DESIGN OF NORMAL CONCRETE MIXES USING NEURAL NETWORK MODEL

A thesis submitted to the Graduate School in partial
fulfillment of the requirements for the degree
Master of Science (Information Technology),
Universiti Utara Malaysia

by
Mohd Dzulkonnain bin Abu Bakar

Sekolah Siswazah
(Graduate School)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certification of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

MOHD DZULKONNAIN BIN ABU BAKAR

calon untuk Ijazah
(candidate for the degree of) Sarjana Sains (Teknologi Maklumat)

telah mengemukakan kertas projek yang bertajuk
(has presented his/her project paper of the following title)

DESIGN OF NORMAL CONCRETE MIXES USING NEURAL NETWORK MODEL

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan, dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the field is covered by the project paper).

Nama Penyelia *(Name of Supervisor)*	: Prof.Ir.Dr.Che Sobry Abdullah/Puan Fadzilah Siraj
Tandatangan *(Signature)*	: [illegible]
Tarikh *(Date)*	: 19/11/2000

GRADUATE SCHOOL
UNIVERSITI UTARA MALAYSIA

PERMISSION TO USE

ABSTRAK (BAHASA MALAYSIA)

Faktor terpenting dalam menentukan kualiti konkrit ialah kekuatannya. Untuk mencapai kekuatan yang dikehendaki, nisbah bahan-bahan dalam konkrit seperti air, simen, pasir dan batu baur hendaklah dikenalpasti. Kaedah rekabentuk campuran yang ada pada masa kini seperti kaedah ACI dan DoE yang melibatkan banyak pengiraan, carta rekabentuk dan jadual adalah rumit serta panjang. Tujuan projek ini adalah untuk membina satu kaedah rekabentuk campuran konkrit lebih mudah dan umum dengan mengunakan teknik rangkaian neural. Tatacara untuk membina model rangkaian neural menggunakan rangkaian perambatan balik dan beberapa isu berkaitan dengan penyediaan data, dibincangkan bagi membantu pembangunan aplikasi yang berkesan. Dapatan projek ini menunjukkan bahawa aplikasi rangkaian neural mampu menyediakan penyelesaian kepada masalah kejuruteraan awam, terutamanya dalam merekabentuk campuran konkrit .

ABSTRACT

The most important factor in determining the quality of concrete is its strength. In order to achieve the required strength, a right proportion of materials in concrete such as water, cement, sand and course aggregate, need to be identified. The present mix design methods such as ACI and DoE methods, which involve numerous calculations, design charts and table look-up are seem to be tedious and lengthy. The purpose of this project is to develop a simpler and generalized concrete mix design method using neural network techniques. A procedure for developing neural network models using back propagation networks is presented, and a number of issues related to data preparation are described to facilitate the development of efficient application. The findings of this project show that the application of neural network is capable of providing solutions to the civil engineering problem, particularly in designing the concrete mixes.

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful. Praise be to Allah the Cherisher and Sustainer of the world. Many thanks to Allah for giving opportunity to me to broaden my academic horizons, which would be useful for the benefit of humankind.

The completion of this project has a special meaning in my life. Therefore I would like to express my gratitude to all these people who have spent their energy and time to assist me in this work.

First of all, my greatest thanks goes to my supervisors Prof. Ir. Dr. Che Sobry Abdullah and Puan Fadzilah Siraj for their close guidance, valuable criticism, advice and technical support in completion of the project and writing this report.

The completion of this project is also due to the hope, prayer and support of my beloved wife Hanissah A. Razak and my children Muhammad Farhan, Muhammad Ariff and new born Muhammad Zikry as a source of inspiration to complete this project on time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Concrete becomes a material that literally forms the basis of our modern society and until today, concrete is the most widely used man-made construction materials. Most of all buildings, drains, dams, piles and bridges are made of concrete, including some portions of highways. Concrete structures are everywhere. Concrete offers a lot of advantages such as the ability to cast, economical, durable, fire resistant, on-site fabrication and aesthetic properties (Sidney and Young, 1981).

Due to lots of usage and its importance especially in civil and construction engineering, there are needs to fully utilize or optimize the capabilities of the concrete. As higher and higher performance is sought from concrete, obtaining the proper mixture proportion to achieve specific objectives is becoming more difficult (Simon, *et al.*, 1999). It has to be designed with the correct mixture in such a way that it can perform to the required strength, durability, workability, safety, economics and other specified elements. Durability is the ability of concrete to withstand the conditions for which it has been designed, without deterioration over a period of years. Meanwhile, the workability is the ability of concrete to compact and easy to work with.

Basically, concrete is produced by mixing cement with fine aggregate (sand), coarse aggregate (gravel or crushed stone), water and small amounts of various chemicals called *admixtures*. The admixtures control properties such as setting time and plasticity, whereas cement is the key ingredient in concrete and acting as the binding agent that holds sand, and gravel together in a stone-like mass when the concrete becomes hard. The process of hardening or setting is actually a chemical reaction called *hydration*. When water is added to the cement, it forms a slurry or gel that coats the surfaces of the aggregate and fills the voids to form the solid concrete.

Concrete strength is affected by many factors such as types and size of aggregate, chemical composition and fineness of cement, water/cement ratio, time, temperature of hydration (Sidney and Young, 1981), air entrapment, admixtures, preparation method (Hsu and Tsai, 1997) and conditions of surrounding area. In order to take account of these factors that affect the concrete strength, in 1975 the British Department of the Environment (DoE) has published a procedure of concrete mix design. The procedure is based on data obtained from the Building Research Establishment, the Transport and Road Research Laboratory, and by the British Cement Association. The method is then revised in 1988. Since this procedure involves a lot of calculations, graphs and tables look-up, the alternative procedure that can provide quicker solution need to be explored. In seeking the alternative method, the neural network technology is able to relate all those factors affecting the concrete strength. Furthermore, in view of the problem nature in this project, which requires a method to solve a classification problem, neural network is found to be the best approach as compared to other artificial intelligence (AI) tools such as expert system, fuzzy logic or case-based reasoning (CBR). The expert system is a rule-based approach, which requires knowledge from human expert to solve a particular problem. It only suits problems that require a solution by following a certain rules (Luger and Stubblefield, 1998), whereas, the fuzzy logic is suitable for solving a problem that deals with uncertainty and inexact information or imprecise term (Welstead, 1994; Awad, 1996). Since the project deals with precision rather that uncertainty, fuzzy logic will not be explored in this project. Meanwhile, CBR is a technique that records and documents cases and then searches the previous cases for solving a current problem. The old solution is adapted to fit a new case, even they are not exactly the same (Awad, 1996). Since the CBR is unable to generalize, it is not suitable for solving the problem of this project. Therefore, for this project, neural network has been employed as a tool to improve the existing process of designing concrete mixes.

1.1 Problem Statement

Concrete is inexpensive and readily available material, and easy to work with. Although concrete has these advantages, the process of designing the concrete mix remains a complex task involving numerous calculations, graphs and tables look-up such as in DoE (1988) and ACI (1994) methods. Every time in designing a concrete mix, the same procedure as outlined by these methods has to be followed. A simpler method such as neural network, which is possibly able to generalize the concrete mix design, can be the alternative design method. This alternative method has to assure that the required strength, durability, workability, safety, economics and other specified elements of the produced concrete are complied with. It should be born in mind that the mix design method is only produce the initial mix proportion and should be followed by trial mixes where adjustments are made to achieve the required performance.

1.2 Objectives

The main objective of this project is to develop a neural network model as an alternative method in designing the concrete mixes. The project will attempt to develop a system that is simpler, faster and more accurate in designing the concrete mixture using the back propagation neural network modeling. The developed model can be used to evaluate the concrete properties and have the know-how of the interaction of different materials and proportion for optimum usage. Users can play around with the input parameters and see their effects to the outputs.

Other objective of this project is to apply Information Technology (I.T.) in civil engineering and to gain the advantages offered by the neural network technology.

1.3 Scope of the project

This project is mainly to model a concrete mix design according the DoE method using neural network. The data for training and testing the neural network model, are generated through the DoE mix design method. This method has been chosen for this project because it is widely used in this country. In order to achieve the objectives of this project, the following exercises have been carried out:

i) Develop Neural Network Simulator using back propagation technique to train and test the generated data

ii) Identify two neural network models to:

    (a) Predict concrete mixture by specifying concrete grade

    (b) Predict concrete grade by specifying the mixture.

iii) Develop user interface for the prediction of concrete mix design and strength.

iv) All program codes are written in Microsoft Visual Basic 5.0 programming language and data is prepared in Microsoft Excel 2000 spreadsheet.

1.4 Significance of the project

The project is significant since it is the first project initiated to design a concrete mix using neural network technique. The result of this project is expected to benefit the construction industry as a whole and those people involved in the industry such as engineers, contractors, developers, ready mix concrete suppliers, concrete product producers and as well as researchers.

Engineers can use the developed software to help them in developing the concrete specification and design. The contractors can use it to design their on-site concrete mix whereas the ready mix concrete suppliers can integrate it with their batching plants system to produce concrete according to the customers' requirements. Meanwhile, the concrete product producers can use it to manipulate the mix of concrete to suit to their products and the developers can estimate the concrete cost by identifying quantity of materials such as cement, sand and course aggregate. This project is also beneficial to the researchers as a reference in finding the solution to their problems especially in the engineering domain.

CHAPTER 2

NEURAL NETWORK

2.0 Overview

In order to make the presentation of the developed neural network more meaningful, a brief description of neural network concepts is explained in this chapter. For better understanding, it is also helpful to explore the actual building blocks of biological neural system.

2.1 What is a Neural Network

The main objective of the neural network technology is to mimic the brain approach in processing data and information by capturing the architectural and functional aspects of intelligent behavior (Schalkoff, 1997). That is why a neural network or sometimes called an artificial neural network is a form of artificial intelligence (Ural and Saka, 1998). The most basic element of the human brain is a specific type of cell, which provides the human beings the abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons and each of these neurons usually can connect between 1000 up to 10000 with other neurons through what is called synapse (Schalkoff, 1997). The power of the brain comes from the numbers of these basic components and the multiple connections between them.

All natural neurons have four basic components, which are dendrites, soma, axon, and synapses. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result. Figure 2.1 shows a simplified biological neuron and the relationship of its four components.

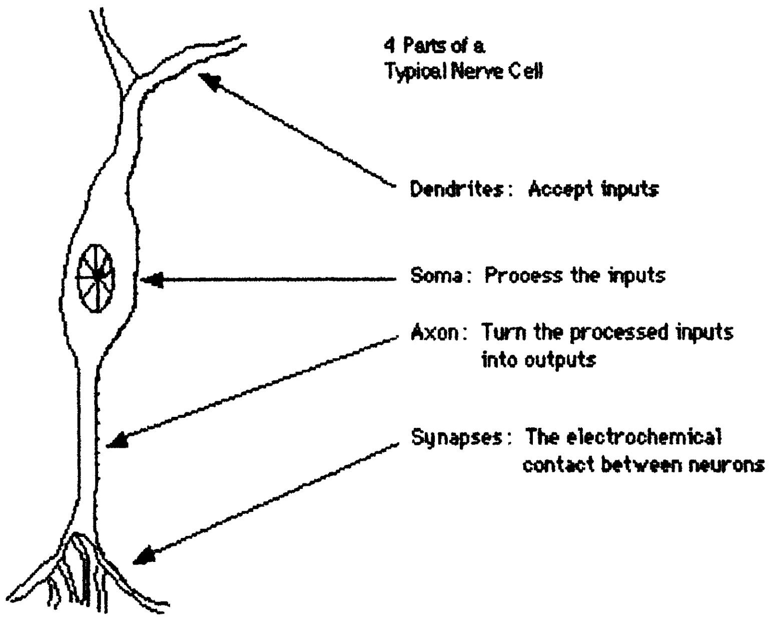

4 Parts of a
Typical Nerve Cell
Dendrites: Accept inputs
Soma: Process the inputs
Axon: Turn the processed inputs into outputs
Synapses: The electrochemical contact between neurons

Figure 2.1: A biological neuron

The basic unit of neural network, which is called artificial neuron, simulates the four basic functions of biological neurons. Artificial neuron is much simpler than the biological neuron. Figure 2.2 shows the basics of an artificial neuron, which is based on the model proposed by (McCulloch and Pitt, 1943). Note that various inputs to the network are represented by the mathematical symbol, x_n. Each of these inputs are multiplied by a connection weight and these weights are represented by w_n. In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output.

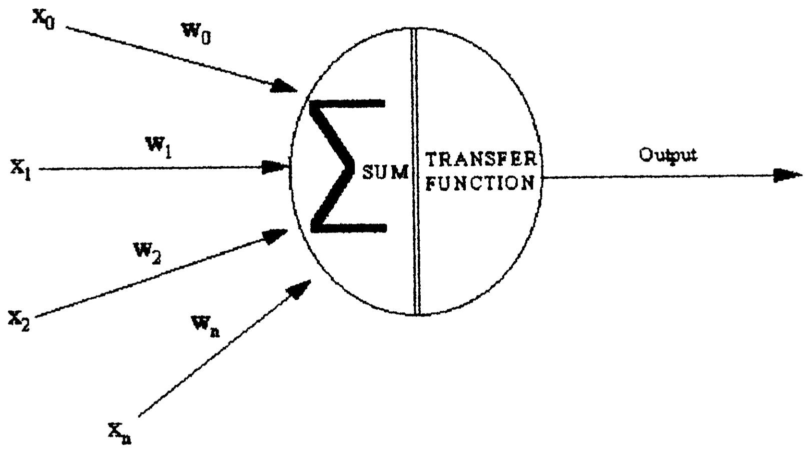


Figure 2.2 : An artificial neuron

The input signal to each neuron can be mathematically defined as:

$$in = \sum_{i=0}^{n} x_i \times w_i \qquad 2\text{-}1$$

and the output signal from neuron is determined after applying a transfer or activation function.

$$out = f(in) \qquad 2\text{-}2$$

Specifically, a neural network consists of a number of artificial neurons or processing units. Each neuron is connected to other neurons by means of directed communication links, each with an associated weight or strength and formed a complete network. The weights represent the knowledge being used by the network to solve a problem. These weights are determined and fixed during the learning or training stage of the network. To train the network, a number of examples of a particular problem are given or shown to it. The network learns and makes adjustments to its weights from these examples in much the same way that people learn through experimentation and interaction with their environment.

The processing units of neural network are structured in three layers of nodes: input, hidden and output layers as shown in Figure 2.3. The number of nodes or processing units in the input and output layer, represents the number of input and output results.

