A NEURAL FUZZY APPROACH TO MODELING THE THERMAL BEHAVIOR OF POWER TRANSFORMERS

Huy Huynh Nguyen

A thesis submitted for the degree of
Master of Engineering

School of Electrical Engineering
Faculty of Health, Engineering & Science
Victoria University
Melbourne, Australia

2007

DECLARATION

I, Huy Huynh Nguyen, declare that the Master by Research thesis entitled A Neural Fuzzy Approach to Modeling the Thermal Behavior of Power Transformers is no more than 60,000 words in length, exclusive of tables, figures, appendices, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

Signature   Date

ABSTRACT

This thesis presents an investigation and a comparative study of four different approaches namely ANSI/IEEE standard models, Adaptive Neuro-Fuzzy Inference System (ANFIS), Multilayer Feedforward Neural Network (MFNN) and Elman Recurrent Neural Network (ERNN) to modeling and prediction of the top and bottom-oil temperatures for the 8 MVA Oil Air (OA)-cooled and 27 MVA Forced Air (FA)-cooled class of power transformers. The models were derived from real data of temperature measurements obtained from two industrial power installations. A comparison of the proposed techniques is presented for predicting top and bottom-oil temperatures based on the historical data measured over a 35 day period for the first transformer and 4.5 days for the second transformer with either a half or a quarter hour sampling time. Comparisons of the results obtained indicate that the hybrid neuro-fuzzy network is the best candidate for the analysis and prediction of the power transformer top and bottom-oil temperatures. The ANFIS demonstrated the best comparative performance in temperature prediction in terms of Root Mean Square Error (RMSE) and peak error.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisors Professor Greg Baxter and Professor Leonid Reznik for their comments and guidance throughout my research, without them none of this work would have been possible. I am very grateful to Professor Baxter for devoting his time in reviewing my thesis. I would also like to thank Dr. Daniel Kitcher and Dr. Juan Shi for their constructive comments on my work.

I am very thankful to my family, my parents, Mr. Huynh Nguyen and Mrs. Hien Nguyen, my sisters, Heather and Rachel, for their help and encouragement during the years.

I would like to dedicate this work to my wife, Kim Chi, and my daughter, Elizabeth. Without their constant love, support and understanding, I would not have had the strength to finish this work.

LIST OF SYMBOLS & ABBREVIATIONS

A	Amperes
ANFIS	Adaptive Neuro Fuzzy Inference System
C	Thermal capacity
ERNN	Elman Recurrent Neural Network
FA	Forced Air cooling mode
FIS	Fuzzy Inference System
IEEE	Institute of Electrical and Electronics Engineers
I_{load}	Load current
k	Time step index
K	Ratio of load L to rated load
KVA	KiloVoltAmperes
logsig	log-sigmoidal transfer function
MF	Membership Function
MFNN	Multilayer Feedforward Neural Network
MIT	Massachusetts Institute of Technology
MLP	Multilayer Perceptron
MVA	MegaVoltAmperes
n	Oil exponent
OA	Oil Air cooling mode
P_{fl}	Rated load
purelin	linear transfer function
R	Ratio of load loss to no-load loss at rated load
RMSE	Root Mean Square Error
T_o	Time constant at rated KVA
θ_{amb}	Ambient air temperature
θ_{bot}	Bottom-oil temperature
θ_{bot_m}	Measured bottom-oil temperature

θ_{bot_p}	Predicted bottom-oil temperature
θ_{fl}	Top-oil rise over ambient temperature at rated load
θ_i	Initial top-oil rise for t = 0
θ_o	Top-oil rise over ambient temperature
θ_{top}	Top-oil temperature
θ_{top_m}	Measured top-oil temperature
θ_{top_p}	Predicted top-oil temperature
θ_u	Ultimate top-oil rise for load L
η	Learning rate
Δt	Sampling period
$^{o}\mathrm{C}$	Degrees Celsius

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

1.1 Background

A power transformer is a static piece of apparatus with two or more windings. By electromagnetic induction, it transforms a system of alternating voltage and current into another system of alternating voltage and current of different values, of the same frequency, for the purpose of transmitting electrical power. For example, distribution transformers convert high-voltage electricity to lower voltage levels acceptable for use in homes and businesses (IEC, 1993).

A power transformer is one of the most expensive pieces of equipment in an electricity system. Monitoring the performance of a transformer is crucial in minimizing power outages through appropriate maintenance thereby reducing the total cost of operation. The transformer top-oil temperature value is one of the most critical parameters when defining the power transformer thermal conditions (Feuchter, 1993). Therefore, it is very useful to accurately predict these temperature values. Traditional approaches to modeling a transformer's top-oil temperature rely a great deal on mathematical formulae which require a high level accuracy of each parameter involved. The use of these mathematical formulae is valid and acceptable when the system is simple and properly monitored. However, in the case of a typical transformer system like those under consideration in this thesis, which is not a straightforward linear system, mathematical formulae like IEEE and MIT become less effective as shown in Chapter 6. To replace these mathematical modeling approaches, this thesis provides the use of neural fuzzy and other soft computing techniques in order to better predict temperatures of a transformer.

1.2 Introduction

Accurate prediction is the most fundamental but not necessarily be the only objective in modeling. The model should serve as a good description of the data for enlightening the properties of the input-output relationship. The model should also be interpretable, so that the user can gain insight and understand the system that produced the data.

In nearly all everyday systems, models are derived from two fundamental sources: empirical data acquired from observation and *a priori* knowledge about the system. These two knowledge sources are invaluable in any modeling process, where all available *a priori* knowledge should be utilized. In addition where inadequacies are found in these two knowledge sources these be compensated by the ability to learn from the data. Depending on the extent to which these two types of knowledge are investigated, three basic levels of model synthesis can be defined (Ljung, 1987):

1. White Box. The model is completely constructed from *a priori* knowledge and physical insight. Here, empirical data are not used during model identification and are only used for validation. Complete *a priori* knowledge of this kind is very rare, because usually some aspects of the distribution of the data are unknown.

2. Grey Box. An incomplete model is constructed from *a priori* knowledge and physical insight, and then the available empirical data are used to adapt the model by finding several specific unknown parameters.

3. Black Box. No *a priori* knowledge is used to construct the model. The model is chosen as a flexible parameterized function, which is used to fit the data.

