38, 0.814286, 0.663, 0.732, 0.622}
                7                7
output1={1.11835 10 }desired={1.1208 10 }
Error={-24512.2} % Deviation ={0.218703}

input2={0.502333, 0.158, 0.442857, 0.294, 0.46, 0.303}
                6                6
output2={3.84392 10 }desired={3.743 10 }
Error={100916.} % Deviation ={2.69612}

input3={0.884667, 0.838, 0.585714, 0.909, 0.896,

 0.4761}
                7                7
output3={3.36643 10 }desired={3.37037 10 }
Error={-39376.2} % Deviation ={0.11683}
```

input4={0.58, 0.1096, 0.471429, 0.319, 0.532, 0.833}

7 7

output4={1.74705 10 }desired={1.74978 10 }

Error={-27293.1} % Deviation ={0.155981}

Mean error={48024.3}

Mean percent deviation={0.796908}

Neurotest1[outs[[1]],outs[[2]],Tspairs]

input={0.350333, 0.346, 0.385714, 0.302, 0.388, 0.217}

6 6

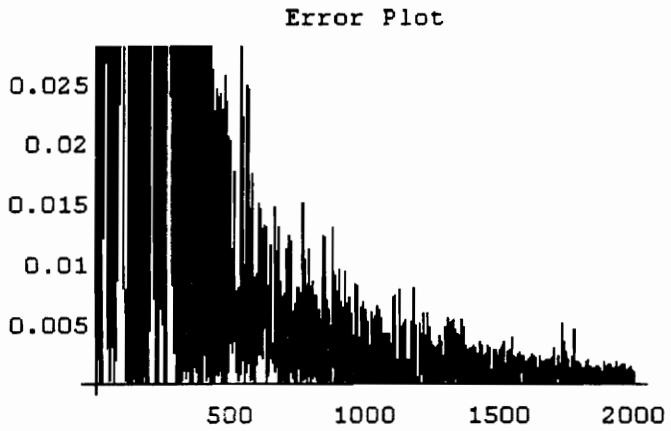
output={8.44034 10 }desired={7.5834 10 }

Error={856.938}%deviation={11.3002}

C.3 CASE STUDY II

```
ioPairs={
{{0.411667,0.38,0.814286,0.663,0.732,0.622},      (*P1*)
{0.32416}},
{{0.502333,0.158,0.442857,0.294,0.46,0.303},      (*P2*)
{0.17486}},
{{0.884667,0.838,0.585714,0.909,0.896,0.4761},    (*P4*)
{0.774074}},
{{0.58,0.1096,0.471429,0.319,0.532,0.833},        (*P3*)
{0.449955}},
{{0.350333,0.346,0.385714,0.302,0.388,0.217},     (*P5*)
{0.251668}}};

outs={0,0,0};
outs[[1]]={{-0.342282, -2.31226, 2.78473, -0.954718, 0.367379,

 -1.85039}, {0.475612, 0.661695, -0.237559,

 0.727532, 0.0660697, 0.866314},

{-0.267698, -0.556785, -0.123483, -0.654603,

 -0.353348, -0.592483}};
outs[[2]]={{-5.29296, 2.28468, -1.33046}};

Neuronested[hidwts_,outwts_,Pairs_] :=
Module[{inputs,hidden,outputs},
        inputs= Map[First,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];

Return[outputs];
];
```

```
EquipOut = Neuronested[outs[[1]],outs[[2]],ioPairs];

ioPairstwo={
{{0.411666667,0.38,EquipOut[[1,1]]},{0.382952568}},    (*P1*)
{{0.502333333,0.157976,EquipOut[[2,1]]},{0.138625}},   (*P2*)
{{0.58,0.109512,EquipOut[[3,1]]},{0.416630916}},        (*P3*)
{{0.884666667,0.838116,EquipOut[[4,1]]},{0.89603612}}      (*P4*)
};