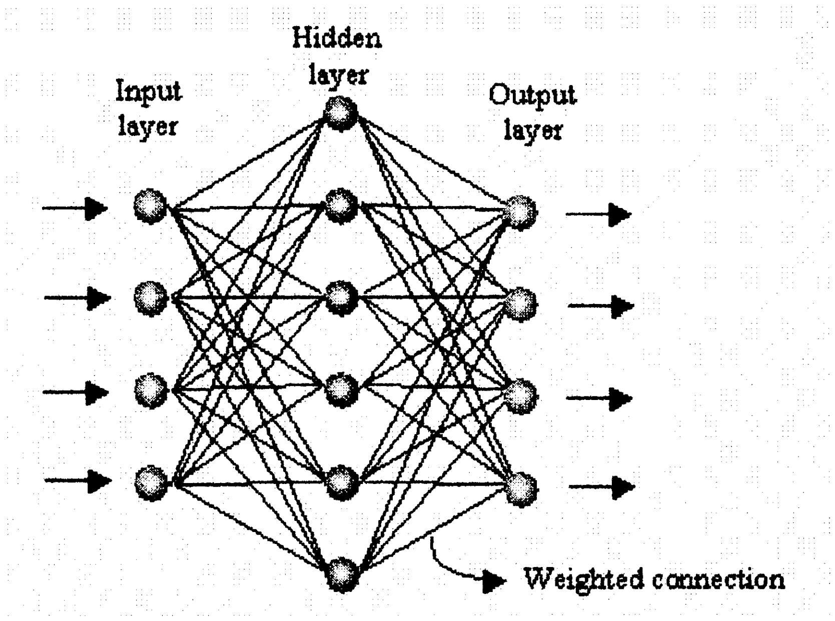


Figure 2.3 : Typical structure of neural network

Briefly, the input layer passed the received input data to the hidden layer, so that it can be processed. The output from the hidden layer is then pass to the output layer as the final network result. The most important function of network is performed by the hidden layer. It is the neurons in this layer that have the job of associating a particular input pattern with the appropriate desired output values. The ability of the hidden layer to perform this job surprisingly well has made neural networks such useful tools.

2.2 Neural Network Models

Neural network model is defined by its topology, learning paradigm and learning algorithm (Bigus, 1996). The topology of neural network describes the flow of data between the input, hidden and the output units. Basically, there are three main categories of connection topologies, which are feed forward, limited recurrent and fully recurrent. In the feed forward topology, the data flows through the network in one direction and the result based solely on the current set of input pattern. The limited recurrent topology on the other hand required the network to store a record of prior inputs and factor them in with the current data to generate the output. Whereas, the fully recurrent topology is quite complex, which provides a two-way connections between all units and data flows to all adjacent units back and forth until the activation of the units stabilizes (Bigus, 1996).

There are two main learning paradigm or approach to develop neural network applications, which are supervised learning and unsupervised learning . In a supervised learning, the designed network is presented with a training set of several input-output pairs of examples. Each time an input is presented, the net produces an output. The produced output is then compared with the corresponding desired or target output. If there is a discrepancy, the net computes the error and makes corrections by updating the weights of the links that connecting the units together. This explains the learning algorithm of the network. The network is consider completed one iteration or epoch of the training process after the entire set of the input-output training pairs have been presented. This training process is repeated several hundreds, or more times until the output errors become small and the net outputs are within a user-specified tolerance level for all the input-output training pairs.

In an unsupervised learning, no sample outputs are provided to the network. When presented with a set of input patterns, the network will make a clustering process by putting all similar patterns into the same cluster.

The well-known neural network models such as Adaptive Resonance Theory (ART), Back propagation (BP), Radial basis function (RBF), Probabilistic neural network (PNN), Kohonen feature maps (KFM), Learning vector quantization (LVQ) and Recurrent back propagation (RBP) are listed in Table 2.1, together with their functions.

Table 2.1: Neural network models and their functions (Bigus, 1996)

Model	Learning	Topology	Function
ART	Unsupervised	Recurrent	Clustering
BP	Supervised	Feed forward	Classification
RBF	Supervised	Feed forward	Classification, time-series
PNN	Supervised	Feed forward	Classification
KFM	Unsupervised	Feed forward	Clustering
LVQ	Supervised	Feed forward	Classification
RBP	Supervised	Limited recurrent	Time-series

2.3 Back Propagation Neural Network

Among the variety of neural network paradigms, the back propagation (also called multilayer perceptron, MLP) is perhaps the most commonly use to train networks and has been applied successfully to a broad range of areas (Awad, 1996; Ural and Saka ,1998) ranging from character and speech recognition to playing backgammon. Back propagation neural network is introduced in this section since it has been implemented in this project.

The structure of the back propagation network is quite similar to the typical structure shown in Figure 2.3. The different with other networks is its learning approach, which is called back propagation algorithm. This algorithm was proposed by (Rumelhart *et al.*, 1986) and was responsible in large part for the reemergence of neural networks in the mid 1980s (Bigus, 1996).

2.3.1 Back Propagation Algorithm

Basically, the back propagation algorithm consists of three stages: feed forward the input pattern, compute and back propagate the errors and finally updating the weights. In the feed forward stage, the input patterns are presented to the network. These inputs are then propagated forward from the input layer, through hidden layer, to the output layer, resulting in the network output. Because back propagation is a supervised learning algorithm, the target or desired outputs are given as part of the training requirements.

During the second stage, the actual network output in each output unit is compared with its target to determine the associated error for that pattern. Based on these errors, the corrections to the connecting weights are calculated. The process of getting these weights correction term, starts with the output units and propagates backward through the hidden layer to the input layer - hence the term back propagation.

The third stage involved with updating the weights by correcting each of the existing weights based on the above calculated weights correction term. Figure 2.4 shows the flow of the back propagation training process or algorithm based on the architecture of back propagation network in Figure 2.5.

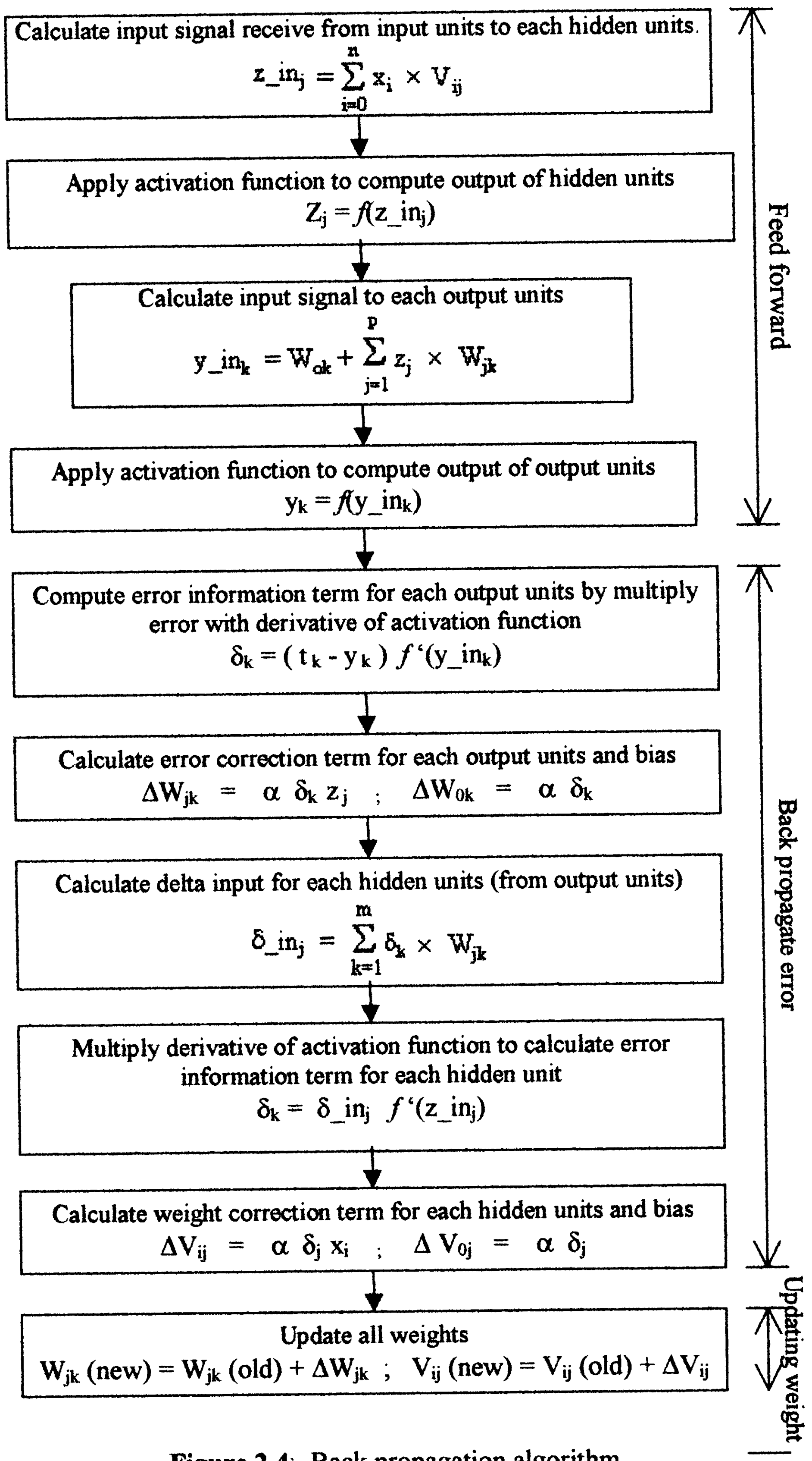

Calculate input signal receive from input units to each hidden units.
$z\_in_j = \sum_{i=0}^{n} x_i \times V_{ij}$
Apply activation function to compute output of hidden units
$Z_j = f(z\_in_j)$
Calculate input signal to each output units
$y\_in_k = W_{ok} + \sum_{j=1}^{p} z_j \times W_{jk}$
Apply activation function to compute output of output units
$y_k = f(y\_in_k)$
Compute error information term for each output units by multiply error with derivative of activation function
$\delta_k = (t_k - y_k)\, f'(y\_in_k)$
Calculate error correction term for each output units and bias
$\Delta W_{jk} = \alpha\, \delta_k\, z_j \ ; \ \Delta W_{0k} = \alpha\, \delta_k$
Calculate delta input for each hidden units (from output units)
$\delta\_in_j = \sum_{k=1}^{m} \delta_k \times W_{jk}$
Multiply derivative of activation function to calculate error information term for each hidden unit
$\delta_k = \delta\_in_j\, f'(z\_in_j)$
Calculate weight correction term for each hidden units and bias
$\Delta V_{ij} = \alpha\, \delta_j\, x_i \ ; \ \Delta V_{0j} = \alpha\, \delta_j$
Update all weights
$W_{jk}(new) = W_{jk}(old) + \Delta W_{jk} \ ; \ V_{ij}(new) = V_{ij}(old) + \Delta V_{ij}$
Feed forward
Back propagate error
Updating weight

Figure 2.4: Back propagation algorithm.

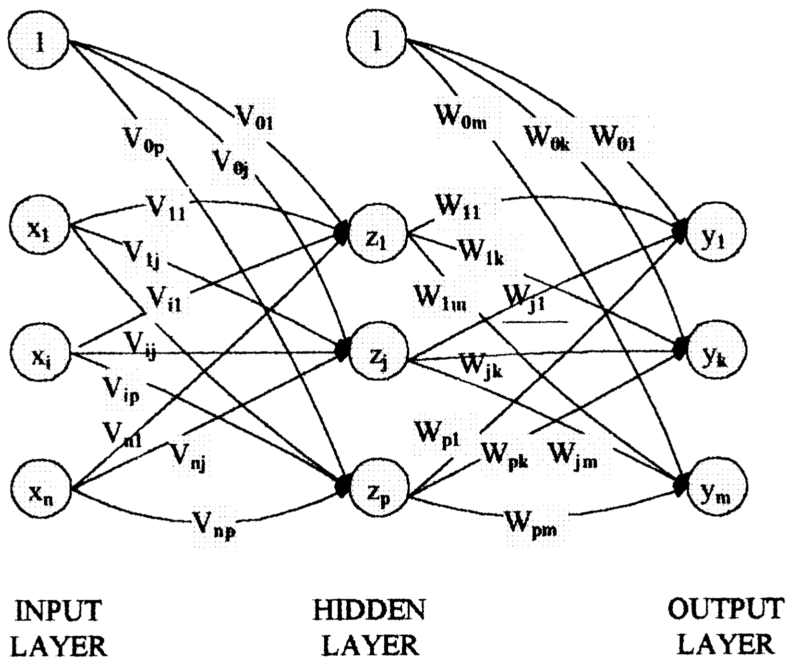


Figure 2.5: Back-propagation neural network architecture

2.3.2 Activation Function for Back Propagation Network

An activation function (sometimes is called a transfer or squashing function) is used to map neuron input activation to an output signal of hidden unit. The function is needed to introduce non-linearity into the network. It is the non-linearity (i.e. the capability to represent nonlinear functions) that makes neural networks so powerful.

The most widely used activation function is sigmoid. This is due to the facts that (Schalkoff, 1997):

i) It squashes very well.

ii) It is semilinear (i.e. nondecreasing and differentiable everywhere).

iii) It is expressible in closed form.

iv) Modifications or extensions lead to or relate to other activation functions.

v) Its derivative is easy to form.

This function is also suitable to use in back propagation network together with hyperbolic tangent (tanh) function. There are two types of sigmoid activation function, namely the binary and the bipolar sigmoid functions. The binary sigmoid function is used when the values in the data set are between 0 and 1. This function also known as logistic function (Schalkoff, 1997). The bipolar sigmoid function is used when the values in the data set are between -1 and +1. The tanh function is quite similar to the bipolar sigmoid function that produces both positive and negative values. The functional form for these three activation functions are written in the following equations.

Binary sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad 2\text{-}3$$

and it first derivative is :

$$f'(x) = f(x)\,[1 - f(x)] \qquad 2\text{-}4$$

Bipolar sigmoid function:

$$f(x) = \frac{2}{1 + e^{-x}} - 1 \qquad 2\text{-}5$$

and it first derivative is :

$$f'(x) = \frac{1}{2}\,[\,1 + f(x)\,][1 - f(x)] \qquad 2\text{-}6$$

Hyperbolic tangent (tanh) function:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad 2\text{-}7$$

and it first derivative is :

$$f'(x) = \frac{4}{(e^{x} + e^{-x})^{2}} \qquad 2\text{-}8$$

CHAPTER 3

LITERATURE REVIEW

3.0 Overview

This chapter provides a review on the neural networks applications and other AI tools in the field of civil engineering. The existing methods of designing the concrete mixes are also discussed.

3.1 Neural Network Application in Civil Engineering

Many of the problems that engineers must deal with are exactly the types of problems for which neural networks appear to be most applicable. Engineers are interpreters of incomplete, noisy data - such as interpreting sensor information to determine the existence, and location of damages in a structural system. Engineers are designers and controllers of complex systems for which there is no exact model of behavior and expected performance is unknown, requiring it to be estimated.

Neural networks are being used in civil engineering in a variety of applications. This section illustrates some of the different types of problems to which neural networks have been applied.

