

CHAPTER 3

METHODOLOGY

3.1 Research Methodology

Methodology employed in this project starts with data collection from respective thermal power plant. Apart from that, literature review has been performed by study in details each of the papers (journals) collected which are very closely related. This study needs to be done in order to capture the main ideas of the relevancy of the project which will make the objectives clearer and achievable.

3.2 Preliminary Data Processing

i. Data Collection

The data collected consists of the data generated from all three (3) units of boilers in the thermal power plant and those data need to undergo several steps of procedure before used as the input variables to the NN model. Data collected comprising more than thousand variables.

ii. Data Cleaning

During data collection, large amount of data is captured and then need to be visualized sequentially to detect any missing or unclear data. It is crucial to ensure that all data obtained are relevant and sufficient to develop ANN modeling as the rests are interrelated and thus redundant.

iii. Data Filtering and Normalization

The accuracy of the training by a trained ANN can never be better than that of the training data, thus a critical scrutiny of obtained real data is required to identify and remove these erroneous data which is known as outliers. The erroneous data will affect the data normalization because it will produce error to the system. Hence, filtering is essential steps to be done to delete all the erroneous data. After data filtering, the data need to be normalized or stabilized in order to make sure that the ANN model will detect the data. Hence, the data will be normalized from the value more than hundreds into the value between 0 and 1. The data are normalized by using the equation shown below.

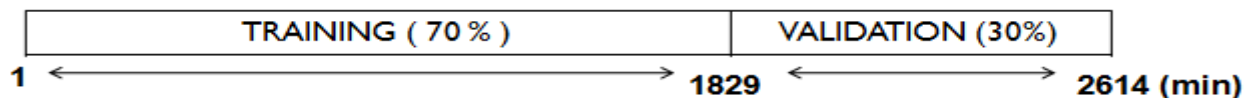
Normalization equation = _____

iv. Data Segmentation (Training and Validation)

Training ANN consists of training and validation. Training and validation have been classified into 70% training and 30% validation whereby these percentages are only will be calculated under data before shutdown.

After training and validation the ANN model by using different activation function and training algorithms, the best activation function with certain training algorithms which produced the smallest root mean square error (RMSE) will be selected to undergo the validation part. The validation part is to train the data once again by using different coding and simulation to finally produce the real forecasted data which occur before the real trip or in other words the forecasted data that detects the trip before the real trip.

DATA SEGMENTATION



v. NN Modeling (NN Topologies)

There are 9 types of multidimensional minimization backpropagation training algorithms but only 4 types have been selected to be used in the hidden layer to produce different errors of output. In constructing the ANN model, coding had been constructed based on the number of hidden layers used and types of function (training and validation). This part is the most crucial part whereby simulating each coding is time consuming. The ANN will be modeled by using 1 and 2 hidden layers, each with 1 to 10 neurons which will produce several different desired outputs. The network is trained by using only up to 2 hidden layers is because the RMSE value for the hidden layers more than 2 will be constant. For each hidden layers, the networks are trained using only up to 10 neurons because the value of the neurons will be greater than 1 which cause errors to the networks. The coding of the NN is done by using MATLAB and the types of training algorithms and activation functions used are discussed in details as below.

- **Training Algorithms**

The 4 types of training algorithms are tabulated in the Table 1.1. Function of these four training algorithms is for the convergence of the algorithms of models from ten to one hundred times faster than other algorithms in the NN MATLAB Toolbox. These faster algorithms fall into two categories. The first category uses heuristic techniques, which were developed from an analysis of the performance of the standard steepest descent algorithm. The second category of fast algorithms uses standard numerical optimization techniques.

Training Algorithms	Description of the functions
(Trainrp) Resilient backpropagation.	Eliminates the harmful effect of having a small slope at the extreme ends of sigmoid transfer function. It is generally much faster than the standard steepest descent algorithms. It also has the nice property that it requires only a modest increase in memory requirements.
(Trainscg) Scaled conjugate gradient.	A search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. This is a well know, highly efficient algorithm that gives good results on a broad spectrum of problems.
(Trainbfg) BFGS quasi-Newton.	This algorithm requires more computation in each iteration and more storage than the conjugate gradient methods, although it generally converges in fewer iterations.
(Trainlm) Levenberg-Marquardt.	Provides a numerical solution to the problem of minimizing a function, generally nonlinear, over a space of parameters of the function. These minimization problems arise especially in least squares curve fitting and nonlinear programming

Table 3.1 Training Algorithms [11]

Trainlm is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization. Trainlm is often the fastest backpropagation algorithm in the toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. However, trainrp is a network training function that updates weight and bias values according to the resilient backpropagation algorithm (Rprop). Trainrp can train any network as long as its weight, net input, and transfer functions have derivative functions.