Tspairstwo ={
{0.350333333,0.346,EquipOut[[5,1]]},{0.36184324}};     (*P5*)

outstwo={0,0,0};

outstwo= Neurontum[3,4,1,ioPairstwo,0.3,0.9,2000];
```




```
outstwo[[1]]={{0.0948155, -1.98772, -1.11391},

 {0.4741, 1.06357, 0.47164},

 {-0.56876, -1.33477, -0.886306},
 {0.614657, 1.45424, 0.467622}};
```

```
outstwo[[2]]={{-6.46473, 1.08067, -4.88583, 2.57272}};

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum,PD,PDsum},

        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= (desired-outputs)*200000000;
        PD= Abs[deviation]*100/(desired*200000000);

        errorsum=0;
        PDsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]]*200000000,
                "desired=",desired[[i]]*200000000];
                Print["Error=",deviation[[i]]," % Deviation =",PD[[i]]];
                Print["                    "];
                errorsum=errorsum+Abs[deviation[[i]]];
                PDsum=PDsum+PD[[i]];];

        Print["Mean error=",errorsum/len];
        Print["Mean percent deviation=", PDsum/len];];


Neurotest1[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,Delta,PD
      },
```

```
inputs= Tspairstwo[[1]];
desired= Tspairstwo[[2]];
hidden= Sigmoid[inputs.Transpose[hidwts]];
outputs= Sigmoid[hidden.Transpose[outwts]];
Delta= -(desired-outputs)*200000000;
PD = Abs[Delta]/(desired*2000000);


        Print["input","=",inputs];
        Print["output","=",outputs*200000000,
        "desired=",desired*200000000];
        Print["Error=",Delta,"%deviation=",PD];
        ];


Neurotest[outstwo[[1]],outstwo[[2]],ioPairstwo]
input1={0.411666667, 0.38, 0.32367}
          7              7
output1={8.33317 10 }desired={7.65905 10 }
         6
Error={-6.74115 10 } % Deviation ={8.80155}

input2={0.502333333, 0.157976, 0.176878}
          7              7
output2={2.7765 10 }desired={2.7725 10 }
Error={-40033.3} % Deviation ={0.144394}

input3={0.58, 0.109512, 0.773286}
          7              7
output3={8.41067 10 }desired={8.33262 10 }
Error={-780516.} % Deviation ={0.9367}

input4={0.884666667, 0.838116, 0.449409}
          8              8
```

output4={1.72998 10 }desired={1.79207 10 }
6
Error={6.20876 10 } % Deviation ={3.46457}

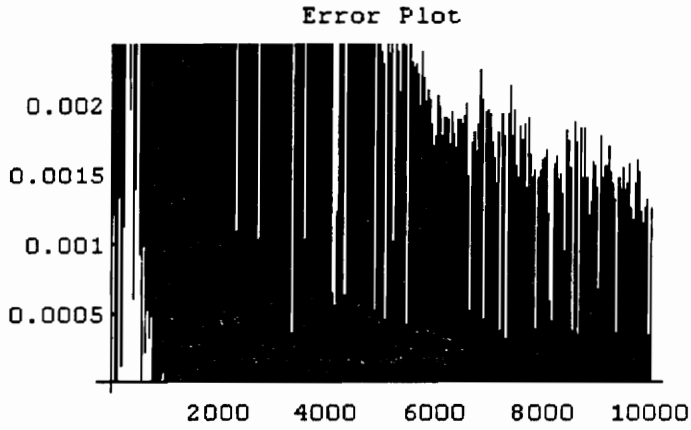
6
Mean error={3.44262 10 }
Mean percent deviation={3.3368}

Neurotest1[outstwo[[1]],outstwo[[2]],Tspairstwo]
input={0.350333333, 0.346, 0.268807}
7 7
output={6.64152 10 }desired={7.23686 10 }
6
Error={-5.9534 10 }%deviation={8.22649}

C.4 CASE STUDY III