Neural network has been investigated as a means to develop efficient predictive models of the structural behavior of concrete slabs (Hegazy, *et al.*, 1998). Four neural networks were trained and tested using the experimental results of 38 full-scale slabs. Hsu and Tsai (1997) have studied a diagnosing model for reinforced concrete structures by using of back propagation neural network technique to assess the severity and location of defects. Theoretical analysis of a simply-supported reinforced concrete beam in specified size (i.e., rectangular cross section and 4 meter span) by finite element program is performed to generate training and testing samples for neural network assessing task. The same approach has been used by (Abdullah, *et al.*, 2000) in estimating the in-situ strength of concrete. They have used multilayer perceptron (MLP) network to develop a generalized prediction models based on combined non-destructive testing techniques, namely, the rebound hammer test and ultrasonic pulse velocity test.

Neural network has been trained to identify civil engineering structures using records from structural responses under different earthquake loading conditions (Amini, *et al.*, 1997). The study indicates that the trained neural network is capable of providing sensible outputs (generalization) when presented with input data that has never been used during its training. Meanwhile, Hsu *et al.* (1993) have shown that neural network also could be applied in hydrology area, where they have used three layer feed-forward network model for simulating the nonlinear hydrologic behavior of watersheds.

3.2 Other Artificial Intelligence (AI) Applications in Civil Engineering

The capabilities of AI approaches have inspired Che Wan Putra, *et al.* (2000) to conduct a study to develop a prototype of expert system that can generate materials scheduling by the integration of construction scheduling and knowledge-based systems. The same approach has been used by (Zhao and Fan, 2000) to design a concrete box girder bridge. They developed an expert system, which integrates the neural network and the fuzzy network in order to provide a few feasible design configurations. On the other approach, Hung and Jan (1997) have developed an unsupervised fuzzy neural network case-based learning model to recall the similar cases, which have been designed before and obtain the solution from these similar cases in a way of adaptation or synthesis.

Genetic algorithm is another AI technique, which is based on Darwinian strife for survival theory. This technique has been applied by Hadi (2000) to calculate the optimum solution in designing a reinforced concrete T beams. He has proven that the genetic algorithm can achieve the goal of optimization of T beams quickly and without the need to use high level of mathematics. .

3.3 Existing Method in Designing Concrete Mixes

The concrete mix design is a process of selecting suitable ingredients for concrete and determining their proportions which would produce, as economically as possible and concrete that satisfies the job requirements such as strength, workability and durability (Gambhir, 1995).

There are two most popular methods in designing concrete mix, which use certain empirical relations as a guide to select the best combination of the ingredients of concrete. The first method is known as American ACI method which is followed the ACI Standard Practice ACI 211.1-91 (1994). This standard describes a method of selection of mix proportions of concrete containing Portland cement alone or together with other cementitious materials, and admixtures (Neville, 1995). This method is based on the fact that for a given maximum size of well shaped aggregate, the water/cement ratio determines the workability of mix and it is largely independent of mix proportions (Gambhir, 1995). Generally, the method consists of eight steps, which take into account the characteristics of materials to be used. These steps are:

i) Choice of slump

ii) Choice of maximum size of aggregate

iii) Estimate of water content and air content

iv) Selection of water/cement ratio

v) Calculation of cement content

vi) Calculation of coarse aggregate content

vii) Estimate of fine aggragate content

viii) Adjustments to mix proportions.

The second method, which is quite similar to the ACI approach, is the British DoE method (Neville, 1995). The method is suitable for the design of normal concrete mixes having 28-day compressive strength as high as 75 N/mm2 for non-air-entrained concrete. Description of the procedure is given in Section 4.2.1.

3.4 Statistical Approach in Designing Concrete Mixes

There is an attempt to design the concrete mixture with statistical approach (Simon, *et al.*, 1997). In the study, a statistically designed mixture experiment is used to identify the best factor settings for optimizing properties of high performance concrete. They have prepared thirty nine batches of concrete in this experiment to find the optimum proportions for a concrete mix meeting the following conditions: 50 – 100mm slump for the fresh concrete, 1-day target compressive strength of 22.06 N/mm2, 28-day target compressive strength of 51.02 N/mm2, target 42-day "rapid chloride" (ASTM C1202) test (RCT) measurement less than 700 coulombs, and minimum cost (dollars per m3). The mixing materials used are water, cement, microsilica (silica fume), high-range water reducing admixture (HRWRA), course aggregate, and fine aggregate.

The same approach has also been used by NSERC (1996) in building the automated concrete mix design prototype model to provide optimum productivity/performance which may result in a significant cost saving to a ready-mix concrete plant. A large number of functions have been used in their concrete mix optimization model to describe the relationships between the many variables affecting the design, batching, mixing, delivery and quality control of ready mix concrete. Generally, in terms of performance, the statistical approach is less compared to the neural network (Abdullah, *et at.*, 2000).

CHAPTER 4

PROJECT METHODOLOGY

4.0 Overview

This chapter describes the methodology adopted for this project. The key issues that have been considered while developing the application such as design of network architecture are discussed. The most important issues in neural network, is data. Since there are no data available for this project, the application starts with generating a concrete mix design data. This generated raw data was not in a suitable format to be used for training and testing the neural network. It had to be pre-processed and translated into useful format before it can be presented to the network. Prior to the development of the neural network simulator, the network architecture or model is identified. The selected topology, learning approach and learning algorithm are used in developing the neural network simulator. Training and testing will be carried out once the development of the data and the simulator are completed. Several network configurations will be trained and tested in order to find out the network model, which gives the best performance. The best weight file will be kept and used in the development of concrete design system.

Generally, there are five phases involved in developing the neural network application for designing the concrete mixture.

i) Design of Neural Network Architecture.

ii) Data Preparation

iii) Development of Neural Network Simulator

iv) Training and Testing

v) Development of Conrete Design System

These phases are described in the proceeding sections.

4.1 Designing Neural Network Architecture

First step in designing neural network model, is to identify the nature of our problem that we are going to solve. This will determine the suitable network topology is selected. Only certain network topologies were suitable to perform the required function to solve our problem. The next step is to look at the input data whether it is all binary (0/1) or bipolar (-1/+1) or the data contains real-valued inputs. These types of data might disqualify some of the network architectures, which used certain activation functions in their learning algorithm. The last step is to determine the number of input and outputs units, and the hidden nodes that can give the best performance.

The problem of this project is to classify the mixture of concrete that can give a required strength based on a various factors. Therefore, the designed network must be able solved the classification problem. The network has to map the features of the inputs and produced the desired output. So, the network has to use the supervised training approach. Since the prepared data, contains real values between 0 and 1, the most suitable activation function is binary sigmoid function. This function is suitable to use in back propagation learning algorithm. Since enough training data is already available and can represent the problem to be solved, the training of the network is done offline.

As such, the back propagation network is selected as the best model since it has satisfied all the above requirements (solve classification problem, need supervised training approach, use binary sigmoid activation function and offline training).

Lastly, in determining the number of input and output units in the network, once again it depends to the problems that are to be solved. For this project, seven factors have been identified as the most significant factors that affect the amount of the four main ingredients in concrete. The details of these factors and four main materials in concrete are listed in Table 4.1 and Table 4.2.

4.2 Data Preparation

It is certainly true that having data is a necessary prerequisite to solve problem using neural network approach. Preparing the data is the most important part of the project development. This procedure is crucial to the success of applying neural network approach. The performance of a neural network is largely depend upon the data set it was trained, especially when using the back propagation network that learns from the input/output examples. In general, the better the training data sets represent the problem to be solved, the better performance of the neural network. Only someone with domain knowledge, someone who understands the data and what it means, can select the right data set to be trained in the neural network (Bigus, 1996). By selecting a wrong data, the neural network may not able to give desired results accurately. Figure 4.1 shows the data preparation process for this project.

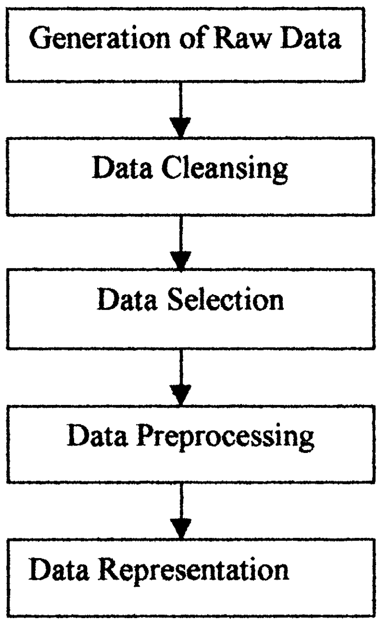


Figure 4.1: Data preparation process

4.2.1 Generation of Raw Data

This project used a set of generic raw data that follows the British DoE (1988) method in designing a concrete mix. Briefly, there are five stages involved in the mix design process.

Stage 1 is to determine the target water/cement ratio. In this stage, characteristic strength of concrete, standard deviation, percentage of defect, type of cement and type of aggregate, are specified. The concept of target mean strength is introduced, which is defined as:

$$fm = fc + k \times s \tag{4-1}$$

where	fm	=	the target mean strength
	fc	=	the characteristic strength
	k	=	a constant
	s	=	the standard deviation obtained from Figure 3 of DoE (1988).
	$k \times s$	=	the margin

The constant k is derived from the mathematics of the normal distribution of concrete strength and it increases as the proportion of defectives is decreased (DoE, 1988). Thus,

$$k \text{ for } 10\% \text{ defectives} = 1.28 \tag{4-2}$$

$$k \text{ for } 5\% \text{ defectives} = 1.64 \tag{4-3}$$

$$k \text{ for } 1\% \text{ defectives} = 2.33 \tag{4-4}$$

For better illustration, an example of a design concrete mix with the following specifications shall be used.

i) characteristic strength of 30 N/mm2 at the age of 28 days.

ii) type of cement is Ordinary Portland Cement (OPC).

iii) type of course aggregate is uncrushed with maximum size of 10 mm.

iv) workability with a slump of 10-30 mm.

v) no previous information of strength test.

vi) 5% defectives level

vii) 40% fine aggregate passing a 600μm sieve

The standard deviation of 8 N/mm2 is selected since we have no previous records of concrete strength test (DoE, 1988). From equations 4-1 and 4-3, the calculated target mean strength is 43.12 N/mm2. Then, a value of 42 N/mm2 is obtained from Table 2 of DoE (1988) for the strength of a mix made with a free-water/cement ratio of 0.5 according to specified age of 28 days, the type of cement is OPC and the aggregate type is uncrushed. This strength is then plotted on Figure 4 of DoE (1988) and a curve drawn from this point and parallel to the printed curves until it intercepts a horizontal line passing through the ordinate representing the target mean strength. The value of 0.49 for free-water/cement ratio is read from the abscissa. If the specification has specified the maximum water/cement ratio, then used the lower value.

Stage 2 is simply of determining the free-water content. In this stage, slump and maximum aggregate size are specified where in this case are 10-30 mm and 10mm respectively. The water content is identified as180 kg/m3 (refer Table 3 of DoE (1988)).

Stage 3 is to determine the cement content, where it was calculated using equation 4-5 with the water/cement ratio (from Stage 1) and water content (from Stage 2), which is equal to 367.35 kg/m3.

$$\text{Cement content} = \frac{\text{water content}}{\text{water/cement ratio}} \qquad 4\text{-}5$$

If the specification has specified the maximum cement content and the calculated is greater than the specified value, then use the specified maximum cement content. Whereas if the specification has specified the minimum cement content and the calculated is less than the specified value, then use the specified minimum cement content. Other than these, use the calculated cement content.

Stage 4 is to determine the total aggregate content by subtracting the cement and water from the wet density of concrete (refer Figure 5 of DoE (1988)). Since there is no information available regarding the relative density of the aggregate, a value 2.6 for uncrushed aggregate should be taken. The value of 2.7 is for crushed aggregate. Therefore, the wet density is 2375 kg/m3 and the total aggregate content of 1827.65 kg/m3 is determined from equation 4-6 below (refer Figure 5 of DoE (1988)).

Total aggregate content = wet density–cement–water 4-6

Stage 5 is to decide how much of the total aggregate should consist of materials smaller than 5mm (classified as sand or fine aggregate content) and larger than that as course aggregate. Depending on water/cement ratio, the maximum size of aggregate, slump and grading of fine aggregate which is defined by its percentage passing a 600μm sieve, the proportion of fine aggregate identified as 46% (refer Figure 6 of DoE (1988)). Finally, the fine and course aggregate content can be calculated as:

Fine aggregate = total aggregate $\times$ proportion of fine 4-7

Course aggregate = total aggregate – fine aggregate 4-8

The process is repeated for other specifications which covers most of the possibilities in the concrete design. For this project, the total of 4080 raw data have been generated and tabulated in Microsoft Excel spreadsheet. A sample of raw data is given in Appendix A.

Limitations to the data set were imposed to maintain simplicity of the problem and not to produce too much data, hence reduced training time. Those limitations are:

i) No specified values of maximum cement content and water/cement ratio are imposed during the generation of data.

ii) The data is catered for Ordinary Portland Cement (OPC) type only.

4.2.2 Data Cleansing

Data cleansing is a process of removing or correcting any data that contain inaccurate values, missing values or other inconsistencies in the data. This process is very important because the performance of neural network is much depends on the accuracy of the training data. The existence of outliers data can reduced the performance of the network (Bigus, 1996). In this project, the cleansing process is done manually by viewing through all the data and any detected incorrect or missing values will be corrected accordingly.

4.2.3 Data Selection

The most important and crucial part in data preparation is to select the right variables that have a strong effect to the output results. After examined the problem, the required result and the data, all the proposed inputs are considered as major factors that affecting the mix proportion of the concrete. Seven input variables and four output variables were proposed in this project. All variables have been listed in Table 4.1 and Table 4.2 and the descriptions of these variables are as follow.

Input variables:

1. Characteristic strength – the specified strength of concrete at the age of 28 days. The generated data consist of seven different strengths, which are 25, 30, 40, 50, 60, 65 and 70 N/mm2.
2. Slump – a measure of the workability of concrete in order to get concrete suitable for placing and compacting under site conditions using the tool available. Slumps that were used are; 0-10mm, 10-30mm, 30-60mm and 60-180mm.
3. Maximum aggregate size – the biggest nominal diameter of the aggregate. The sizes that were used are 10mm, 20mm and 40mm.
4. % of defectives level – is defined from the normal distribution of concrete strength. The defectives levels that were used to generate the project's data are: 10%, 5% and 1%, but user can specified a value between 0% to 10%.
5. Type of aggregate – uncrushed or crushed.
6. % of fine aggregate passing a 600μm sieve – is between 0% to 100%

7. Standard deviation – is between 4 N/mm2 to 8 N/mm2. If no previous record available, the value of 8 N/mm2 will be adequate.

With the above inputs, the mix design process shall be proceeded to determine the mix proportion of water, cement, fine aggregate and course aggregate. These four materials are classified as the output variables.

4.2.4 Data Preprocessing

Once we have a clean of selected data, it must be preprocessed to transform into a form that is presentable to the neural network. The transformation process is done by normalizing every single data in order to distribute the data evenly and scale it into an acceptable range for the network, which depends on the activation function used. After examining the data set, the following normalization scheme was proposed (also known as linear scaling):

$$X(new) = X(old) / X_{max} \qquad 4\text{-}9$$

or

$$X(new) = \frac{X(old) - X_{min}}{X_{max} - X_{min}} \qquad 4\text{-}10$$

where, X(new) and X(old) represented a new and old input value respectively. X_{max} and X_{min} are representing the maximum and minimum values of the selected item respectively. Equation 4(9) is used if zero (0) is existed in the list. The result of the normalization will return the values between 0 and 1.