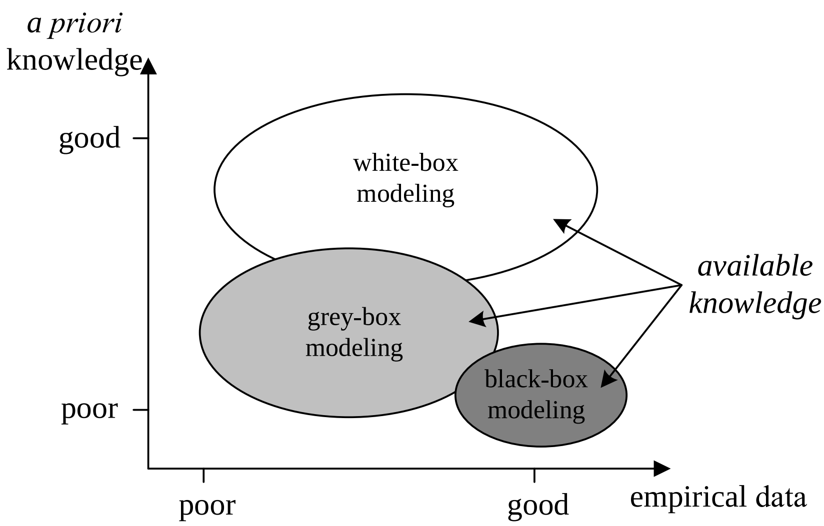

Figure 1.1: The different modeling approaches

The three different modeling approaches are shown in Figure 1.1. When the available *a priori* knowledge and the empirical data are poor, the most suitable approach is expected to be the grey-box modeling type, even though none of the three modeling approaches can be easily applied, due to the lack of knowledge. For most applications, both *a priori* knowledge and empirical data need to be employed when developing a model.

The *a priori* knowledge comes in many different forms, a common one being linguistic constructs or rules describing the input-output relationships. These are qualitative descriptions where qualitative refers only to the characteristics of something being described, rather than exact numerical values. For example: the temperature is high. This type of knowledge can be of a great benefit in the modeling process.

Fuzzy sets (Zadeh, 1965) developed by Zadeh are powerful tools for capturing such qualitative *a priori* knowledge, especially in engineering fields such as control and system identification. Fuzzy models consist of a series of linguistic rules, which can easily be understood and constructed. As the information is stored as a set of interpretable rules, fuzzy models are said to be *transparent*. Conventionally, the fuzzy modeling approach assumes the *a priori* knowledge is correct, equivalent to the *white box modeling* approach. This often proves to be unrealistic, as with the vagueness and subjectivity of language. Indeed, it is unlikely an expert can accurately describe the complete behavior of a system by simply using the underlying data. For example, *a priori* knowledge is expressed in the following linguistic rule (see Chapter 3):

If the temperature is high *then* the pressure will increase

The vagueness in this rule is the statement "the temperature is high". The temperature can be any value in a range.

On the other hand, neural networks have been used extensively in *black box modeling* approaches. These are highly flexible models that can be successfully adapted to model many different input-output mappings by learning from data. However, there is one major set back of neural networks is that they do not produce transparent models, that is the stored knowledge is not easily interpretable. Thus, the main criticism addressed to neural networks concerns their black box nature (Lee, 1990).

Recently, to avoid the inadequacies of both fuzzy and neural networks modeling, neural fuzzy modeling approaches have been introduced as an ideal technique for utilizing both *a priori* knowledge and empirical data. Neural fuzzy approaches combine the desired attributes of fuzzy logic and neural networks, hence producing flexible models that can learn from empirical data and can be represented linguistically by fuzzy rules. With reference to the different levels of model synthesis described above, neural fuzzy modeling can be regarded as *grey box modeling*. The advantages of neural fuzzy modeling are:

1. Model identification can be performed using both empirical data and qualitative knowledge.
2. The resulting models are transparent.

1.3 Neural Fuzzy Modeling: The Basics

Fundamentally, neural fuzzy modeling is the task of building models from a combination of *a priori* knowledge and empirical data. Normally, such *a priori* knowledge is used to define a suitable model structure; this model is then adapted such that it successfully reproduces the available empirical data. This adaptation step is often called *learning*. The main objective of neural fuzzy modeling is to construct a model that accurately predicts the value(s) of the output variable(s) when new values of the input variables are presented.

1.3.1 Objectives

The basic objectives of neural fuzzy modeling are summarized below:

- Predictability. This is generally the most important objective, which aims to construct a model from observed data so that the model approximates the behavior of the system that produced the data. The ability to model unseen data is called *generalization* and is the fundamental principle of neural fuzzy modeling.

- Interpretability. This can only be achieved if the final model is transparent, that is the representation of the extracted knowledge can be clearly defined. There are several different levels of model transparency:

  1. Indication of which input variables affect the output variable(s).
  2. Use of simple mathematical relationships.
  3. Qualitative descriptions of the modeled system.

- Efficiency. For many applications, resources are limited and therefore physical constraints are placed on the model's computational requirements; small models that can be implemented by efficient algorithms are required.

- Adaptability. The ability to adapt and learn from the patterns of data that were not present in the original training and testing sets.

1.3.2 Main Phases

The preliminary phase is the first and most important step of a neural fuzzy modeling process. It aims to collect a set of data, which is expected to be a representative sample of the system to be modeled. In this phase, known as *data preprocessing*, data are cleaned to make learning easier. This involves incorporation of all relevant domain knowledge at the level of an initial data analysis, including any sort of preliminary filtering on the observed data such as missing data treatment or feature selection. The preprocessing phase returns the data set in a structured input-output form, commonly called a *training set.*

Once this preliminary phase is completed, the learning phase begins. This thesis will focus exclusively on this second phase assuming that data have already been preprocessed. The learning phase is essentially a search, in a space of possible model configurations, of the model that best represents the power transformer temperature values. As in any other search task, the learning procedure requires a search space, where the solution is to be found, and some assessment criterion to measure the quality of the solution. As far as the search space is concerned, the Adaptive Neuro-Fuzzy Inference System (ANFIS), Multilayer Feedforward Neural Network (MFNN) and Elman Recurrent Neural Network (ERNN) are chosen to compare with the conventional Institute of Electrical and Electronics Engineers (IEEE) and Massachusetts Institute of Technology (MIT) models for this thesis. As far as the assessment of a model is concerned, both qualitative and quantitative criteria are defined, according to the main goal of neural fuzzy modeling that is to attain a good generalization. The next step is to search for the best model. The search algorithm consists of three main steps: structural identification, parametric identification and model validation.