On the other hand, trainscg is a network training function that updates weight and bias values according to the scaled conjugate gradient method. Trainscg can train any network as long as its weight, net input, and transfer functions have derivative functions. Backpropagation is used to calculate derivatives of performance with respect to the weight and bias variables. Finally, trainbfg is a network training function that updates weight and bias values according to the BFGS quasi-Newton method. Trainbfg can train any network as long as its weight, net input, and transfer functions have derivative functions.

- **Activation Functions**

There are three types of activation function used in ANN which consists of linear, tan-sigmoid and log sigmoid.

1. Linear Transfer Function

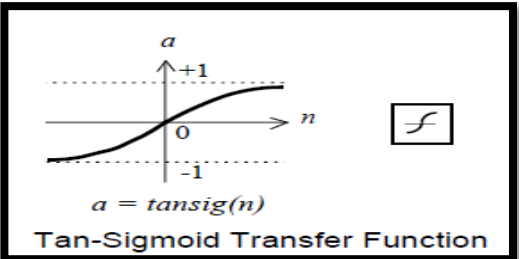
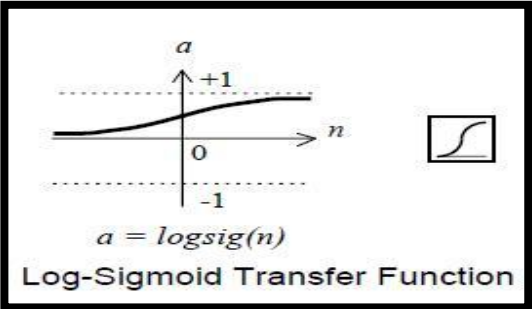
The linear transfer function calculates the neuron's output by simply returning the value passed to it. This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. A linear network cannot be made to perform a nonlinear computation.

2. Log-Sigmoid Transfer Function

This transfer function is commonly used in back-propagation networks, in part because it is differentiable. If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value.

3. Tan-Sigmoid Transfer Function

Multi layer alternatively can use Tan-Sigmoid Transfer function other than Log-Sigmoid.

Tan-Sigmoid Transfer Function	
Log-Sigmoid Transfer Function	

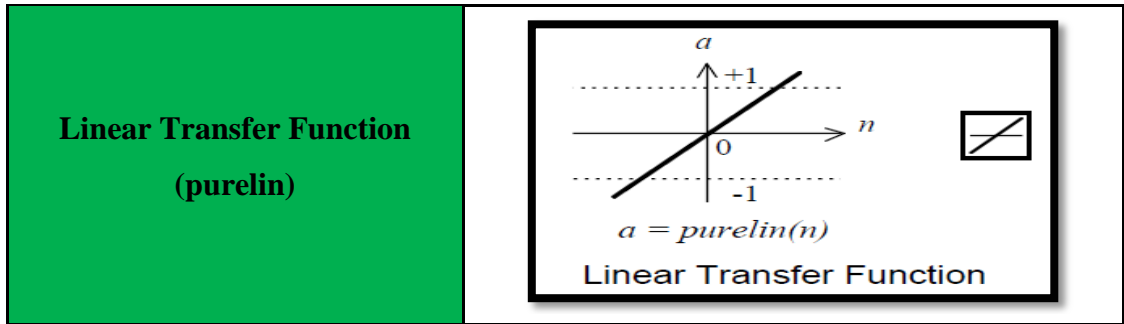


Table 4.2 Types of Activation Functions [11]