```
Tspairs = {
{{0.022266,0.053,0.4},{0.225535}},
{{0.033222,0.078,0.266667},{0.160048}},
{{0.048066,0.15,0.4},{0.27054}},
{{0.441081,0.868,0.266667},{0.696715}},
{{0.012566,0.059,0.833333},{0.452578}},
{{0.012566,0.173,0.4},{0.28128}},
{{0.110447,0.2874,0.2},{0.257453}},
{{0.015708,0.169,0.266667},{0.195423}},
{{0.069685,0.175,0.266667},{0.222468}}
};


ioPairs={
{{0.646541,0.555,0.666667},{0.82723}},
{{0.342768,0.567,0.066667},{0.41686}},
{{0.085118,0.318,0.4},{0.36981}},
{{0.132536,0.199,0.5},{0.39266}},
{{0.159044,0.451,0.2},{0.370298}},
{{0.023562,0.077,0.266667},{0.157885}},
{{0.007069,0.034,0.266667},{0.141805}},
{{0.384885,0.405,0.4},{0.536888}},
{{0.009621,0.042,0.266667},{0.13705}},
{{0.023562,0.072.0.266667},{0.160965}},
{{0.010681,0.042,0.2},{0.11341}},
{{0.004172,0.08,0.4},{0.225713}},
{{0.178129,0.261,0.4},{0.37277}},
{{0.04948,0.68,0.4},{0.580083}},
{{0.004663,0.031,0.266667},{0.123138}},
{{0.282744,0.345,0.266667},{0.390028}},
{{0.009621,0.0306,0.2},{0.127325}},
```

```
{{0.012566,0.06,0.266667},{0.139853}}
};

outs={0,0,0};

outs= Neurontum1[3,4,1,ioPairs,0.2,0.9,10000];
```




```
outs[[1]] ={{-0.796648, -1.28414, -1.09229},

 {0.569639, 0.479947, 1.41274},

 {-0.844284, -1.22041, -1.17892},

 {-0.826069, -1.4214, -1.09667}};
outs [[2]]={{-2.49122, 2.82731, -2.8042, -3.27715}};

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
       i,len,errorsum,PD,PDsum},

         inputs= Map[First,Pairs];
         desired= Map[Last,Pairs];
         hidden= Sigmoid[inputs.Transpose[hidwts]];
         outputs= Sigmoid[hidden.Transpose[outwts]];
         deviation= (desired-outputs)*400000;
         PD= Abs[deviation]*100/(desired*400000-25000);
```

```
        errorsum=0;
        PDsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]]*400000-25000,
                "desired=",desired[[i]]*400000-25000];
                Print["Error=",deviation[[i]]," % Deviation =",PD[[i]]];
                Print["                    "];
                errorsum=errorsum+Abs[deviation[[i]]];
                PDsum=PDsum+PD[[i]];];

        Print["Mean  error=",errorsum/len];
        Print["Mean percent deviation=", PDsum/len];];

Neurotest[outs[[1]],outs[[2]],ioPairs]

input1={0.646541, 0.555, 0.666667}
output1={291358.}desired={305892.}
Error={14533.8} % Deviation ={4.7513}

input2={0.342768, 0.567, 0.066667}
output2={141682.}desired={141744.}
Error={61.7479} % Deviation ={0.043563}

input3={0.085118, 0.318, 0.4}
output3={127670.}desired={122924.}
Error={-4746.2} % Deviation ={3.86109}