- **Structural identification** is typically an iterative process that seeks the structure that is expected to have the best performance. The model structure from the previous step in the iteration is usually used as basis for the selection of a more promising structure, either refined or simplified.

- **Parametric identification** is the process to determine the relevant parameters of the model. This is typically an inner loop into the structural identification, since it returns the best model for a fixed structure.
- **Model validation**. At this stage the quality of the model is evaluated by analyzing how well it represents (captures) the data. Typically validation is performed by a combination of statistical measures that evaluate the generalization capability of the model, together with qualitative criteria, whose purpose is to establish how the model relates to the *a priory* knowledge, how easy it will be to use and interpret.

1.4 Organization of the Thesis

The objectives of this thesis are to:

- describe the steps involved in building a neural fuzzy model.
- briefly describe the current conventional methods.
- review the methodologies of fuzzy logic and neural networks and the significance of combining the two technologies.
- develop a neural fuzzy model to predict the temperatures of a power transformer.
- compare the results when using conventional techniques against Adaptive Neuro-Fuzzy Inference System (ANFIS), Multilayer Feedforward Neural Network (MFNN) and Elman Recurrent Neural Network (ERNN) to predict the temperatures of a power transformer.
- make recommendations for future research in the field.

Chapter 2 presents an overview of the conventional approaches used in this thesis to model the thermal behavior of a transformer. It provides the derivation of relevant equations used in the prediction of the transformer's top and bottom oil temperatures based on ANSI/IEEE standard models.

Chapter 3 presents the notations, structures and operations on fuzzy logic and neural networks.

Chapter 4 describes the strengths and weaknesses of fuzzy logic and neural network modeling approaches. An integrated model (neural fuzzy) that benefits from the strengths of each of fuzzy logic and neural network models are presented, namely ANFIS (Adaptive Neuro Fuzzy-based Inference System).

Chapter 5 shows the application of ANFIS modeling the transformer's thermal behavior. Particularly, ANFIS is used to predict the transformer's top and bottom-oil temperature. A comparison between ANFIS and other models, i.e. the IEEE and neural network models (multilayer feedforward and Elman) are carried out in this chapter.

Finally, in Chapter 6, conclusions of this research and recommended directions for further investigation are discussed.

CHAPTER 2 : THERMAL MODELING OF A POWER TRANSFORMER

2.1 Introduction

Abnormal temperature readings almost always indicate some type of failure in a transformer. For this reason, it has become common practice to monitor the top and bottom-oil temperatures of a transformer. Utilities can save millions of dollars by using a model that allows them to predict these values accurately. Accurate top and bottom-oil temperature prediction also allows system planners and operators to plan for and take necessary corrective action in response to transformer outages.

2.2 Conventional Models

There are several models that have been used in practice for predicting transformer temperatures (Pierce, 1992), (Pierce, 1994), (IEEE, 1995) and (Lesieutre, 1997). These models can be used with manufacturer-supplied coefficients (e.g. rated load, thermal capacity, oil exponent, etc.) provided that the necessary transformer parameters are monitored. If the required parameters are not monitored routinely then the models cannot be used. Parameters that are routinely measured include the ambient temperature, top-oil temperature, bottom oil temperature and load current. One model that has been employed is the so-called top-oil-rise model (IEEE, 1995). The top-oil-rise model is governed by the first order differential equation,

$$T_o \frac{d\theta_o}{dt} = -\theta_o + \theta_u \tag{2.0}$$

which has the solution,

$$\theta_o = (\theta_u - \theta_i)(1 - e^{-t/T_o}) + \theta_i \tag{2.1}$$

where,

$$\theta_u = \theta_{fl}\left(\frac{K^2(R+1)}{R+1}\right)^n \tag{2.2}$$

$$T_o = \frac{C\theta_{fl}}{P_{fl}} \tag{2.3}$$

$$K = \frac{I}{I_{rated}} \tag{2.4}$$

and,

θ_o : top-oil-rise over ambient temperature (^oC);

θ_{fl} : top-oil-rise over ambient temperature at rated load (^oC);

θ_u : ultimate top-oil-rise for load L (^oC);

θ_i : initial top-oil-rise for t = 0 (^oC);

θ_{amb} : ambient air temperature (^oC);

T_o : time constant at rated KVA (h);

P_{fl} : rated load (MVA);

C : is the thermal capacity of the transformer, Watt-hours/^{o}C (Wh/^oC);

n : is an empirically derived exponent used to calculate the variation of top or bottom-oil temperature with changes in load. The value of *n* has been specified by the manufacturer for each mode of cooling to approximately account for effects of change in resistance with change in load (IEEE, 1995).

K : ratio of load *L* to rated load;

R : ratio of load loss to no-load loss at rated load.

If Equation 2.1 is solved, the top (θ_{top}) oil temperature, is then given by:

$$\theta_{top} \text{ or } \theta_{bot} = \theta_o + \theta_{amb} = (\theta_u - \theta_i)(1 - e^{-t/T_o}) + \theta_i + \theta_{amb} \tag{2.5}$$

To accurately predict θ_{top}, we need to find the parameters T_o, R and θ_{fl}. There are several ways to do this. One way is using linear regression along with measured data. To use linear regression we must first construct a discrete-time form of (2.1). Applying the forward Euler discretization rule,

$$\frac{d\theta_o[k]}{dt} = \frac{\theta_o[k] - \theta_o[k-1]}{\Delta t} \tag{2.6}$$

where Δt is the sampling period. Solving we get,

$$\theta_o[k] = \frac{T_o}{T_o + \Delta t}\theta_o[k-1] + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)} \left(\frac{\left(\frac{I[k]}{I_{rated}}\right)^2 R + 1}{R+1} \right)^n \tag{2.7}$$

where $I[k]$ is the per-unit transformer current (based on the rated value of the transformer) at time step index *k*.

when the load current is near its rating, or $R > 1$ and $K^2 R > 1$, top-oil temperature rise over ambient temperature from (2.7) may then be given by,

$$\theta_o[k] = \frac{T_o}{T_o + \Delta t}\theta_o[k-1] + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)} \left(\frac{I[k]}{I_{rated}} \right)^{2n} \tag{2.8}$$

$$\theta_o[k] = K_1 \theta_o[k-1] + K_2 I[k]^{2n} \tag{2.9}$$

For comparison purposes, this model will be called **Model 1**. This is the model used in the MIT monitoring system, and has been shown to be reliable in the MIT pilot transformer test facility (Lesieutre, 1997).