Another transformation that was encountered in this project is to map symbols to numerical values. In order to change the slump's input that has been categorized as 0-10mm, 10-30mm, 30-60mm or 60-180mm, an evenly distributed value between 0 and 1 is chosen. So, the above four categories can be mapped into 0, 1/3 (0.3333), 2/3 (0.6667) and 1 respectively.

In this project, transforming the outputs values are different from the values in the inputs variables. Since all the outputs can be any values, it is difficult for the network to make a prediction correctly. Therefore, the outputs values must be classified into a number of classes and each class is representing a range of values. For example in the water content's output variable, where all values are grouped into classes with 5kg of range. From the data set, 28 classes have been identified for this first output. The process of transforming the output data is easily can be done in Microsoft Excel by using the provided look-up function.

4.2.5 Data Representation

After the data had been preprocessed, it has to be represented in a suitable form in order for neural network to learn well. If not, the network will not learn at all (Bigus, 1996). Table 4.1 and Table 4.2 shows the final data representation for all inputs and outputs respectively, together with their normalization methods.

Table 4.1: Representation of input variables

Input no.	Input name	Data type	Value	Represented by	Normalization method
1	Characteristic strength	Continuous value	25 to 70	0 to 1	linear scaling
2	% of defective level	Continuous value	0 to 10	0 to 1	linear scaling
3	Standard deviation	Continuous value	4 to 8	0 to 1	linear scaling
4	Type of aggregate	Symbolic	uncrushed crushed	0 1	mapping
5	Slump	Symbolic	0 - 10mm 10-30mm 30-60mm 60-180mm	0 0.3333 0.6667 1	mapping
6	Maximun aggregate size	Continuous value	10 to 40	0 to 1	linear scaling
7	% of fine aggregate	Continuous value	0 to 100	0 to 1	linear scaling

Table 4.2: Representation of output variables

Output no.	Output name	Value	Range of each class	No. of class	Represented by	Normalization method
1	Water	115 to 250	5	28	0 to 1	mapping
2	Cement	185 to 825	10	65	0 to 1	mapping
3	Fine aggregate	240 to 1260	20	52	0 to 1	mapping
4	Course aggregate	420 to 1800	20	70	0 to 1	mapping

4.2.6 Data Randomization and Segmentation

The network must not in any event be trained continuously with the same input patterns and then switched to another pattern (Freeman and Skapula, 1992). The network would tend to memorize the latest pattern only. So, to avoid this situation, the order of the data set had to be randomized. The data was sorted according to the random numbers created by Microsoft Excel in order to get the randomized data set.

The data set is then segmented into two subsets of data. First subset is for training data set, which contains 80% of the total data and the remaining 20% are used for testing the network. The training data set once again had been randomized twice, which produced another two set of training data with same input patterns but in different order. The best set out of these three will be used in training the best model until the highest generalization is achieved.

4.3 Development of Neural Network Simulator

The program has been developed as the neural network simulator based on the selected architecture, which is back propagation network. Therefore, the algorithm of the simulator must follow the back propagation learning algorithm. Figure 4.2 shows the flowchart of Neural Network Simulator's algorithm that has been used for training and testing purposes and it's coding was written in Visual Basics. Please refer Appendix B for the codes.

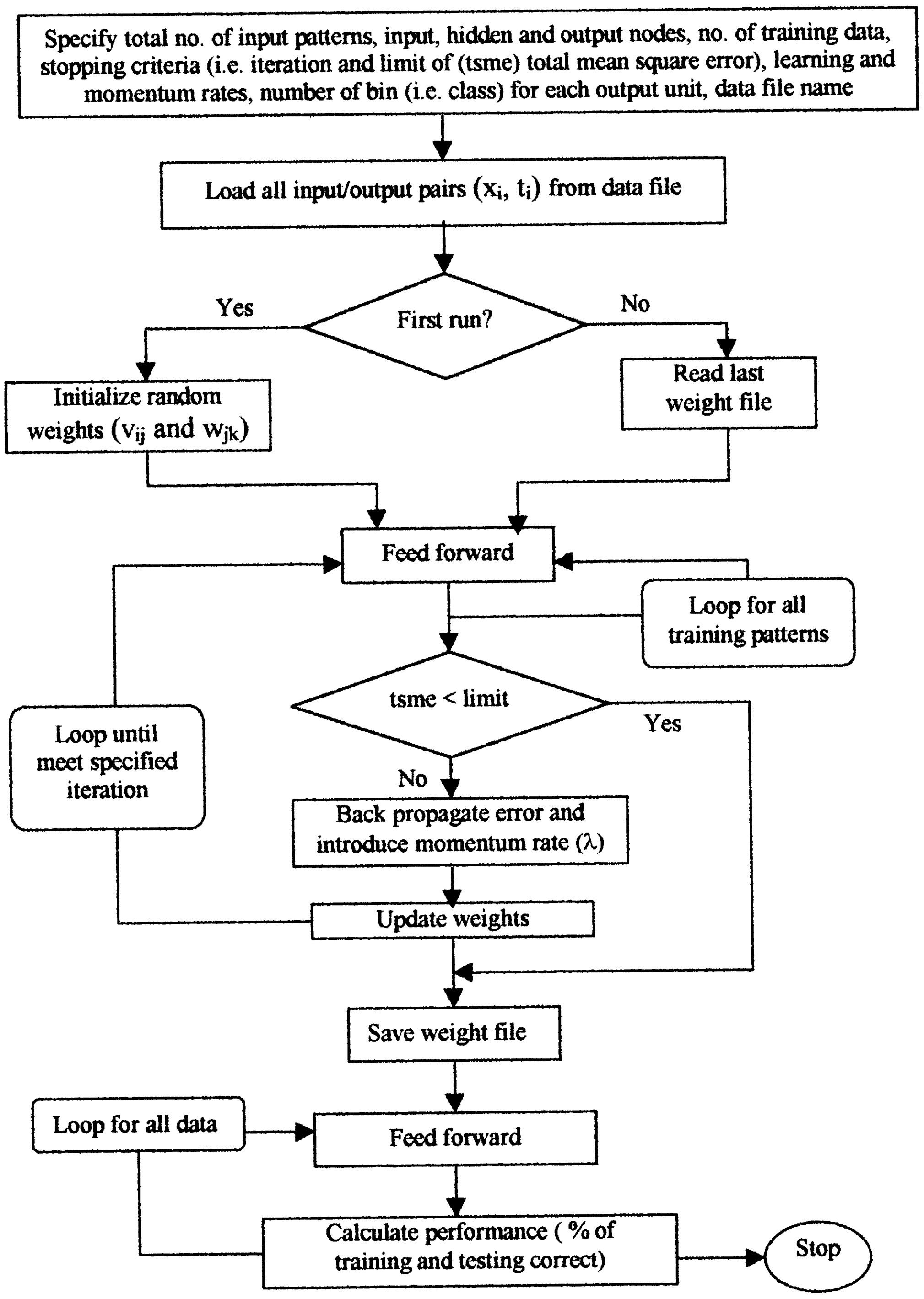

Specify total no. of input patterns, input, hidden and output nodes, no. of training data, stopping criteria (i.e. iteration and limit of (tsme) total mean square error), learning and momentum rates, number of bin (i.e. class) for each output unit, data file name
Load all input/output pairs (x_i, t_i) from data file
First run?
Yes
No
Initialize random weights (v_{ij} and w_{jk})
Read last weight file
Feed forward
Loop for all training patterns
tsme < limit
Yes
No
Loop until meet specified iteration
Back propagate error and introduce momentum rate (λ)
Update weights
Save weight file
Loop for all data
Feed forward
Calculate performance (% of training and testing correct)
Stop

Figure 4.2 : Flow chart of Neural Network Simulator

4.4 Neural Network Training and Testing

Training and testing the network will start immediately prior to the completion of the simulator development and data preparation. The training will follows exactly to the designed algorithm of the simulator.

First of all, several parameters need to be specified. Beside the total number of input patterns, number of bin for every class and the network configuration (input, hidden and output units), the major parameters used in supervised training have to do with how the error is computed and how big a step to take when adjusting the connection weights in achieving the desired output.

In neural network, this step is represented by a learning rate. By selecting a slower learning rate, the chances of getting a good generalization are very high (Bigus, 1996). So, the learning rate of 0.1 was specified for the whole training exercises. If large learning rate is used, learning becomes unstable, where the network oscillates back and forth across the desirable value (Fausett, 1996). On the other hand, if learning rate is too small, learning process will take longer time. Therefore, the momentum rate was introduced as another training parameter to speed up the learning process. This parameter will give some momentum to the weight change in the next iteration and thus overcome the above limitations. The momentum rate of 0.5 was introduced during the training process.

During the training, all the training patterns will be feed into the feed forward algorithm. In this stage, as mentioned earlier in the design of the neural network architecture, the binary sigmoid activation function had been applied in the training algorithm rather than other functions since the values in the data set are between 0 and 1. If the computed total mean square error (tmse) less than the specified tolerance (limit), then the connection weights were adjusted by back propagating the error and updated accordingly.

Stopping criteria allows the neural network enough time to make significant adjustments. It also can be used to avoid over-training, where the network is in the situation of memorizing the patterns rather than learning the features of the problem. Therefore, two stopping criteria were set in the designed neural network simulator, which can be specified by the users. In this project, they were set as:

i) Total mean square error (tmse) less than 0.0001. Training will stop when the tmse reaches this value.

ii) Total of iteration of 100. Training will also stop after completing the last iteration.

If the specified iterations are too big, the network may be in over-training situation. Where as, with too small iterations, the network may be in under-training problem. It will not has enough knowledge to solve the problem. In order to overcome these problem, a little features has been attached to the simulator (reload the latest weight file). So, the users can continue the training if the network is still in under-training situation. In this project, the training was carried out in this manner, where the performance of the network was checked after every 100 iterations. The training is only stop when the test result starts to decrease.

After meting the stopping conditions, the performance of the network will be calculated. Once again, the all training and testing data sets will be feed into the feed forward algorithm in order to perform the classification of the outputs. From there, the percentage of correct classifications is calculated.

4.5 The development of Concrete Mix Design System

The program of the Concrete Mix Design System is more simple compared to the Neural Network Simulator. Even though it is simple, it has to perform two functions.

1. To predict the amounts of 4 mixing materials in concrete based on the specified strength and the other 6 factors that can affect the concrete mixes.
2. To find the concrete strength based on the amounts of 4 mixing materials and the above 6 factors, which more like reversing the first function.

In working of the second function, another network with different configuration of nodes has to be trained. The training and testing procedures are still the same as the above network. After this second network has been trained and tested, its weights file will kept separately from the first one and ready to be loaded into the Concrete Mix Design System. The flowchart of the system is shown in Figure 4.3 and the program's codes and the user's manual of the system are listed in Appendix C and Appendix D respectively.

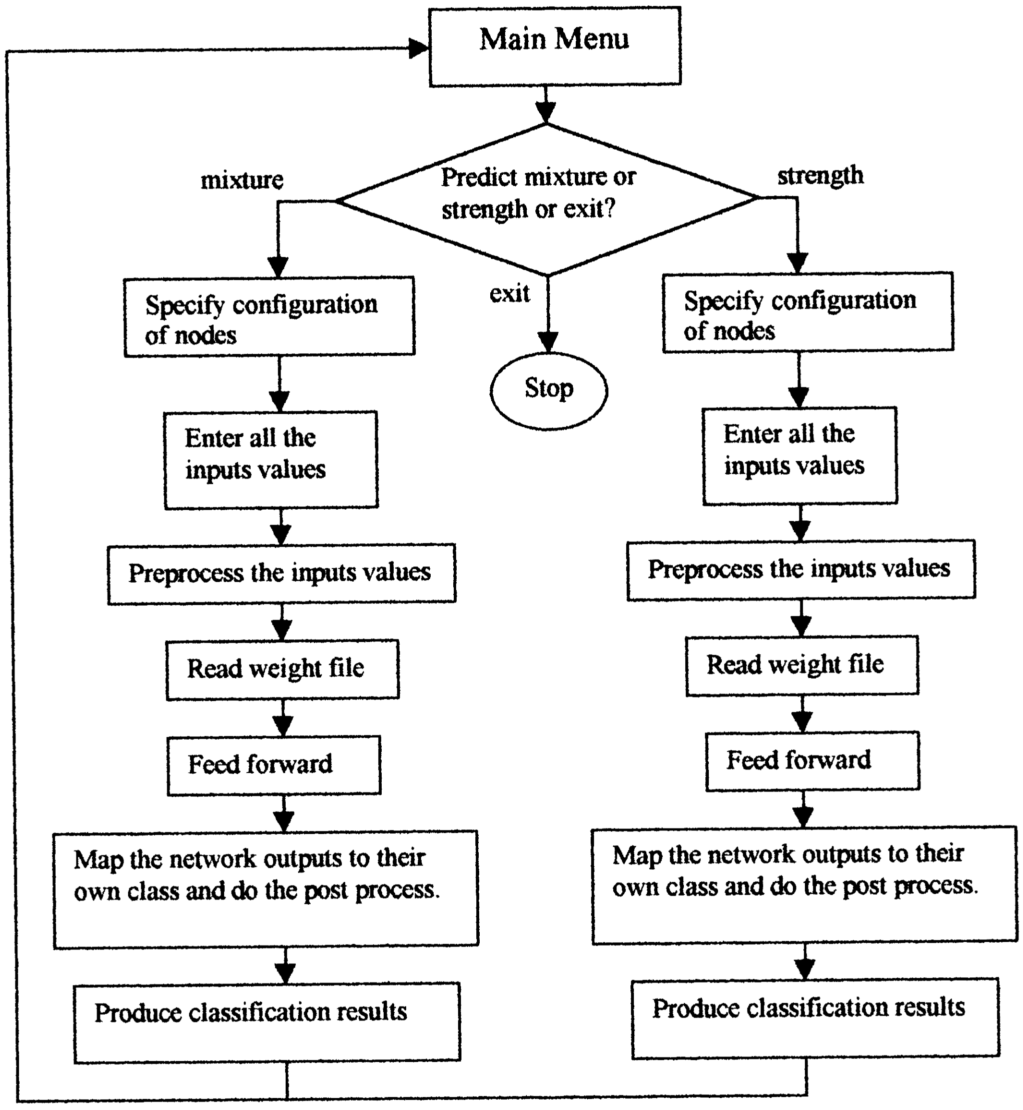

Main Menu
Predict mixture or strength or exit?
mixture
strength
exit
Stop
Specify configuration of nodes
Enter all the inputs values
Preprocess the inputs values
Read weight file
Feed forward
Map the network outputs to their own class and do the post process.
Produce classification results
Specify configuration of nodes
Enter all the inputs values
Preprocess the inputs values
Read weight file
Feed forward
Map the network outputs to their own class and do the post process.
Produce classification results

Figure 4.3 : Flowchart of Concrete Design System

CHAPTER 5

RESULTS AND DISCUSSIONS

The training and testing procedures are followed according to description in Section 4.3. A total of 4080 input/output patterns of concrete mix design were used to train and test the network. 80% of the total patterns were used for training and the other 20% were used for testing the performance of the network.