input4={0.132536, 0.199, 0.5}
output4={133136.}desired={132064.}
Error={-1072.06} % Deviation ={0.811777}
```

input5={0.159044, 0.451, 0.2}
output5={119083.}desired={123119.}
Error={4036.09} % Deviation ={3.2782}

input6={0.023562, 0.077, 0.266667}
output6={37408.}desired={38154.}
Error={746.} % Deviation ={1.95523}

input7={0.007069, 0.034, 0.266667}
output7={29189.2}desired={31722.}
Error={2532.84} % Deviation ={7.9845}

input8={0.384885, 0.405, 0.4}
output8={196611.}desired={189755.}
Error={-6856.06} % Deviation ={3.61311}

input9={0.009621, 0.042, 0.266667}
output9={30601.4}desired={29820.}
Error={-781.356} % Deviation ={2.62024}

input10={0.023562, 0.072, 0.266667}
output10={36615.4}desired={39386.}
Error={2770.57} % Deviation ={7.03441}

input11={0.010681, 0.042, 0.2}
output11={20944.9}desired={20364.}
Error={-580.873} % Deviation ={2.85245}

input12={0.004172, 0.08, 0.4}
output12={61203.3}desired={65285.2}
Error={4081.92} % Deviation ={6.25245}

input13={0.178129, 0.261, 0.4}
output13={129413.}desired={124108.}

Error={-5304.57} % Deviation ={4.27416}

input14={0.04948, 0.68, 0.4}
output14={205103.}desired={207033.}
Error={1929.85} % Deviation ={0.932146}

input15={0.004663, 0.031, 0.266667}
output15={28525.4}desired={24255.2}
Error={-4270.21} % Deviation ={17.6053}

input16={0.282744, 0.345, 0.266667}
output16={131883.}desired={131011.}
Error={-871.397} % Deviation ={0.665131}

input17={0.009621, 0.0306, 0.2}
output17={19439.7}desired={25930.}
Error={6490.33} % Deviation ={25.0302}

input18={0.012566, 0.06, 0.266667}
output18={33587.7}desired={30941.2}
Error={-2646.48} % Deviation ={8.55327}

Mean error={3572.91}
Mean percent deviation={5.67325}

Neurotest[outs[[1]],outs[[2]],Tspairs]

input1={0.022266, 0.053, 0.4}
output1={58345.9}desired={65214.}
Error={6868.07} % Deviation ={10.5316}

input2={0.033222, 0.078, 0.266667}
output2={38631.}desired={39019.2}
Error={388.182} % Deviation ={0.994848}

input3={0.048066, 0.15, 0.4}
output3={82208.7}desired={83216.}
Error={1007.27} % Deviation ={1.21043}

input4={0.441081, 0.868, 0.266667}
output4={257604.}desired={253686.}
Error={-3918.22} % Deviation ={1.54452}

input5={0.012566, 0.059, 0.833333}
output5={166239.}desired={156031.}
Error={-10207.5} % Deviation ={6.54196}

input6={0.012566, 0.173, 0.4}
output6={81905.6}desired={87512.}
Error={5606.4} % Deviation ={6.40643}

input7={0.110447, 0.2874, 0.2}
output7={74295.4}desired={77981.2}
Error={3685.78} % Deviation ={4.7265}

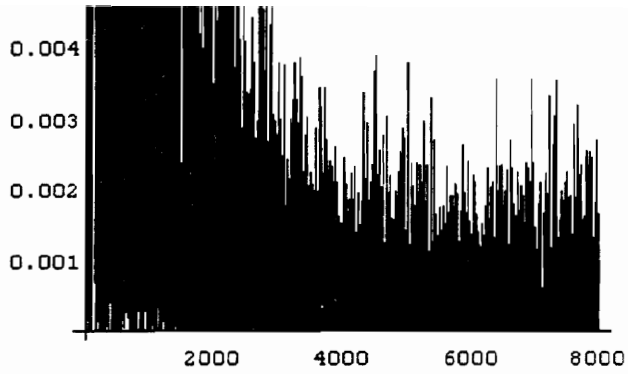
input8={0.015708, 0.169, 0.266667}
output8={52275.4}desired={53169.2}
Error={893.812} % Deviation ={1.68107}

input9={0.069685, 0.175, 0.266667}
output9={60450.6}desired={63987.2}
Error={3536.56} % Deviation ={5.52698}

Mean error={4012.42}
Mean percent deviation={4.35159}

C.5 CASE STUDY IV