Using the value n = 1 for a transformer in the forced cooling state (IEEE, 1995), Equation (2.7) is given by (Lesieutre, 1997):

$$\theta_o[k] = \frac{T_o}{T_o + \Delta t}\theta_o[k-1] + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)(R+1)}\left(\frac{I[k]}{I_{rated}}\right)^2 + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)(R+1)} \tag{2.10}$$

$$\theta_o[k] = K_1 \theta_o[k-1] + K_2 I[k]^2 + K_3 \tag{2.11}$$

In this thesis this is called <u>**Model 2**</u>.

Model 2 is a simplified model that has the limitation that it does not accurately account for the effects of ambient temperature dynamics on top-oil temperature. It can be shown that the model proposed by (Lesieutre, 1997) accounts for dynamic variations in ambient temperature. The model proposed by Lesieutre can be viewed as a slight modification of model 1,

$$T_o \frac{d\theta_{top\ \mathrm{or\ bot}}}{dt} = -\theta_{top\ \mathrm{or\ bot}} + \theta_{amb} + \theta_u \tag{2.12}$$

where,

θ_u is still defined by Equation (2.2) and θ_{amb} is the ambient temperature. Following the same assumptions as above, when n = 1, Equation (2.12) has the solution:

$$\theta_{top\ \mathrm{or\ bot}}[k] = \frac{T_o}{T_o + \Delta t}\theta_{top\ \mathrm{or\ bot}}[k-1] + \frac{\Delta t}{T_o + \Delta t}\theta_{amb}[k] + \frac{\Delta t \theta_{fl} R}{(T_o + \Delta t)(R+1)}\left(\frac{I[k]}{I_{rated}}\right)^2 + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)(R+1)} \tag{2.13}$$

$$\theta_{top\ \mathrm{or\ bot}}[k] = K_1 \theta_o[k-1] + (1-K_1)\theta_{amb}[k] + K_2 I[k]^2 + K_3 \tag{2.14}$$

This model has been designated **Model 3.**

It has been shown in (Tylavsky, 2000) that if $(1-K_1)$ in (2.14) is replaced by another coefficient, K_4, the result,

$$\theta_{\text{top or bot}}[\mathrm{k}] = \mathrm{K}_1\theta_{\mathrm{o}}[\mathrm{k}-1] + \mathrm{K}_4\theta_{\mathrm{amb}}[\mathrm{k}] + \mathrm{K}_2\mathrm{I}[\mathrm{k}]^2 + \mathrm{K}_3 \qquad (2.15)$$

is a model whose performance is slightly better due to the additional free parameter. This model is referred to as the **semi-physical model** because it is not entirely based on physical principles. This model has not been used in the work presented in this thesis and has been included here for the sake of completeness.

CHAPTER 3 : NEURAL NETWORKS AND FUZZY LOGIC

3.1 Introduction

This chapter describes the basic concepts, operations and structures of neural networks and fuzzy systems, and reviews previous work as presented in the literature.
More specifically section 3.2 will introduce:

- the concepts, notations and operations of neural networks, and,
- two of the most popular neural networks: multilayer feedforward and Elman recurrent neural networks.

The learning rules and structures of two of the most commonly used neural networks will also be briefly described.

In Section 3.3, the following will be introduced:

- the concepts, notations and operations of fuzzy sets;
- fuzzy inference systems which employ fuzzy "if-then" rules and fuzzy reasoning, and;
- Mamdani and Takagi-Sugeno fuzzy inference systems.

3.2 Neural Networks

Artificial neural networks, commonly referred to as "neural networks" are systems that are intentionally constructed to make use of some organizational principles similar to those of the human brain. They represent a promising new generation of information processing systems.

Neural networks have a large number of highly interconnected *processing elements* (*nodes* or *units*). A neural network is a massively parallel distributed processor inspired by the real biological neuron in the brain; therefore, it has the ability to learn, recall, and generalize as a consequence of training patterns or data.

3.2.1 Biological Neural Networks

Almost 100 years ago, a Spanish histologist, Santiago Ramon y Cajal (1911), the father of modern brain science, realized that the brain was made up of discrete units (*processing elements*) he called *neurons,* the Greek word for nerves. A typical biological neuron is shown in Figure 3.1. A human brain consists of approximately 10^{11} neurons of different shapes. Cajal described neurons as polarized cells that receive signals via highly branched extensions, called *dendrites*, and send information along unbranched extensions, called *axons*. The end of an axon splits into strands. Each strand terminates in a small bulblike shape called a *synapse* (There are approximately 10^{4} synapses per neuron in a human brain), where the neuron introduces its signal to the neighboring neurons. The signal in the form of an electric impulse is then received by a dendrite. This type of signal transmission involves a complex chemical process in which specific transmitter substances are released from the sending side. This raises or lowers the electric potential inside the cell body called *soma* of the receiving neuron. The receiving neuron *fires* if its electric potential reaches a certain level called *threshold*, and a pulse or *action potential* of fixed strength and duration is sent out through the axon to synaptic junctions to other neurons. After firing, a neuron has to wait for a period of time called the *refractory period* before it can fire again. Synapses are *excitatory* if they let passing impulses cause the firing of the receiving neuron, or *inhibitory* if they let passing impulses hinder the firing of the neuron. A good overview of biological neural networks can be found in (Hassoun, 1995).