Table 5.1 shows the results of the training and testing of the network to predict concrete mix with different number of hidden units after 100 iterations. The selection the best network configuration will based on which hidden unit gives the highest testing result. If several hidden units produced same testing result, then the hidden unit with the lowest training result will be selected.

Table 5.1: Selection of the hidden units in the network to predict concrete mix

No. of HU	% Correct	Output 1 Water	Output 2 Cement	Output 3 Fine agg.	Output 4 Course agg.	Average
5	Training	73.34	46.84	56.43	40.17	54.20
	Testing	72.91	46.93	57.96	43.14	55.24
10	Training	81.43	59.65	64.58	48.96	63.66
	Testing	82.60	56.49	66.05	51.23	64.09
15	Training	90.75	65.29	67.86	55.20	69.78
	Testing	90.20	63.24	67.65	55.76	69.21
20	Training	86.18	62.41	68.60	57.93	68.78
	Testing	86.25	62.50	68.34	57.84	68.73
25	Training	81.34	61.82	70.28	56.62	67.52
	Testing	82.35	59.80	72.79	56.13	67.77
30	Training	82.13	60.97	69.52	54.87	66.87
	Testing	83.95	60.42	71.57	57.35	68.32

The hidden unit 15 was selected since it produced the best testing result (69.21% of correct classification).

Table 5.2: Selection of training data set for the network to predict concrete mix

HU 15	% Correct	Output 1	Output 2	Output 3	Output 4	Average
Data set 1	Training	90.75	65.29	67.86	55.20	69.78
	Testing	90.20	63.24	67.65	55.76	69.21
Data set 2	Training	90.87	67.95	66.76	62.75	72.08
	Testing	90.81	68.26	66.54	61.52	71.78
Data set 3	Training	90.69	67.37	65.29	60.81	71.04
	Testing	90.69	66.54	66.79	60.78	71.20

Table 5.2 shows that Data set 2 is giving the highest testing result (71.78% of correct classification). Therefore, Data set 2 is used for further training of the 15 hidden units' network.

As indicated by the results in Table 5.3, the best performance of the network was obtained after 1600 iterations, which gives 83.7% correct of testing patterns. After 1700 iterations, the testing result starts to drop even though the training result becomes better. This situation of what we called over-training was happen here, where the network starts to memorize and not generalize.

Table 5.3: The best performance of the network in predicting the concrete mix

No. of iteration	% Correct	Output 1	Output 2	Output 3	Output 4	Average
100	Training	90.87	67.95	66.76	62.75	72.08
	Testing	90.81	68.26	66.54	61.52	71.78
200	Training	91.67	67.31	71.02	62.59	73.15
	Testing	91.67	66.42	72.43	61.76	73.07
300	Training	93.47	71.11	75.12	64.95	76.16
	Testing	93.01	69.85	76.59	64.22	75.92
400	Training	94.57	73.68	77.42	66.33	78.00
	Testing	94.36	72.30	76.72	64.46	76.96
500	Training	94.94	75.31	77.91	68.08	79.06
	Testing	94.61	74.51	76.72	64.58	77.61
600	Training	95.34	76.65	77.97	68.72	79.67
	Testing	95.22	76.47	77.45	66.67	78.95
700	Training	99.17	77.79	77.94	69.94	81.21
	Testing	99.26	77.57	78.06	68.50	80.85
800	Training	100.00	78.40	78.16	70.53	81.77
	Testing	100.00	78.92	78.92	69.12	81.74
900	Training	100.00	79.01	78.46	70.86	82.08
	Testing	100.00	78.80	80.15	70.83	82.45
1000	Training	100.00	79.23	78.83	70.65	82.18
	Testing	100.00	79.78	80.88	70.47	82.78
1100	Training	100.00	79.44	79.04	70.80	82.32
	Testing	100.00	80.27	81.13	71.20	83.15
1200	Training	100.00	79.69	79.38	70.89	82.49
	Testing	100.00	80.51	81.37	70.96	83.21
1300	Training	100.00	80.15	79.60	70.80	82.64
	Testing	100.00	80.51	81.62	71.08	83.30
1400	Training	100.00	80.79	79.81	70.96	82.89
	Testing	100.00	80.64	81.37	71.32	83.33
1500	Training	100.00	81.25	80.18	71.17	83.15
	Testing	100.00	81.00	81.62	71.45	83.52
1600	Training	100.00	81.86	80.58	71.32	83.44
	Testing	100.00	81.62	81.74	71.45	83.70
1700	Training	100.00	82.35	80.63	71.51	83.62
	Testing	100.00	81.25	81.74	71.69	83.67

The training and testing the network for predicting the concrete strength are similar to the network for predicting the concrete mix. Table 5.4 shows the results of the training and testing of the network to predict concrete strength with different number of hidden units after 100 iterations.

Table 5.4: Selection of the hidden units in the network to predict concrete strength

HU	Training	Testing
1	87.71	87.38
2	93.08	92.65
3	95.53	94.98
4	93.63	92.89
5	95.96	95.10
6	95.19	94.49
7	95.68	94.97
8	95.68	94.61
9	93.99	93.38
10	94.91	93.99
11	93.75	93.14
12	93.19	91.91

The hidden unit 5 was selected since it produced the best testing result (95.10% of correct classification).

Table 5.5: Selection of training data set for the network to predict concrete strength

HU 5	Training	Testing
Data set 1	95.96	95.10
Data set 2	95.68	94.79
Data set 3	95.22	96.08

Table 5.5 shows that Data set 3 is giving the highest testing result (96.08% of correct classification). Therefore, Data set 3 is used for further training of the 5 hidden units' network.

Table 5.6: The best performance of the network in predicting the concrete strength

No. of iteration	Training	Testing
100	95.22	96.08
200	98.35	98.28
300	98.86	99.02
400	99.14	99.39
500	99.20	99.51
600	99.35	99.51
700	99.54	99.51
800	99.75	100.00
900	99.97	100.00
1000	100.00	100.00
1100	100.00	100.00
1200	100.00	100.00

From the results in Table 5.6, the network model with 5 hidden units was selected as the best model, which gives the ultimate performance of 100% after 1000 iterations.

The final neural network models for predicting the concrete mix and the concrete strength are shown in Table 5.7 and Table 5.8 respectively.

Table 5.7: Final Neural Network Model for Prediction of Concrete Mix

Architecture	Back Propagation Network
Number of Input Units	7
Number of Hidden Units	15
Number of Output Units	4
Activation Function	Binary Sigmoid
Learning Rate	0.1
Momentum Rate	0.5
Training Pattern	Data set 2
Iterations	1700

Table 5.8: Final Neural Network Model for Prediction of Concrete Strength

Architecture	Back Propagation Network
Number of Input Units	10
Number of Hidden Units	5
Number of Output Units	1
Activation Function	Binary Sigmoid
Learning Rate	0.1
Momentum Rate	0.5
Training Pattern	Data set 3
Iterations	1000

CHAPTER 6

CONCLUSION AND RECOMMENDATION

This project proves the capability and reliability of neural network model in performing the design of concrete mixes. The performance of the developed models with the accuracy of 83.70% and 100% respectively, are satisfactory. The developed neural network simulator by using the back propagation architecture has demonstrated its ability in training the given input/output patterns.

Future study should look into the following:

a) Other neural network models such as Radial Basis Functions (RBF) and Probabilistic Neural Network (PNN).

b) Integrating neural network with fuzzy logic.

c) Consider other factor such as method of preparation.

d) Using real field data – produce more meaningful model.

Bibliography

Abdullah, C.S., Abu Bakar, A. and Yusuf, Y. (2000), *Neural Network Modelling for In-Situ Concrete Strength*, , Procedings of the 4th Asia-Pacific Structural Engineering and Construction Conference, Kuala Lumpur, pages 223-236.

ACI 211.1-91 (1994), *Standard practice for selecting proportions for normal, heavyweight, and mass concrete*, ACI Manual of Concrete Practice, Part 1: Materials and General Properties of Concrete, Detroit, Michigan.

Amini, F., Chen, H.M., Qi, G.Z. and Yang, C.S. (1997), *Generalized Neural Network Based Model for Structural Dynamic Identification, Analytical and Experimental Stidies*, in Proceeding of the 1997 IASTED International Conference on Intelligent Information System (IIS '97) http://www.computer.org/procedings/iis/8218/82180138abs.htm

Awad, E.M., (1996), *Building Expert System – Principles, Procedures, and Applications*, West Publishing.

Bigus, J.P. (1996), *Data Mining with Neural networks: Solving Business Problems - from application development to decision support*, McGraw Hill, New York

Che Wan Putra, C.W.F, Abd Majid, M.Z. and Kasim, N. (2000), *Integrating Construction Scheduling and Knowledge-based Systems for Generating Material Scheduling*, Procedings of the 4th Asia-Pacific Structural Engineering and Construction Conference, Kuala Lumpur, pages 51-57.

Department of the Environment, (1988), *Design of Normal Concrete Mixes*, Building Research Establishment, U.K.

Freeman, J. A. & Skapula, D. M. (1992). *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley Publishing Company, New York.

Fausett, L. (1994), *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, New Jersey: Prentice Hall

Gambhir, M.L. (1995), *Concrete Technology*, McGraw-Hill, New York.

Hadi, M.N.S. (2000), *Optimum Design of Reinforced Concrete T Beams by Genetic Algorithms*, , Procedings of the 4th Asia-Pacific Structural Engineering and Construction Conference, Kuala Lumpur, pages 161-168.

Hegazy, T., Tully, S., and Marzouk, H. (1998), *A neural network approach for predicting the structural behavior of concrete slabs*, Canadian Journal of Civil Engineering, Volume 25, Number 4, pages 668-677

Hsu, K.L., Gupta, H.V. and Sorooshian, S. (1993), *Artificial Neural Network Modeling Of The Rainfall-Runoff Process*, Water Resources Research, 29 (4), pages 1185-1194.

Hsu, D.S. and Tsai, C.H. (1997), *Reinforced concrete structural damage diagnosis by using artificial neural network*, in Proceeding of the 1997 IASTED International Conference on Intelligent Information System (IIS '97)
http://www.computer.org/procedings/iis/8218/82180149abs.htm

Hung, S.L. and Jan, J.C. (1997), *Machine learning in engineering design- an unsupervised fuzzy neural network case-based learning model*, in Proceeding of the 1997 IASTED International Conference on Intelligent Information System (IIS '97)

Luger, G.F. and Stubblefield, W.A. (1998), *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley.

McCulloch, W.S. and Pitts, W, *A logical calculus of the ideas imminent in nervous activity*. In Schalkoff, R.J., editor, *Artificial Neural Network*, pages 74-75, McGraw-Hill.

Neville, A.M. (1995), *Properties of Concrete*, Longman, England.

NSERC, (1996), *Optimization of Concrete Mix Performance/Production*, Alberta Construction Industry Research Chair, University of Alberta.
http://cem.civil.ualberta.ca/Research/concrete-mix-design-summary.htm

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1996), *Learning internal representations by error propagation*. In Bigus J.P., editor, *Data Mining with Neural Network*, page 69, McGraw-Hill

Schalkoff, R.J. (1997), *Artificial Neural Network*, McGraw-Hill, New York.

Sidney, M. and Young, J.F. (1981), *Concrete*, Prentice Hall.

Simon, M.J., Lagergreen, E.S. and Sayder, K.S. (1997), *Concrete Mixture Optimization using Statistical Mixture Design Methods*, Procedings of the PCI/FHWA International Symposium on High Performance Concrete, New Orleans, Luisiana. Pages 230-244.

Simon, M.J., Sayder, K.S., and Frohnsdorff, G. (1999), *Concrete Durability and Repair Technology Conference*, Uni. Of Dundee, UK.

Ural, D.N. and Saka, H. (1998), *Liquefaction Assessment by Artificial Neural Networks*, in electronic journal of geotechnical engineering.
http://geotech.civen.okstate.edu/ejge/ppr9803/

Welstead, S.T., (1994), *Neural Network and Fuzzy Logic Applications in C/C++*, John Wiley and Sons.

Zhao, Z. and Fan, S.C. (2000), *An Expert System for Box Girder Bridge Design,* Procedings of the 4th Asia-Pacific Structural Engineering and Construction Conference, Kuala Lumpur, pages 75-81.