```
ioPairs ={{{0.222142,0.20332,0.5,0.63,0.4,0.0854265,0.4},{0.2291277}},
{{0.4445,0.423332,0.5,0.606667,0.4,0.175968,0.4},{0.3103803}},
{{0.2363,0.417,0.25,0.38,0.4,0.100073,0.4},{0.1583977}},
{{0.379933,0.556,0.25,0.593333,0.4,0.291939,0.8},{0.2935377}},
{{0.9583,0.91268,0.5,0.606667,0.4,0.89315,0.4},{0.8909147}},
{{0.589867,0.632,0.75,0.42,0.8,0.35725,0.4},{0.376331}},
{{0.294,0.588,0.25,0.636667,0.8,0.188875,0.4},{0.211912}},
{{0.2905,0.2768,0.5,0.606667,0.8,0.15,0.4},{0.216325}},
{{0.819733,0.564,0.5,0.746667,0.4,0.4668,0.4},{0.589253}},
{{0.3416,0.488,0.25,0.676667,0.8,0.17475,0.4},{0.2882497}}
};

Tspairs ={{{0.215193,0.20332,0.5,0.601333,0.4,0.0810735,0.4},{0.2139037}},
{{0.440333,0.44,0.25,0.873333,0.8,0.24674,0.8},{0.3191883}},
{{0.2141,0.21408,0.5,0.58,0.4,0.0669,0.4},{0.2013393}},
{{0.444,0.444,0.5,0.58,0.8,0.20515,0.8},{0.2509123}},
{{0.2905,0.2768,0.5,0.606667,0.8,0.15,0.4},{0.2292723}},
{{0.5936,0.564,0.5,0.606667,0.4,0.37965,0.4},{0.5012577}},
{{0.653467,0.748,0.25,0.74,0.8,0.4763,0.4},{0.528449}}
};


outs={0,0,0};

outs= Neurontum[7,6,1,ioPairs,0.3,0.9,8000];
```



outs[[1]]

{{0.327668, -0.426516, 0.29873, 1.29571, 0.11342,

1.55933, -0.0870809},

{-0.555733, -0.336815, -0.393273, -0.43195,

-0.139424, -0.721752, -0.207788},

{-0.753077, -0.1738, -0.4109, -0.608726, 0.00505423,

-1.02603, 0.0258896},

{-0.762064, -0.139216, -0.398818, -0.739388,

0.0833498, -1.05328, 0.0672192},

{-0.685056, -0.219926, -0.351675, -0.615592,

0.054432, -0.956148, -0.127042},

{-0.253802, -0.156687, 0.843217, 1.34814, 1.57577,

-2.97151, 1.56144}}

```
outs[[2]]
{{3.94483, -0.580749, -1.32537, -1.44981, -1.04605,

 -3.26669}}

Neurotest[hidwts_,outwts_,Pairs_] :=
 Module[{inputs,hidden,outputs,desired,deviation,
      i,len,errorsum,PD,PDsum},

        inputs= Map[First,Pairs];
        desired= Map[Last,Pairs];
        hidden= Sigmoid[inputs.Transpose[hidwts]];
        outputs= Sigmoid[hidden.Transpose[outwts]];
        deviation= (desired-outputs)*3000000;
        PD= Abs[deviation]*100/(desired*3000000-125000);

        errorsum=0;
        PDsum=0;
        len= Length[inputs];
        For [i=1,i<=len,i++,
                Print["input",i,"=",inputs[[i]]];
                Print["output",i,"=",outputs[[i]]*3000000-125000,
                "desired=",desired[[i]]*3000000-125000];
                Print["Error=",deviation[[i]]," % Deviation =",PD[[i]]];
                Print["                    "];
                errorsum=errorsum+Abs[deviation[[i]]];
                PDsum=PDsum+PD[[i]];];

        Print["Mean  error=",errorsum/len];
        Print["Mean percent deviation=", PDsum/len];];
```

```
Neurotest[outs[[1]],outs[[2]],ioPairs]
input1={0.222142, 0.20332, 0.5, 0.63, 0.4, 0.0854265,