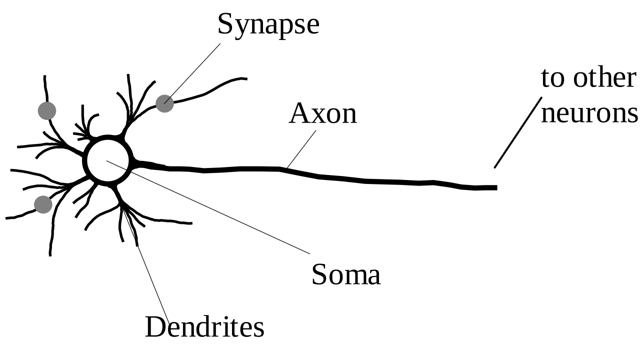

Figure 3.1: A biological neuron

3.2.2 Artificial Neural Networks

The goal of artificial neural network research is to develop mathematical models of its biological counterpart in order to imitate the capabilities of biological neural structures with a view to the design of intelligent control systems.

The first mathematical model of the neuron, shown in Figure 3.2, was introduced by Warren McCulloch and Walter Pitts (McCulloch, 1943). It is known as the McCulloch-Pitts model, it does not possess any learning or adaptation capability. Many of the later neural network models use this model as the basic building block. This model consists of a single neuron, which receives a set of inputs ($x_1, x_2, \ldots, x_n$). This set of inputs is multiplied by a set of weights ($w_1, w_2, \ldots, w_n$). Here, weights are referred to as strengths of the synapses. These weighted values are then summed and the output is passed through an activation (transfer) function. The activation function is also referred to as a *squashing function* in that it squashes (limits) the permissible range of the output signal to some finite value.

The output y is 1 (firing) or 0 (not firing) according to whether the weighted input sum is above or below a certain threshold (bias) ϑ .

$$\begin{aligned} u &= \sum_{i=1}^{n} w_i x_i - \vartheta \\ &= \mathbf{w}^T.\mathbf{x} \\ y &= f(u), \end{aligned} \tag{3.1}$$

where the weight vector $\mathbf{w}$ = $[w_1 \quad w_2 \quad \cdots \quad w_n]^T$ and the input vector $\mathbf{x}$ = $[x_1 \quad x_2 \quad \cdots \quad x_n]^T$. In this model, the weight vector and the threshold (bias) term are fixed. In general, $\mathbf{A}^T$ is the transpose of a vector $\mathbf{A}$.

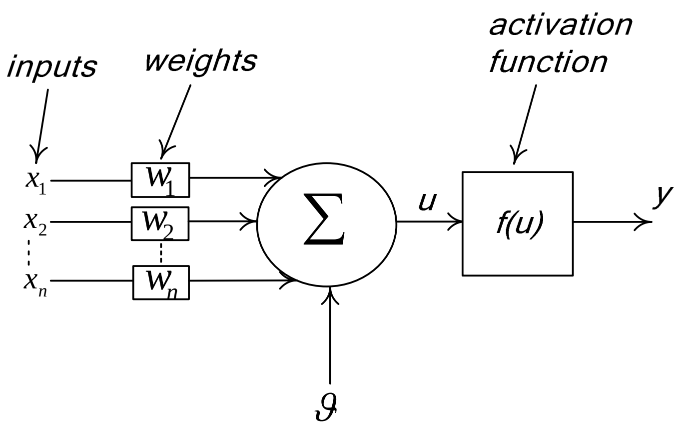

Figure 3.2: An artificial neuron

3.2.3 The Perceptron

Researchers used the McCulloch-Pitts model to develop many neural network models with learning and adaptation capabilities. One such model is the Perceptron as shown in Figure 3.3 as developed by Frank Rosenblatt in 1958. The perceptron is the simplest form of a neural network used for pattern classification. Basically, it consists of a single neuron with adjustable weights (w_i's) and threshold (ϑ). The main purpose is to train the perceptron until the neuron's output y matches the desired output y_d. To do this a learning procedure was used by Rosenblatt to adjust the free parameters (i.e. w_i's) for his perceptron brain model. For an in depth description of the training, formulation and convergence of the perceptron, the reader is directed to (Rosenblatt, 1958) and (Rosenblatt, 1962).

Since Rosenblatt's original Perceptron was introduced, many other neural network models, that make use of this Perceptron model, have been developed.

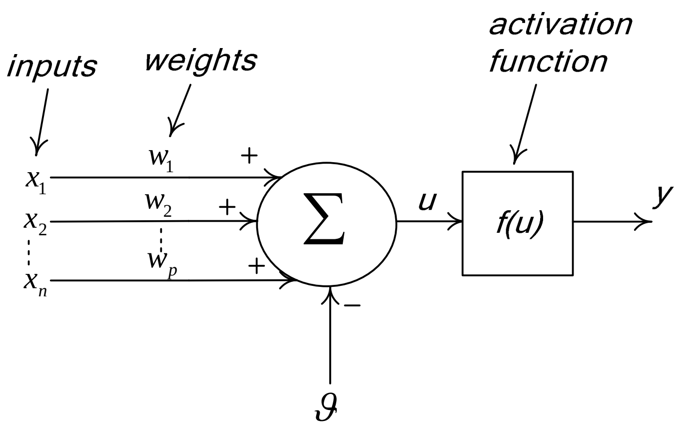

Figure 3.3: Structure of a perceptron

3.2.4 Feedforward Neural Networks

The *feedforward neural network* is the first and simplest type of artificial neural networks. In this network, the data moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. Feedforward Neural Networks are Neural Networks that utilize more than one neuron (node) and contain no feedback paths within the network. There are two different configurations of feedforward neural networks, namely *Single-Layer Feedforward Neural Networks* and *Multilayer Feedforward Neural Networks*.

3.2.4.1 Single-Layer Feedforward Neural Networks

A Single-Layer Feedforward Neural Network is shown in Figure 3.4. In this network, there is only one input layer and one output layer. A *layer* can be one to several neurons (nodes) connecting in parallel. The network is strictly feedforward, that is, there is no feedback connections from the outputs back to the inputs. Usually, no connections exist between the neurons (nodes) in a particular layer. The network shown in Figure 3.4 is fully connected, that is, all inputs are connected to all the nodes. Partially connected networks are those where some of the connection links are missing.