Appendix A

Sample of Raw Data

Appendix A Sample of Raw Data

Item	1.1			1.2		1.4	1.5	1.6		1.7	2.1	2.2	2.3	3.1	4.1	4.2	4.3	5.1	5.2	5.3	5.4
Ref/Cal	Specified	Specified	Fig.1	Fig.3	C1/specified	C2	Specified	Specified	Specified	Tab.2, Fig.2	Specified	Specified	Tab.3	C3	Specified	Fig.5	C4	% passing	Fig.6	C5	C5
Description	Characteristic strength	Proportion defective	k	Standard deviation	Margin	Target mean strength	Cement type	Course aggregate	fine aggregate	Free water/ cement ratio	Slump	Maximum agg. Size	Free water content	Cement content	Rel. density of agg.	Concrete density	Tot. agg. content	Grading of fine agg.	Proportion fine agg.	Fine agg. content	Course agg. content
1	25	1	2.33	8	18.64	43.64	OPC	Uncrushed	Uncrushed	0.48	0-10	10	150	313	2.6	2410	1948	15	55	1071	876
2	45	2.5	1.96	8	15.68	60.68	OPC	Crushed	Crushed	0.40	10-30	40	175	438	2.6	2380	1768	40	36	636	1131
3	25	1	2.33	8	18.64	43.64	OPC	Uncrushed	Uncrushed	0.48	10-30	10	180	375	2.6	2375	1820	15	57	1037	783
4	25	1	2.33	8	18.64	43.64	OPC	Uncrushed	Uncrushed	0.48	0-10	40	115	240	2.6	2460	2105	100	16	337	1769
5	50	10	1.28	8	10.24	60.24	OPC	Uncrushed	Uncrushed	0.37	60-180	40	175	473	2.6	2400	1752	40	36	631	1121
6	60	1	2.33	8	18.64	78.64	OPC	Crushed	Crushed	0.30	60-180	40	205	683	2.6	2340	1452	15	44	639	813
7	30	5	1.64	8	13.12	43.12	OPC	Crushed	Crushed	0.55	0-10	10	180	327	2.6	2370	1863	60	37	689	1174
8	50	5	1.64	8	13.12	63.12	OPC	Uncrushed	Uncrushed	0.35	0-10	40	115	329	2.6	2460	2016	100	14	282	1734
9	25	10	1.28	4	5.12	30.12	OPC	Crushed	Crushed	0.69	0-10	40	155	225	2.6	2405	2025	100	18	365	1661
10	60	5	1.64	4	6.56	66.56	OPC	Crushed	Crushed	0.37	0-10	20	170	459	2.6	2390	1761	100	19	335	1426
11	60	10	1.28	8	10.24	70.24	OPC	Uncrushed	Uncrushed	0.31	30-60	20	180	581	2.6	2375	1614	100	21	339	1275
12	50	5	1.64	4	6.56	56.56	OPC	Crushed	Crushed	0.44	0-10	40	155	352	2.6	2405	1898	80	19	361	1537
13	25	5	1.64	4	6.56	31.56	OPC	Crushed	Crushed	0.66	10-30	10	205	311	2.6	2340	1824	60	41	748	1076
14	65	1	2.33	4	9.32	74.32	OPC	Crushed	Crushed	0.32	0-10	10	180	563	2.6	2370	1628	60	34	553	1074
15	30	1	2.33	8	18.64	48.64	OPC	Crushed	Crushed	0.50	30-60	20	210	420	2.6	2310	1680	40	39	655	1025
16	50	5	1.64	4	6.56	56.56	OPC	Uncrushed	Uncrushed	0.39	30-60	40	160	410	2.6	2400	1830	100	19	348	1482
17	65	10	1.28	8	10.24	75.24	OPC	Crushed	Crushed	0.32	0-10	40	155	484	2.6	2405	1766	40	24	424	1342
18	40	10	1.28	4	5.12	45.12	OPC	Crushed	Crushed	0.53	30-60	10	230	434	2.6	2340	1676	60	41	687	989
19	50	10	1.28	8	10.24	60.24	OPC	Crushed	Crushed	0.42	60-180	40	205	488	2.6	2340	1647	100	22	362	1285
20	25	1	2.33	8	18.64	43.64	OPC	Uncrushed	Uncrushed	0.48	60-180	40	175	365	2.6	2400	1860	60	32	595	1265
21	70	10	1.28	4	5.12	75.12	OPC	Crushed	Crushed	0.32	30-60	10	230	719	2.6	2340	1391	40	45	626	765
22	40	1	2.33	4	9.32	49.32	OPC	Uncrushed	Uncrushed	0.45	10-30	10	180	400	2.6	2375	1795	80	32	574	1221
23	25	10	1.28	4	5.12	30.12	OPC	Uncrushed	Uncrushed	0.62	30-60	10	205	331	2.6	2340	1804	15	66	1191	613
24	60	1	2.33	4	9.32	69.32	OPC	Uncrushed	Uncrushed	0.32	0-10	10	150	469	2.6	2410	1791	40	40	717	1075
25	25	10	1.28	8	10.24	35.24	OPC	Crushed	Crushed	0.62	0-10	40	155	250	2.6	2405	2000	100	17	340	1660
26	25	5	1.64	8	13.12	38.12	OPC	Uncrushed	Uncrushed	0.54	0-10	20	135	250	2.6	2440	2055	100	21	432	1623
27	65	10	1.28	4	5.12	70.12	OPC	Crushed	Crushed	0.35	0-10	10	180	514	2.6	2370	1676	100	24	402	1274
28	30	1	2.33	4	9.32	39.32	OPC	Uncrushed	Uncrushed	0.53	30-60	10	205	387	2.6	2340	1748	80	34	594	1154
29	60	5	1.64	4	6.56	66.56	OPC	Crushed	Crushed	0.37	0-10	20	170	459	2.6	2390	1761	80	22	387	1373
30	50	5	1.64	4	6.56	56.56	OPC	Uncrushed	Uncrushed	0.39	30-60	10	205	526	2.6	2340	1609	100	28	451	1159
31	50	1	2.33	8	18.64	68.64	OPC	Crushed	Crushed	0.36	60-180	40	205	569	2.6	2340	1566	15	45	705	861
32	30	5	1.64	4	6.56	36.56	OPC	Crushed	Crushed	0.60	10-30	10	206	342	2.6	2340	1793	80	32	574	1219
33	40	5	1.64	4	6.56	46.56	OPC	Uncrushed	Uncrushed	0.46	60-180	10	225	489	2.6	2340	1626	60	44	715	910
34	25	5	1.64	8	13.12	38.12	OPC	Uncrushed	Uncrushed	0.54	10-30	10	180	333	2.6	2375	1862	80	33	614	1247
35	25	10	1.28	4	5.12	30.12	OPC	Crushed	Crushed	0.69	0-10	20	170	246	2.6	2390	1974	80	26	513	1460
36	50	10	1.28	8	10.24	60.24	OPC	Crushed	Crushed	0.42	0-10	10	180	429	2.6	2370	1761	80	29	511	1251
37	50	1	2.33	4	9.32	59.32	OPC	Crushed	Crushed	0.43	10-30	20	190	442	2.6	2360	1728	60	28	484	1244
38	30	1	2.33	8	18.64	48.64	OPC	Uncrushed	Uncrushed	0.45	10-30	40	140	311	2.6	2435	1984	40	30	595	1389
39	25	5	1.64	4	6.56	31.56	OPC	Crushed	Crushed	0.66	60-180	20	225	341	2.6	2320	1754	15	61	1070	684
40	40	1	2.33	4	9.32	49.32	OPC	Uncrushed	Uncrushed	0.45	60-180	20	195	433	2.6	2375	1747	15	55	961	786
41	50	5	1.64	4	6.56	56.56	OPC	Uncrushed	Uncrushed	0.39	30-60	20	180	462	2.6	2375	1733	15	46	797	936
42	50	5	1.64	4	6.56	56.56	OPC	Crushed	Crushed	0.44	30-60	10	230	523	2.6	2340	1587	100	29	460	1127
43	25	1	2.33	8	18.64	43.64	OPC	Uncrushed	Uncrushed	0.48	0-10	20	135	281	2.6	2440	2024	60	28	567	1457
44	40	5	1.64	8	13.12	53.12	OPC	Uncrushed	Uncrushed	0.42	0-10	20	135	321	2.6	2440	1984	60	27	536	1448
45	40	1	2.33	8	18.64	58.64	OPC	Uncrushed	Uncrushed	0.38	10-30	40	140	368	2.6	2435	1927	80	19	366	1561
46	25	1	2.33	4	9.32	34.32	OPC	Uncrushed	Uncrushed	0.58	10-30	10	180	310	2.6	2375	1885	15	61	1150	735
47	25	5	1.64	4	6.56	31.56	OPC	Crushed	Crushed	0.66	0-10	40	155	235	2.6	2405	2015	15	40	806	1209
48	60	10	1.28	4	5.12	65.12	OPC	Crushed	Crushed	0.38	0-10	10	180	474	2.6	2370	1716	40	41	704	1013
49	40	5	1.64	4	6.56	46.56	OPC	Uncrushed	Uncrushed	0.46	60-180	10	225	489	2.6	2320	1606	15	68	1092	514
50	50	10	1.28	8	10.24	60.24	OPC	Crushed	Crushed	0.42	30-60	40	190	452	2.6	2360	1718	100	19	326	1391
51	65	10	1.28	4	5.12	70.12	OPC	Crushed	Crushed	0.35	30-60	20	210	600	2.6	2310	1500	40	35	525	975
52	40	10	1.28	4	5.12	45.12	OPC	Crushed	Crushed	0.53	0-10	40	155	292	2.6	2405	1958	80	20	392	1566
53	25	1	2.33	4	9.32	34.32	OPC	Crushed	Crushed	0.63	0-10	20	170	270	2.6	2390	1950	60	30	585	1365
54	25	5	1.64	8	13.12	38.12	OPC	Uncrushed	Uncrushed	0.54	30-60	10	205	380	2.6	2340	1756	40	50	878	878
55	65	10	1.28	8	10.24	75.24	OPC	Crushed	Crushed	0.32	30-60	40	190	594	2.6	2360	1576	15	37	583	993
56	65	10	1.28	8	10.24	75.24	OPC	Crushed	Crushed	0.32	60-180	10	250	781	2.6	2290	1259	80	35	441	818

Appendix B

Program Codes of Neural Network Simulator

Appendix B Program Codes of Neural Network Simulator

```
Private Sub Command1_Click()

'Declaration of Arrays
Dim ref_no(), X(), T(), V(), W(), Zin(), Z(), Yin(), Y()
Dim delta_V(), delta_W(), del_V(), del_W(), mom_V(), mom_W()
Dim del_in(), Ep(), bin(), acc()
Dim timestart As Date
Dim timestop

'*****************************************************************

   numdata = 4080  'total number of data
   n = 7        'number of input
   p = 15        'number of hidden unit
   m = 4        'number of output

   numtraindata = Int(numdata * 0.8)  '80% of training data

   iter = 100    'no. of iteration or epoch
   alpha = 0.1    'learning rate
   lamda = 0.5    'momentum rate

 'Redimension of Arrays
 ReDim X(numdata, n + 1), T(numdata, m), Y(numdata, m), Ep(m), bin(m),
 acc(m), ref_no(numdata)

   bin(1) = 28   'no. of class in Output 1 (i.e. water content)
   bin(2) = 65   'no. of class in Output 2 (i.e. cement content)
   bin(3) = 52   'no. of class in Output 3 (i.e. fine aggregate)
   bin(4) = 70   'no. of class in Output 4 (i.e. course aggregate)

   limit = 0.0001  'stopping criteria

   first_run = 1  '1 for new run : 2 for next run

   best_wt_file$ = "c:/CDS/wttest.txt"      'best weight filename
   data_file$ = "c:/CDS/data_mix.txt"      'data set filename
   act_func = 1   '1 for binary sigmoid  :  2 for bipolar sigmoid


   For i = 1 To m
     acc(i) = 1 / bin(i) 'max. value of every classes in every outputs
   Next i

'Load input/output patterns
   Open data_file$ For Input As #1     '

     For ndata = 1 To numdata
          Input #1, ref_no(ndata) 'reference no. of data
       For i = 1 To n
          Input #1, X(ndata, i)   'input pattren
       Next i
       For k = 1 To m
          Input #1, T(ndata, k)   'target or output pattern
       Next k
     Next ndata
   Close #1

'Redimension of Arrays
ReDim V(n + 1, p), W(p + 1, m), Zin(p), Z(p)
ReDim Yin(m), delta_W(p + 1, m), del_W(m), del_V(p), del_in(p)
ReDim delta_V(n + 1, p), mom_V(n + 1, p), mom_W(p + 1, m)
ReDim count_train(m), count_test(m), per_test_correct(m), per_train_correct(m)

'**********************************************************
   If first_run = 1 Then

'Initialize weights (bet. -0.5 to 0.5)
      For j = 1 To p
        For i = 0 To n
        V(i, j) = Rnd - 0.5     'weight from input units to HU
        Next i
      Next j

      For k = 1 To m
        For j = 0 To p
        W(j, k) = Rnd - 0.5     'weight from HU to output units
        Next j
      Next k

 'Save initial weights as last weights file
 Open "c:/a_tesis/weight.txt" For Output As #1

      For k = 1 To m
        For j = 0 To p
        Write #1, W(j, k)
        Next j
      Next k

         For j = 1 To p
            For i = 0 To n
            Write #1, V(i, j)
            Next i
         Next j
Close #1

   End If  ' end of if first run = 1
'***********************************************************

'Load the last weights file
Open "c:/a_tesis/weight.txt" For Input As #1

       For k = 1 To m
          For j = 0 To p
          Input #1, W(j, k)
          Next j
       Next k

       For j = 1 To p
          For i = 0 To n
          Input #1, V(i, j)
          Next i
       Next j
 Close #1
'***********************************************************

 timestart = Now     'set training and testing time

 For id = 1 To iter      'iteration or epoch loop


 'Initialize momentum term
    For j = 1 To p
       For i = 0 To n
       mom_V(i, j) = 0
       Next i
    Next j

    For k = 1 To m
       For j = 0 To p
       mom_W(j, k) = 0
       Next j
    Next k


    For k = 1 To m
       Ep(k) = 0   'initial value for square error
 Next k

 Total_Ep = 0   'initial value for total square error

'**********************************************************

 For ct = 1 To numtraindata  'loop for training data set

 Form1.Caption = "ITER " & id & " Record no.: " & ct 'just to monitor the
training progress

   For j = 1 To p
   del_in(j) = 0   'initial value of delta input of hidden unit
   Next j

'============
'Feed-forward
'============


  'CALCULATE Y(ct,k) - network output signal

  'Calculate input signals to each hidden unit.
       For j = 1 To p
         Zin(j) = V(0, j)
         For i = 1 To n
            Zin(j) = Zin(j) + X(ct, i) * V(i, j)
         Next i

  'Apply act. fn. to compute output signal
  If act_func = 1 Then Z(j) = 1 / (1 + Exp(-Zin(j)))       'if using binary sigmoid
  If act_func = 2 Then Z(j) = (2 / (1 + Exp(-Zin(j)))) - 1  'if using bipolar sigmoid
       Next j

  'Calculate input signals from HU to output units
       For k = 1 To m
          Yin(k) = W(0, k)
          For j = 1 To p
             Yin(k) = Yin(k) + Z(j) * W(j, k)
          Next j

  'Apply act. fn. to compute output signal
  If act_func = 1 Then Y(ct, k) = 1 / (1 + Exp(-Yin(k)))        'if using binary
 sigmoid
  If act_func = 2 Then Y(ct, k) = (2 / (1 + Exp(-Yin(k)))) - 1   'if using bipolar
 sigmoid

        Ep(k) = Ep(k) + 0.5 * (T(ct, k) - Y(ct, k)) * (T(ct, k) - Y(ct, k)) 'total
 square error
        Total_Ep = Total_Ep + Ep(k)
        Next k

'=========================
'Backpropagation Error
'=========================
   'Each output unit receives a target pattern corresponding
   'to the input training pattern.

   'Compute error information term
         For k = 1 To m
      If act_func = 1 Then del_W(k) = (T(ct, k) - Y(ct, k)) * (Y(ct, k) - (Y(ct, k)) *
(Y(ct, k)))
      If act_func = 2 Then del_W(k) = (T(ct, k) - Y(ct, k)) * 0.5 * (1 + Y(ct, k)) *
(1 - Y(ct, k))

            'Calculate its weight correction term
            For j = 1 To p
               delta_W(j, k) = alpha * del_W(k) * Z(j) + lamda * mom_W(j, k)
               mom_W(j, k) = delta_W(j, k)
            Next j

            'Calculate bias correction term
               delta_W(0, k) = alpha * del_W(k) + lamda * mom_W(0, k)
               mom_W(0, k) = delta_W(0, k)
         Next k

    'Each HU, sums its delta inputs (from units in the layer above)
         For j = 1 To p
            For k = 1 To m
               del_in(j) = del_in(j) + del_W(k) * W(j, k)
            Next k

      'Multiply derivative of its act.fn. to calculate error info. term
      If act_func = 1 Then del_V(j) = del_in(j) * (Z(j) - (Z(j)) * (Z(j)))
      If act_func = 2 Then del_V(j) = del_in(j) * 0.5 * (1 + Z(j)) * (1 - (Z(j)))
         Next j

           'Calculate its weight correction term
           For j = 1 To p
              For i = 1 To n
                 delta_V(i, j) = alpha * del_V(j) * X(ct, i) + lamda * mom_V(i, j)
                 mom_V(i, j) = delta_V(i, j)
              Next i
                 delta_V(0, j) = alpha * del_V(j) + lamda * mom_V(0, j)
                 mom_V(0, j) = delta_V(0, j)
           Next j

   '=========================
   'Update weights and bias
   '=========================
        For k = 1 To m
        For j = 0 To p
        W(j, k) = W(j, k) + delta_W(j, k)
        Next j
        Next k

        For j = 1 To p
        For i = 0 To n
        V(i, j) = V(i, j) + delta_V(i, j)
        Next i
        Next j

 Next ct     'Go to next pattern
 tmse = Total_Ep / m / numtraindata   'Total Mean Square Error
 If tmse < limit Then id = iter      'Stopping condition
'  Debug.Print tmse

  Next id    'Go to next iteration or epoch
'********************************************************************
*

'Save current weights as last weights file
Open "c:/a_tesis/weight.txt" For Output As #1

      For k = 1 To m
        For j = 0 To p
        Write #1, W(j, k)
        Next j
        Next k

        For j = 1 To p
        For i = 0 To n
        Write #1, V(i, j)
        Next i
        Next j
 Close #1
'********************************************************************

 For k = 1 To m
 count_train(k) = 0
 count_test(k) = 0
 Next k

 count_error = 0

    maxx = 0

   For dt = 1 To numdata
```

```
'CALCULATE PERFORMANCE
'=====================
'STEP 5: Calculate input signals to each hidden unit.

        For j = 1 To p
           Zin(j) = V(0, j)
           For i = 1 To n
              Zin(j) = Zin(j) + X(dt, i) * V(i, j)
           Next i

      'Apply act. fn. to compute output signal (binary sigmoid)
       If act_func = 1 Then Z(j) = 1 / (1 + Exp(-Zin(j)))
       If act_func = 2 Then Z(j) = (2 / (1 + Exp(-Zin(j)))) - 1
        Next j

'STEP 6: Calculate input signals from HU to output units
        For k = 1 To m
           Yin(k) = W(0, k)
           For j = 1 To p
              Yin(k) = Yin(k) + Z(j) * W(j, k)
           Next j

      'Apply act. fn. to compute output signal (binary sigmoid)
       If act_func = 1 Then Y(dt, k) = 1 / (1 + Exp(-Yin(k)))
       If act_func = 2 Then Y(dt, k) = (2 / (1 + Exp(-Yin(k)))) - 1

    'To count correct training data
    If dt <= numtraindata Then
       If Abs(T(dt, k) - Y(dt, k)) < acc(k) Then
         count_train(k) = count_train(k) + 1
       End If
    Else
       'To count correct testing data
       If Abs(T(dt, k) - Y(dt, k)) < acc(k) Then
         count_test(k) = count_test(k) + 1
       End If
    End If

    'Calculate absolute error
    per_error = Abs(T(dt, k) - Y(dt, k))
    count_error = count_error + per_error
         Next k


    Next dt

    For k = 1 To m
    per_train_correct(k) = count_train(k) / numtraindata * 100
    per_test_correct(k) = count_test(k) / (numdata - numtraindata) * 100
    Next k
```

```
   Debug.Print "% Training correct"; per_train_correct(1); per_train_correct(2);
per_train_correct(3); per_train_correct(4)
   Debug.Print "% Testing correct "; per_test_correct(1); per_test_correct(2);
per_test_correct(3); per_test_correct(4)

   per_predict = (1 - (count_error / m) / numdata) * 100   'percentage of prediction
based on absolute error
   Debug.Print "% of Prediction   "; per_predict

   timestop = DateDiff("s", timestart, Now)   'stop training and testing time
   Debug.Print "time taken (s) ="; timestop

'*********************************************************
'Save last weights as the best weight file
      Open best_wt_file$ For Output As #1
         For k = 1 To m
         For j = 0 To p
         Write #1, W(j, k)
         Next j
         Next k

         For j = 1 To p
         For i = 0 To n
         Write #1, V(i, j)
         Next i
         Next j
       Close #1

'*********************************************************************
 For i = 1 To 100 ' Loop 3 times.
    Beep    ' Sound a tone just to alert that training complete
 Next i
'*********************************************************************

 End    'end of program

 End Sub

 Private Sub Command2_Click()
 End
 End Sub
```