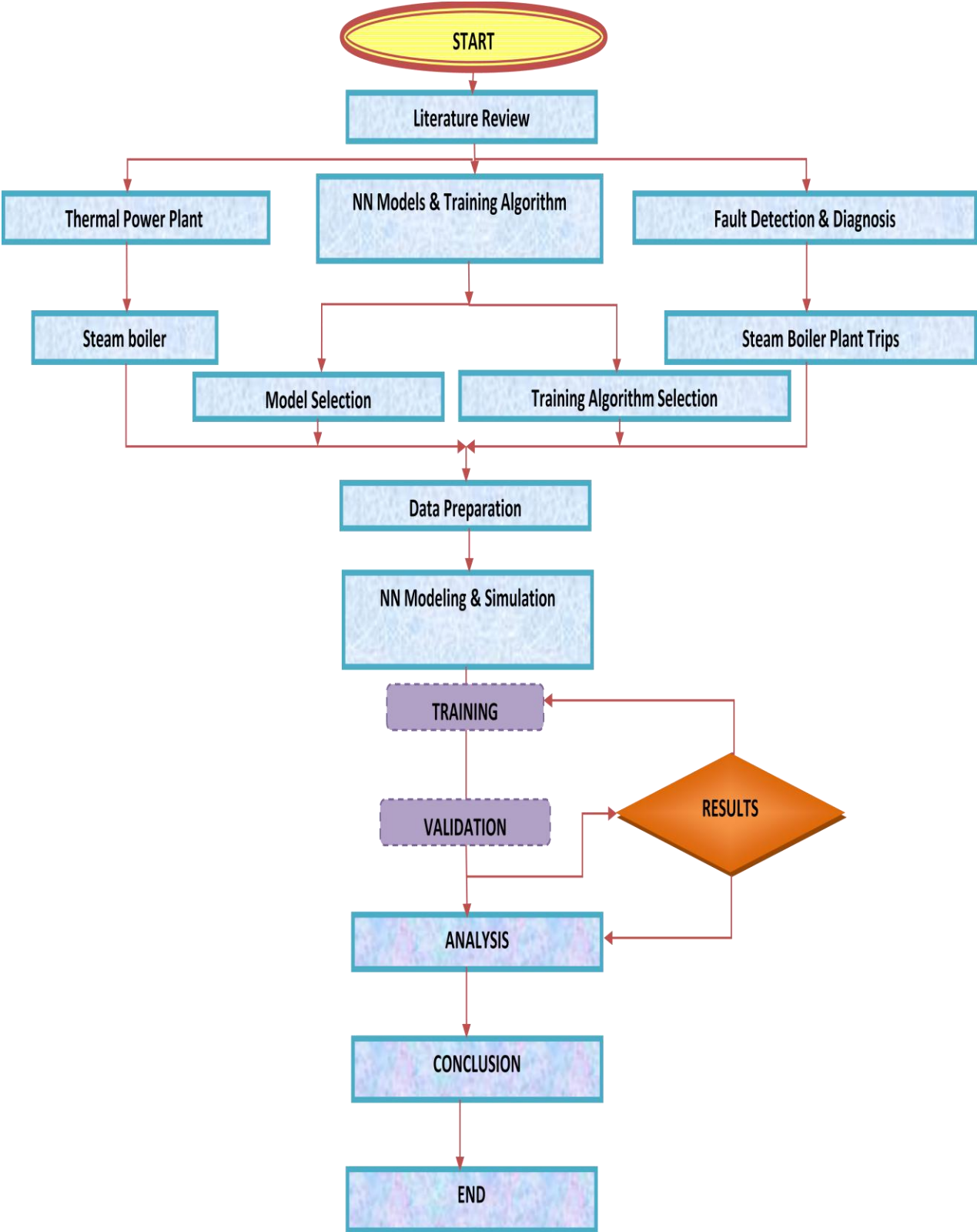
The function train carries out such a loop of calculation. In each pass, the function train proceeds through the specified sequence of inputs, calculating the output, error, and network adjustment for each input vector in the sequence as the inputs are presented.

To test the network, the original inputs are presented, and its outputs are calculated with simulation. The simulation will arise in order to measure the performance of each network. The performances of the network are calculated by using root mean square error (RMSE) as shown below in order to find out the weights that minimize error. The smallest the error, the better the output obtained. Then, the desired output obtained from MATLAB will be compared with the actual output and finally will come up with several recommendations and discussions. The RMSE are tabulated accordingly based on the number of neurons and training function and the smallest RMSE are identified based on the data collected.

—

- K = number of iterations
- Q = total number of iterations (epochs)
- t = target output
- a = actual output

3.3 Project Activities



3.4 Gantt Chart (FYP I) & Gantt Chart (FYP II)

Refer to Appendix

3.5 Tools Required

FYP I	FYP II
MICROSOFT EXCEL MATLAB	MICROSOFT EXCEL MATLAB (NN ToolBox)

For both FYP 1 and FYP 2, the tools used for simulation of the data are Microsoft Excel and MATLAB.