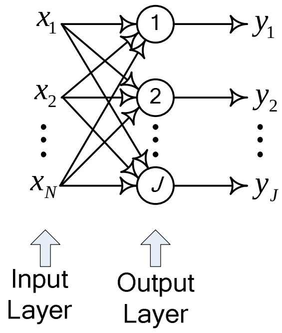

Figure 3.4: Structure of a single-layer feedforward neural network

3.2.4.2 Multilayer Feedforward Neural Networks (MFNN)

A Multilayer Feedforward Neural Network is shown in Figure 3.5 is the most widely used neural networks, particularly within the area of systems and control (Narendra, 1990). Similar to the single-layer feedforward neural networks, there is one input layer and one output layer, and no interconnections between the nodes in a particular layer. But different from the single-layer feedforward neural networks, multilayer neural networks have a number of intermediate or *hidden layers* (any layer between the input and output layers, is called a *hidden layer* because it is internal to the network and has no direct contact with the external environment) existing between the input and output layer. One, two or even no hidden layers are used for most applications. The small number of hidden layers is due to the fact that the training process becomes too long and tedious if the architecture of the neural network becomes large. In Figure 3.5, one hidden layer is present in this multilayer neural network, where $J \neq K \neq N; J, K, N \in \Re$. To get the output from the network, a set of input data is first presented as inputs to the input layer in turn. The outputs from this layer are then fed, as inputs to the first hidden layer, and subsequently the outputs from the first hidden layer are fed, as weighted inputs (the outputs from the first hidden layer are multiplied by the weights), to the second hidden layer. This process carries on until the output layer is reached. An example of a feedforward neural network is the multilayer perceptron (MLP) (commonly called the multilayer feed forward network). The reader is referred to the discussion of other feedforward neural networks in (Haykin, 1994).

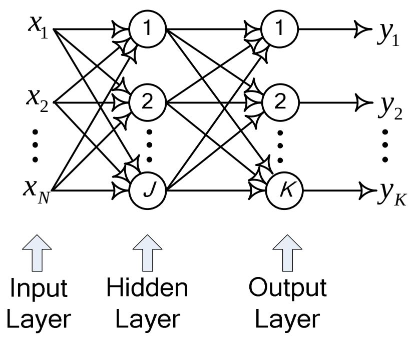

Figure 3.5: Structure of a multilayer feedforward neural network

3.2.5 Recurrent Neural Networks

The feedforward neural networks previously discussed are strictly ‘feedforward’ networks in which there are no feedbacks from the output of one layer to the inputs of the same layer (i.e. no interconnection between the nodes within the same layer) or earlier layers of nodes. Also, these networks have no *memory* (i.e. the input vector [input data set] at any time instant determines the output, assuming the weights do not vary). A recurrent neural network as shown in Figure 3.6 is different from the feedforward neural network because it has feedback connections. Similar to the use of feedback in control systems, recurrent neural networks take into consideration the dynamic behavior of systems.

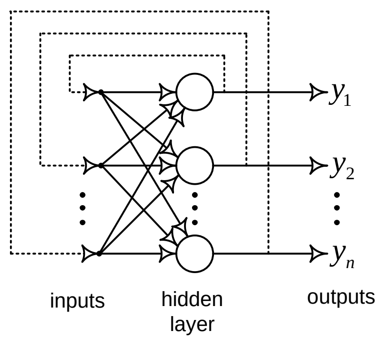

Figure 3.6: Structure of a recurrent neural network

The output of a node at any time instant t depends on its inputs at time instant t and those feedback connections whose values are a time instant earlier (t - Δt), where Δt is the sampling time. As the current output of the recurrent neural network depends on both current and prior inputs, recurrent networks function behave just like memories which have stored values. Examples of recurrent neural networks include the Elman neural network (Elman, 1990), the Hopfield network (Hopfield, 1982) and the Jordan network (Jordan, 1986). The Elman network will be discussed in Section 3.2.6 of this thesis. Readers may refer to (Haykin, 1994) for more details of other neural networks.

3.2.6 Elman Recurrent Neural Network (ERNN)

The Elman neural network is a partial recurrent network model that was first proposed by Elman (Elman, 1990). Different to the original recurrent network, an Elman network has a number of *context nodes* in the input layer as shown in Figure 3.7. The context nodes do nothing more than duplicate the activity of a hidden layer, at the previous time step, at the input of the network. This variation allows the Elman network to deal with *conflicting patterns*. (Conflicting patterns refer simply to a one-to-many mapping: that is, multiple outputs generated from a single input pattern.) Such a condition will confound a standard recurrent network. The Elman network, however, deals with such a situation by augmenting the input pattern with the condition of a hidden layer at the previous time step. Thus, the feedback units are essentially establishing a *context* for the current input, allowing the network to discriminate between “identical” input patterns that occur at different times. The advantage of Elman networks over fully recurrent networks is that back propagation (Werbos, 1974) is used to train the network while this is not possible with other recurrent networks where the training algorithms are more complex and therefore slower (El Choubassi, 2003).

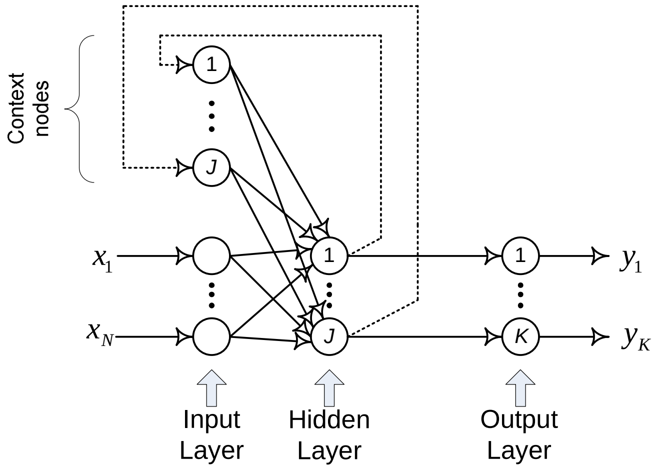

Figure 3.7: Structure of an Elman recurrent neural network

3.2.7 Neural Networks Selection

The multilayer Feedforward neural network is one of the more commonly used structures within the neural network development. Applications using multilayer feedforward neural networks include image processing, prediction, signal processing, and robotics. The multilayer feedforward neural network has proven to be quite capable of approximating non-linear functions (Rumelhart, 1986). Recently, many papers have been published on the use of Feedforward Neural Networks (FNN) for a variety of applications with good results (Brouwer, 2004), (Mirhassani, 2005), (Thukaram, 2006) and (Marashdeh, 2006). To date none of the reported applications are in the field of transformer top-oil temperature prediction.