Appendix C

Program codes of Concrete Design System

```
'************************************
'* Program of Predict Concrete Mix *
'************************************

Private Sub Command1_Click()

Dim X(), T(), V(), W(), Zin(), Z(), Yin(), Y(), bin(), acc()

   n = 7
   p = 25
   m = 4


ReDim X(n), Y(m * 4), bin(m * 4), acc(m * 4)
ReDim V(n + 1, p), W(p + 1, m), Zin(p), Z(p), Yin(m)


X1 = Val(grade)
If X1 >= 25 And X1 <= 70 Then
X(1) = (X1 - 25) / (70 - 25)
Else: MsgBox "Concrete grade only between 25 to 70"
Errors = 1
End If
X2 = Val(defect)
If X2 >= 0 And X2 <= 10 Then
X(2) = X2 / 10
Else: MsgBox "Proportion defective only between 0% to 10%"
Errors = 1
End If
X3 = Val(std)
If X3 >= 4 And X3 <= 8 Then
X(3) = (X3 - 4) / (4)
Else: MsgBox "Standard deviation only between 4 N/mm2 to 8 N/mm2", X1
Errors = 1
End If
X6 = Val(aggsize)
If X6 >= 10 And X6 <= 40 Then
X(6) = X6 / 40
Else: MsgBox "Maximum aggregate size only between 10mm to 40mm", X1
Errors = 1
End If
X7 = Val(passing)
If X7 >= 0 And X7 <= 100 Then
X(7) = X7 / 100
```

```
Else: MsgBox "% of fine aggregate passing 600micron seive only between 0% to
100%", X1
Errors = 1
End If

If Option1.Value Then X(4) = 1
If Option2.Value Then X(4) = 0
If Option1.Value = False And Option2.Value = False Then
MsgBox "Please specify Aggregate type"
Errors = 1
End If

If Option3.Value Then X(5) = 0
If Option4.Value Then X(5) = 0.3333
If Option5.Value Then X(5) = 0.6667
If Option6.Value Then X(5) = 1
If Option3.Value = False And Option4.Value = False And Option5.Value = False
And Option6.Value = False Then
MsgBox "Please specify Slump"
Errors = 1
End If

If Errors = 1 Then
Form3.Hide
Form3.Show
Else
Errors = 0

   bin(1) = 28
   bin(2) = 65
   bin(3) = 52
   bin(4) = 70

   For i = 1 To 4
      acc(i) = 1 / bin(i)
   Next i

 'Load weight file
 '================
 Open "c:/a_tesis/mix25.txt" For Input As #1
 '*****************************************************************

      For k = 1 To m
         For j = 0 To p
         Input #1, W(j, k)
         Next j
         Next k

         For j = 1 To p
         For i = 0 To n
```

```
        Input #1, V(i, j)
        Next i
        Next j
Close #1


'Calculate input signals to each hidden unit.
'=================================================

        For j = 1 To p
          Zin(j) = V(0, j)
          For i = 1 To n
            Zin(j) = Zin(j) + X(i) * V(i, j)
          Next i

      'Apply act. fn. to compute output signal (binary sigmoid)
      Z(j) = 1 / (1 + Exp(-Zin(j)))

        Next j

'Calculate input signals from HU to output units
'====================================================

        For k = 1 To m
          Yin(k) = W(0, k)
          For j = 1 To p
            Yin(k) = Yin(k) + Z(j) * W(j, k)
          Next j

      'Apply act. fn. to compute output signal (binary sigmoid)
       Y(k) = 1 / (1 + Exp(-Yin(k)))


'Classification
'===============


 For c = 1 To bin(k)

   'Output 1
   '========
     If Y(1) < acc(1) * c And k = 1 Then
      wclow = (c - 1) * 5 + 115
      wchig = c * 5 + 115
      Text1 = wclow
      Text2 = wchig
      c = bin(1)
     End If

   'Output 2
   '========
     If Y(2) < acc(2) * c And k = 2 Then
      cementlow = (c - 1) * 10 + 185
      cementhig = c * 10 + 185
      Text3 = cementlow
      Text4 = cementhig
      c = bin(2)
    End If

 'Output 3
 '=========
    If Y(3) < acc(3) * c And k = 3 Then
      sandlow = (c - 1) * 20 + 240
      sandhig = c * 20 + 240
      Text5 = sandlow
      Text6 = sandhig
      c = bin(3)
    End If

 'Output 4
 '=========
    If Y(4) < acc(4) * c And k = 4 Then
      agglow = (c - 1) * 20 + 420
      agghig = c * 20 + 420
      Text7 = agglow
      Text8 = agghig
      c = bin(4)
    End If
    Next c
 Next k

End If


End Sub

Private Sub Command2_Click()
PrintForm
End Sub

Private Sub Command3_Click()
Unload Me
Form1.Show
End Sub



'****************************************
'* Program of Pridict Concrete Strength *
'****************************************

Private Sub Command1_Click()
```

```
PrintForm
End Sub

Private Sub Command2_Click()
Dim X(), T(), V(), W(), Zin(), Z(), Yin(), Y(), bin(), acc()

   n = 10
   p = 5
   m = 1


ReDim X(n), Y(m), bin(m), acc(m)
ReDim V(n + 1, p), W(p + 1, m), Zin(p), Z(p), Yin(m)


X1 = Val(defect)
If X1 >= 0 And X1 <= 10 Then
X(1) = X1 / 10
Else: MsgBox "Proportion defective only between 0% to 10%"
Errors = 1
End If
X2 = Val(std)
If X2 >= 4 And X2 <= 8 Then
X(2) = (X2 - 4) / (4)
Else: MsgBox "Standard deviation only between 4 N/mm2 to 8 N/mm2", X1
Errors = 1
End If
X5 = Val(aggsize)
If X5 >= 10 And X5 <= 40 Then
X(5) = X5 / 40
Else: MsgBox "Maximum aggregate size only between 10mm to 40mm", X1
Errors = 1
End If
X6 = Val(passing)
If X6 >= 0 And X6 <= 100 Then
X(6) = X6 / 100
Else: MsgBox "% of fine aggregate passing 600micron seive only between 0% to
100%", X1
Errors = 1
End If
X7 = Val(wc)
If X7 >= 115 And X7 <= 250 Then
X(7) = (X7 - 115) / (250 - 115)
Else: MsgBox "Water content only between 115kg to 250kg", X1
Errors = 1
End If
X8 = Val(Cement)
If X8 >= 185 And X8 <= 834 Then
X(8) = (X8 - 185.4839) / (833.3333 - 185.4839)
Else: MsgBox "Cement content only between 185kg to 834kg", X1
```

```
Errors = 1
End If
X9 = Val(sand)
If X9 >= 242 And X9 <= 1265 Then
X(9) = (X9 - 242.6667) / (1264.4306 - 242.6667)
Else: MsgBox "Fine aggregate content only between 242kg to 1265kg", X1
Errors = 1
End If
X10 = Val(cagg)
If X10 >= 420 And X10 <= 1809 Then
X(10) = (X10 - 420) / (1808.8 - 420)
Else: MsgBox "Course aggregate content only between 420kg to 1809kg", X1
Errors = 1
End If


If Option1.Value Then X(3) = 1
If Option2.Value Then X(3) = 0
If Option1.Value = False And Option2.Value = False Then
MsgBox "Please specify Aggregate type"
Errors = 1
End If

If Option3.Value Then X(4) = 0
If Option4.Value Then X(4) = 0.3333
If Option5.Value Then X(4) = 0.6667
If Option6.Value Then X(4) = 1
If Option3.Value = False And Option4.Value = False And Option5.Value = False
And Option6.Value = False Then
MsgBox "Please specify Slump"
Errors = 1
End If

If Errors = 1 Then
Form2.Hide
Form2.Show
Else
Errors = 0

   bin(1) = 10

   For i = 1 To m
      acc(i) = 1 / bin(i)
   Next i

'*********************************************************************
'Load weight file
'====================
   Open "c:/a_tesis/fgrade.txt" For Input As #1
```

```
    For k = 1 To m
       For j = 0 To p
       Input #1, W(j, k)
       Next j
       Next k

       For j = 1 To p
       For i = 0 To n
       Input #1, V(i, j)
       Next i
       Next j
Close #1


'Calculate input signals to each hidden unit.
'============================================
        For j = 1 To p
           Zin(j) = V(0, j)
           For i = 1 To n
              Zin(j) = Zin(j) + X(i) * V(i, j)
           Next i

      'Apply act. fn. to compute output signal (binary sigmoid)
       Z(j) = 1 / (1 + Exp(-Zin(j)))
        Next j

'Calculate input signals from HU to output units
'===============================================
        For k = 1 To m
           Yin(k) = W(0, k)
           For j = 1 To p
              Yin(k) = Yin(k) + Z(j) * W(j, k)
           Next j

      'Apply act. fn. to compute output signal (binary sigmoid)
       Y(k) = 1 / (1 + Exp(-Yin(k)))

'Classification
'==============
 For c = 1 To bin(k)
      If Y(1) < acc(1) * c And k = 1 Then
       grade = (c - 1) * 5 + 25
       c = bin(k)
      End If


      Next c

         Next k
End If
End Sub


Private Sub Command3_Click()
Unload Me
Form1.Show

End Sub
```

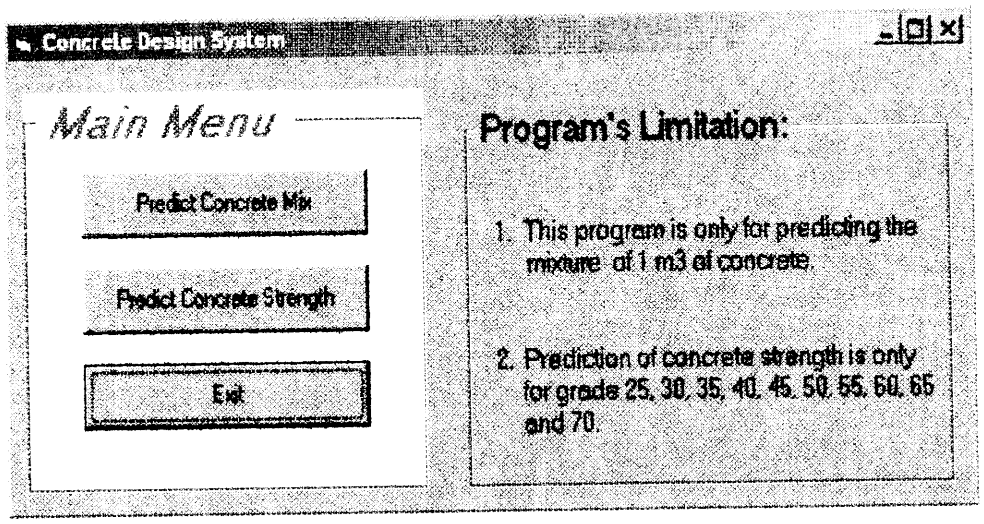
Appendix D

User's Manual

Appendix D User's Manual

1.0 Main Menu of Concrete Design System

- Click Predict Concrete Mix to find the mixes of concrete or
- Click Predict Concrete Strength to find the concrete strength