 0.4}
output1={530558.}desired={562383.}
Error={31824.9} % Deviation ={5.65893}

input2={0.4445, 0.423332, 0.5, 0.606667, 0.4,

 0.175968, 0.4}
output2={781749.}desired={806141.}
Error={24392.1} % Deviation ={3.02579}

input3={0.2363, 0.417, 0.25, 0.38, 0.4, 0.100073, 0.4}
output3={381337.}desired={350193.}
Error={-31144.1} % Deviation ={8.89341}

input4={0.379933, 0.556, 0.25, 0.593333, 0.4,

 0.291939, 0.8}
output4={758623.}desired={755613.}
Error={-3010.04} % Deviation ={0.398357}

input5={0.9583, 0.91268, 0.5, 0.606667, 0.4, 0.89315,

 0.4}
              6              6
output5={2.51031 10 }desired={2.54774 10 }
Error={37430.1} % Deviation ={1.46915}

input6={0.589867, 0.632, 0.75, 0.42, 0.8, 0.35725,

 0.4}
                       6
```

output6={986315.}desired={1.00399 10 }
Error={17678.5} % Deviation ={1.76082}

input7={0.294, 0.588, 0.25, 0.636667, 0.8, 0.188875,

0.4}
output7={582375.}desired={510736.}
Error={-71639.2} % Deviation ={14.0267}

input8={0.2905, 0.2768, 0.5, 0.606667, 0.8, 0.15, 0.4}
output8={568058.}desired={523975.}
Error={-44083.2} % Deviation ={8.41323}

input9={0.819733, 0.564, 0.5, 0.746667, 0.4, 0.4668,

0.4}
6 6
output9={1.68601 10 }desired={1.64276 10 }
Error={-43251.6} % Deviation ={2.63286}

input10={0.3416, 0.488, 0.25, 0.676667, 0.8, 0.17475,

0.4}
output10={614588.}desired={739749.}
Error={125161.} % Deviation ={16.9194}

Mean error={42961.5}
Mean percent deviation={6.31986}

Neurotest[outs[[1]],outs[[2]],Tspairs]
input1={0.215193, 0.20332, 0.5, 0.601333, 0.4,

0.0810735, 0.4}
output1={504061.}desired={516711.}

Error={12650.1} % Deviation ={2.4482}

input2={0.440333, 0.44, 0.25, 0.873333, 0.8, 0.24674, 0.8}
output2={809602.}desired={832565.}
Error={22962.8} % Deviation ={2.75808}

input3={0.2141, 0.21408, 0.5, 0.58, 0.4, 0.0669, 0.4}
output3={470100.}desired={479018.}
Error={8917.74} % Deviation ={1.86167}

input4={0.444, 0.444, 0.5, 0.58, 0.8, 0.20515, 0.8}
output4={616371.}desired={627737.}
Error={11366.4} % Deviation ={1.81069}

input5={0.2905, 0.2768, 0.5, 0.606667, 0.8, 0.15, 0.4}
output5={568058.}desired={562817.}
Error={-5241.31} % Deviation ={0.931263}

input6={0.5936, 0.564, 0.5, 0.606667, 0.4, 0.37965, 0.4}
output6={1.32943 10 }desired={1.37877 10 }
Error={49341.5} % Deviation ={3.57865}

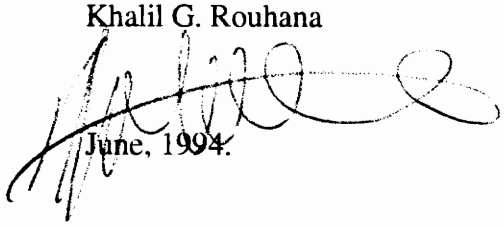
input7={0.653467, 0.748, 0.25, 0.74, 0.8, 0.4763, 0.4}
output7={1.38446 10 }desired={1.46035 10 }
Error={75885.2} % Deviation ={5.19638}

Mean error={26623.6}
Mean percent deviation={2.65499}

VITA

Khalil Rouhana was born on November 26, 1970 in Zahle, Lebanon. He graduated from St. Joseph high school, Zahle, in June 1988. He received the degree of Bachelor of Science in Mechanical Engineering from the American University of Beirut in October 1992. In August 1993, he joined the graduate construction engineering and management program in the Civil Engineering department at Virginia Tech. After completing the requirements of a Master of Science degree in Civil Engineering in June 1994, he is working as a construction engineer for Harmon Contract, Inc., in Columbia, Maryland.

Khalil G. Rouhana



June, 1994.