Elman networks have been applied widely in the fields of identification, prediction and control. Recently, Elman networks have been found to provide good results in electric load forecasting (Tsakourmis, 2002), daily peak temperature forecasting (Vitabile, 2004) and detecting and classifying attacks in computer networks (Aquino, 2005).

For the above reasons, Elman and multilayer feedforward neural networks have been employed to predict the transformer's top-oil temperature in this thesis. A comparison of these two networks with the conventional ANSI/IEEE standard models (see Chapter 2) and the ANFIS model (see Chapter 4) can be found in Chapter 5.

3.2.8 Learning in Neural Networks

The ability of neural networks to learn about their environment and to adaptively fine-tune their parameters to improve the systems' performance is one of their strong points. Being able to model systems allows neural networks to be used in a variety of control systems applications. Learning in a neural network is performed to mimic the behavior of a biological neuron and is still undergoing intense research. Through learning a neural network is able to adapt itself and subsequently improve its performance in a gradual manner. The learning process is completed when the neural network is able to produce the desired outputs when different inputs are applied to it. More specifically, the neural network learns or adapts itself by adjusting its parameters (i.e. weights and threshold). There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

In this thesis, the scope is confined to modeling the transformer's top and bottom-oil temperatures prediction with desired input-output data sets, so the resulting networks must have adjustable parameters (i.e. weights and threshold) that are update by a supervised learning rule.

3.2.9 Supervised Learning

Supervised learning is also known as *learning with a teacher*. Figure 3.8 shows the structure of this form of learning. The *teacher* (supervisor) has the knowledge of the system's environment. A training data set, comprising an input vector $\mathbf{x}$ (input data set) and the corresponding desired output vector $\mathbf{y}_{\mathbf{desired}}$ (output data set), is presented to the network by the teacher. With *a priori* knowledge of the environment, the teacher is able to provide the neural network with desired outputs for the set of training data. The teacher is often an uncontrollable or unknown part of the learning process. The aim in supervised learning is to make the neural network emulate the teacher. The actual output vector $\mathbf{y}_{\mathbf{actual}}$ is produced by the network, and compared with the desired outputs presented by the teacher. The error $(\mathbf{e} = \mathbf{y}_{\mathbf{desired}} - \mathbf{y}_{\mathbf{actual}})$ is then used to adjust the parameters of the network, so as to make the neural network emulate the teacher more and more closely. Usually, many training cycles or epochs (from a few to hundreds) are required to properly train the network. When the neural network is able to emulate the teacher well enough, the learning process is then completed. One of the most popular supervised learning methods is *backpropagation* (also known as the *backward error propagation*) (Werbos, 1974). The *gradient descent* (Boyd, 2004) and the (Levenberg, 1944, Marquardt, 1963) *Levenberg-Marquardt* methods are often used for neural network learning in conjunction with the backpropagation process to form two types of learning, they are called *Levenberg-Marquardt backpropagation* (Hagan, 1994) and *gradient descent backpropagation* (Jang, 1997).

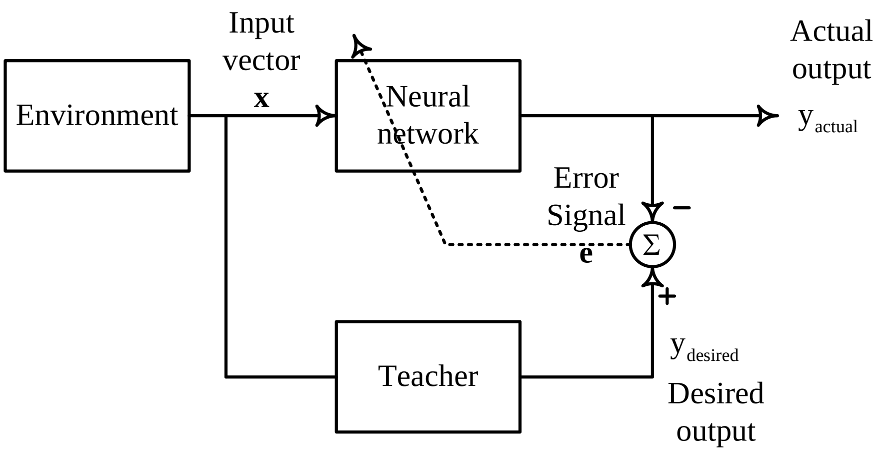

Figure 3.8: Structure of a supervised learning.

The Levenberg-Marquardt backpropagation method has been proven to be more reliable and faster than the gradient descent backpropagation method (Burns, 2001). In this thesis, the Levenberg-Marquardt backpropagation method is used for training the multilayer feedforward and Elman networks.

3.3 Fuzzy Logic

3.3.1 Fuzzy Sets

This section summarizes the basic concepts and notations of fuzzy set theory and fuzzy logic that will be needed in the following sections. Since research on the subject has been underway for over 30 years it is difficult to cover all aspects of developments in this area. A detailed treatment of the subject may be found in (Zadeh, 1965), (Zadeh, 1973) and (Zimmermann, 1985).

The idea of fuzzy sets is introduced by way of an example. Let X be the range of temperature values known as the *universe of discourse* (or more simply *universe*) and its elements be denoted as x. Let A be a set of high temperature values that are at least $30^{\circ}\mathrm{C}$ and $f_A(x)$ be the function called the characteristic function of A.

$$f_A(x): X \to 0,\ 1, \tag{3.2}$$

where,

$f_A(x) = 1$, if $x \in A$;

$f_A(x) = 0$, if $x \notin A$;

This set maps universe X to a set of two elements. For any element x of universe X, the characteristic function $f_A(x)$ is equal to 1 if x is ($\geq 30^{\circ}\mathrm{C}$) belonging to set A, and is equal to 0 if x is ($< 30^{\circ}\mathrm{C}$) not belonging to set A. A is commonly known as a *classical* or *crisp* set which has a "clear-cut" boundary. The characteristic function $f_A(x)$ is shown in Figure 3.9. It describes the crisp set of all temperature values greater than or equal to $30^{\circ}\mathrm{C}$.