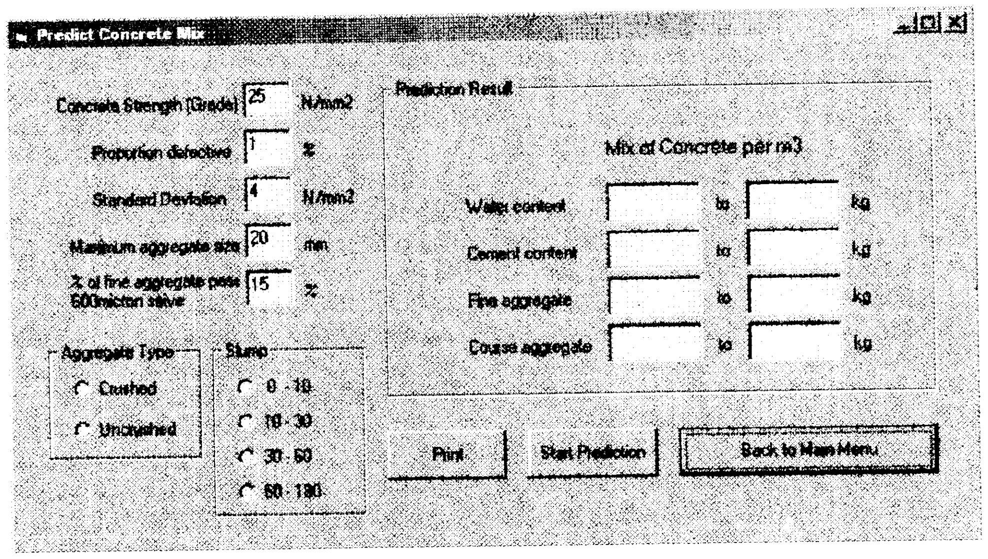
Program's Limitation:

1. This program is only for predicting the mixture of 1 m3 of concrete.

2. Prediction of concrete strength is only for grade 25, 30, 35, 40, 45, 50, 55, 60, 65 and 70.

2.0 Predict Concrete Mix

- The screen below will appeared after you choose "Predict Concrete Mix" button.
- Enter all the parameters required in order to get the prediction result for mix of 1m3 of concrete.
- Then click "Start Prediction".



Concrete Strength (Grade) 25 N/mm2

Proportion defective 1 %

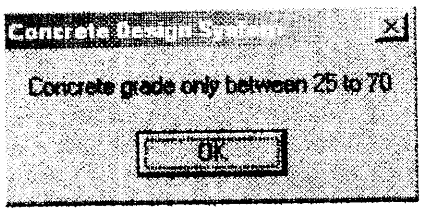
Standard Deviation 4 N/mm2

Maximum aggregate size 20 mm

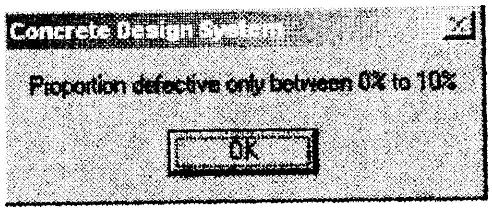
% of fine aggregate pass 600micron sieve 15 %

Water content		to		kg
Cement content		to		kg
Fine aggregate		to		kg
Course aggregate		to		kg

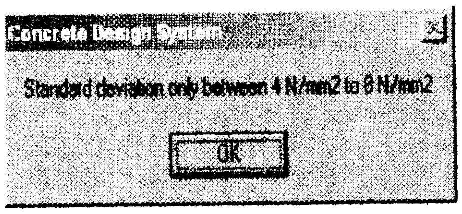
- If the invalid values are entered, the following messages will appeared:


Concrete Design System

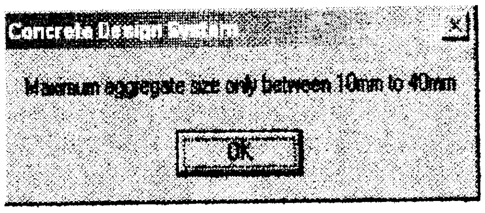
Concrete grade only between 25 to 70


Concrete Design System

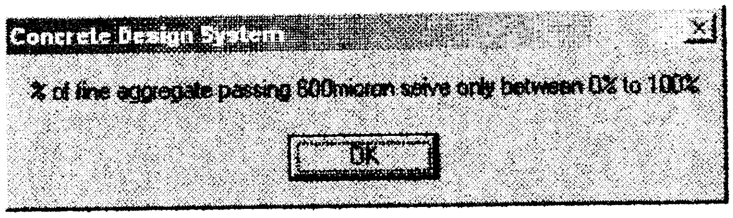
Proportion defective only between 0% to 10%


Concrete Design System

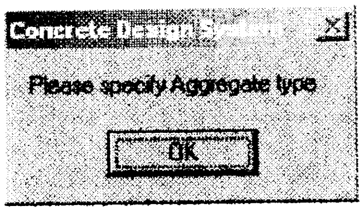
Standard deviation only between 4 N/mm2 to 8 N/mm2


Concrete Design System

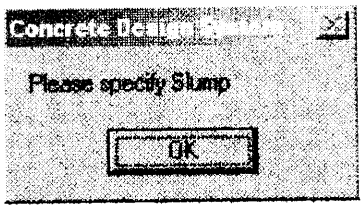
Maximum aggregate size only between 10mm to 40mm


Concrete Design System

% of fine aggregate passing 600micron seive only between 0% to 100%

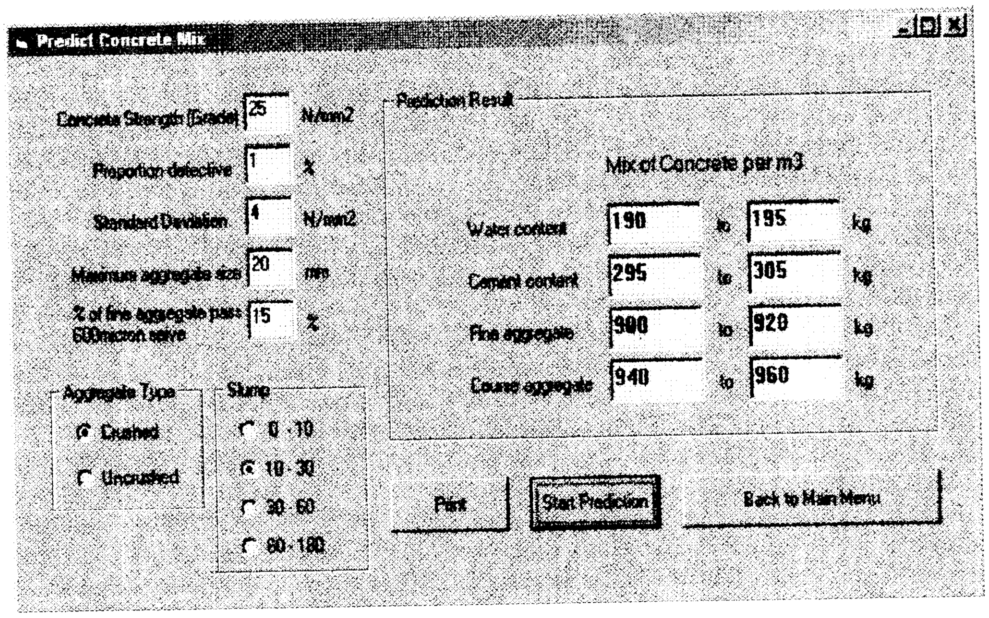

Concrete Design System

Please specify Aggregate type


Concrete Design System

Please specify Slump

- The prediction result will only appeared if the entered values are valid. The following is example of the prediction result of concrete mix.


Predict Concrete Mix

Concrete Strength (Grade) 25 N/mm2

Proportion defective 1 %

Standard Deviation 4 N/mm2

Maximum aggregate size 20 mm

% of fine aggregate pass 600micron sieve 15 %

Aggregate Type: [✓ Crushed] Uncrushed

Slump: 0 - 10 [✓ 10 - 30] 30 - 60 60 - 180

Prediction Result

Mix of Concrete per m3

Water content 190 to 195 kg

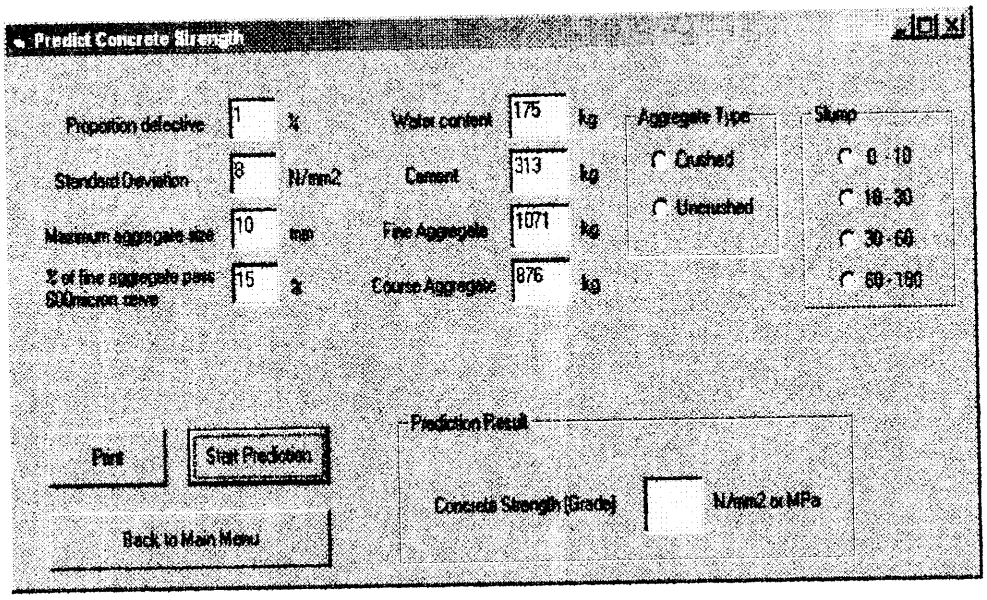
Cement content 295 to 305 kg

Fine aggregate 900 to 920 kg

Course aggregate 940 to 960 kg

3,0 Predict Concrete Strength

- The screen below will appeared after you choose "Predict Concrete Strength" button.
- Once again enter all the parameters required and then click "Start Prediction"


Predict Concrete Strength

Proportion defective 1 %

Standard Deviation 8 N/mm2

Maximum aggregate size 10 mm

% of fine aggregate pass 600micron sieve 15 %

Water content 175 kg

Cement 313 kg

Fine Aggregate 1071 kg

Course Aggregate 876 kg

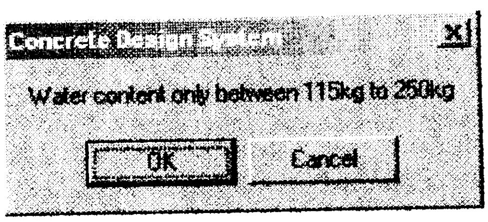
Aggregate Type: Crushed Uncrushed

Slump: 0 - 10 10 - 30 30 - 60 60 - 180

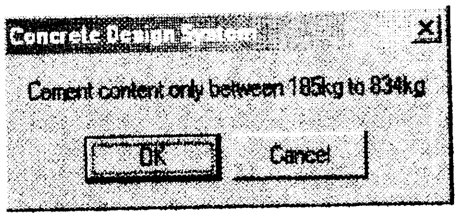
Prediction Result

Concrete Strength (Grade) N/mm2 or MPa

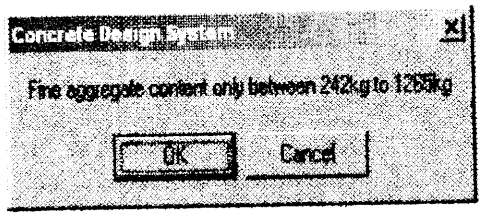
- The above messages and the additional messages below, will appeared if the invalid values are entered.


Concrete Design System

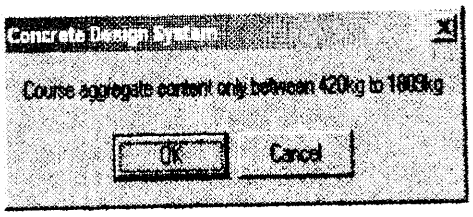
Water content only between 115kg to 250kg


Concrete Design System

Cement content only between 185kg to 834kg

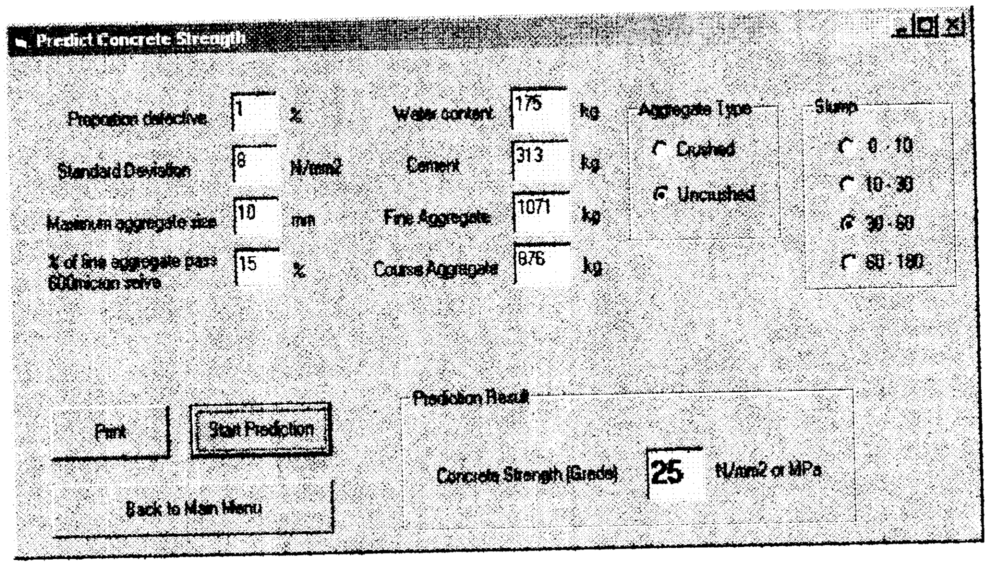

Concrete Design System

Fine aggregate content only between 242kg to 1265kg


Concrete Design System

Course aggregate content only between 420kg to 1869kg

♦ The prediction result will only appeared if the entered values are valid. The following is example of the prediction result of concrete strength.


Predict Concrete Strength

Proportion defective: 1 %
Standard Deviation: 8 N/mm2
Maximum aggregate size: 10 mm
% of fine aggregate pass 600micron seive: 15 %
Water content: 175 kg
Cement: 313 kg
Fine Aggregate: 1071 kg
Course Aggregate: 876 kg

Aggregate Type: Crushed [✓ Uncrushed]
Slump: 0 - 10, 10 - 30, [✓ 30 - 60], 60 - 180

Prediction Result
Concrete Strength (Grade): 25 N/mm2 or MPa