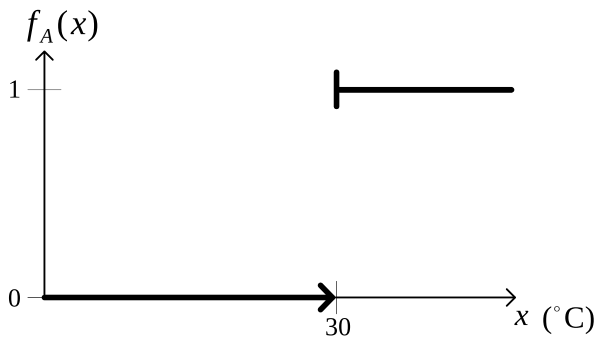

Figure 3.9: The characteristic function $f_A(x)$ of all temperature values greater than or equal to $30^\circ\mathrm{C}$.

In a crisp set, the almost identical elements like the temperature values of $29.9^\circ\mathrm{C}$ and $30.1^\circ\mathrm{C}$ are treated as being completely different. On the other hand, a *fuzzy* set is a set with a vague boundary. Each value of temperature is associated with a degree of membership. A *degree of membership* may assume values between 0 and 1. That is, the transition from "belonging to a set" to "not belonging to a set" is gradual. A fuzzy set *A* of universe *X* is defined by $\mu_A(x)$ called the *membership function* (MF) of x in *A*.

$$\mu_A(x) : X \rightarrow [0,1], \tag{3.3}$$

where

$\mu_A(x) = 1$ if x is totally in *A*;

$\mu_A(x) = 0$ if x is not in *A* at all;

$0 \langle \mu_A(x) \langle 1$ if x is partly in *A*.

Figure 3.10 shows the membership function $\mu_A(x)$ of all temperature values, e.g. the temperature $25^\circ\mathrm{C}$ with a value of 0.7. This means the temperature of $25^\circ\mathrm{C}$ corresponds to the property "high temperature" with a membership degree of 0.7 on a scale from 0 to 1. The closer the membership degree is to 1 the more strongly *x* satisfies the property "high temperature".

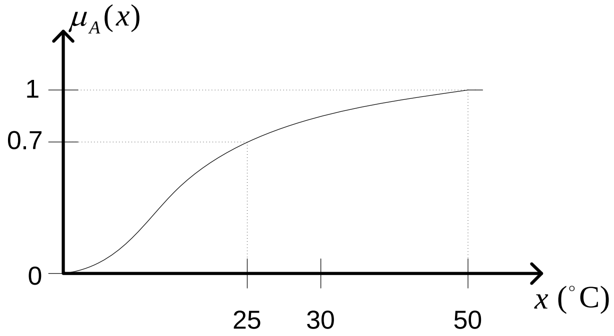

Figure 3.10: The membership function $\mu_A(x)$ of all temperature values

Obviously the definition of a fuzzy set is a natural extension of the definition of a classical set in which the characteristic function is permitted to have continuous values between 0 and 1. If the value of the membership function $\mu_A(x)$ is restricted to either 0 or 1, then *A* is reduced to a classical set, and $\mu_A(x)$ is the characteristic function of *A*.

3.3.2 Linguistic Variables and Linguistic Values

The following simple example serves as an introduction to the concept of *linguistic variables* and *linguistic values*.

In everyday communication we often use short sentences, which carry the same amount of information as their longer counterparts. When we say that "the weather is too hot" we actually mean that "the weather's temperature belongs to the too hot (very high) category." Even if we knew that the temperature was exactly 42°C, in everyday communication we prefer saying that "the weather is too hot," as we would assume that there is a common understanding what a *very high temperature* in weather terms means. The term *temperature* may attain two different values: numerical (42°C) and linguistic (too hot). Variables, for which values are words or sentences, rather than numbers, are called *linguistic variables*. In this example, the variable temperature may have *linguistic values* such as *very high* (*too hot*), *high*, *medium*, *low*, *and very low*. This is why linguistic values are sometimes referred to as fuzzy sets.

A fuzzy set is uniquely specified by its membership function. To describe membership functions more specifically, the nomenclature used in the literature (Jang, 1997) will be followed.

Support

The *support* of a fuzzy set *A* is the set of all points *x* in *X* such that $\mu_A(x) > 0$:

$$\text{support(A)} = \{\, x \mid \mu_A(x) > 0 \,\}. \tag{3.4}$$

Core

The *core* of a fuzzy set *A* is the set of all points *x* in *X* such that $\mu_A(x) = 1$:

$$\text{Core(A)} = \{\, x \mid \mu_A(x) = 1 \}. \tag{3.5}$$

Normality

A fuzzy set *A* is *normal* if its core is non-empty. In other words, we can always find a point $x \in X$ such that $\mu_A(x) = 1$.

Crossover points

A *crossover point* of a fuzzy set *A* is a point $x \in X$ at which $\mu_A(x) = 0.5$:

$$\text{crossover}(A) = \{\, x \mid \mu_A(x) = 0.5 \,\}. \tag{3.6}$$

Fuzzy singleton

A fuzzy set whose support is a single point in *X* with $\mu_A(x) = 1$ is called a *fuzzy singleton*

Corresponding to the ordinary set operations, i.e., *union, intersection* and *complement*, fuzzy sets have similar operations, which were initially defined by Zadeh in (Zadeh, 1965) . These fuzzy sets operations are *containment* (or *subset*), *union* (or *disjunction*), *intersection* (or *conjunction*), *complement* (or *negation*), *Cartesian product* and *co-product.*

As mentioned earlier, a fuzzy set is completely characterized by its MF. A more convenient way to define a MF is to express it as a mathematical formula.

3.3.3 Types of Membership Functions

Eight types of membership function (MF) will be described. They are bell MF, triangular MF, Gaussian MF, two-sided Gaussian MF, pi-shaped MF, product of two sigmoidal MFs, difference between two sigmoidal MFs, and trapezoidal MF.

The triangular MF, is expressed as:

$$f(x;a,b,c)=\begin{cases}0, & x\le \mathrm{a}\\ \dfrac{x-\mathrm{a}}{\mathrm{b}-\mathrm{a}}, & \mathrm{a}\le x\le \mathrm{b}\\ \dfrac{c-x}{c-b}, & \mathrm{b}\le x\le \mathrm{c}\\ 0, & \mathrm{c}\le x\end{cases} \tag{3.7}$$

where the parameters a, b and c describe the shape of the triangular MF.

The trapezoidal MF is expressed as:

$$f(x;a,b,c,\mathrm{d})=\begin{cases}0, & x\le \mathrm{a}\\ \dfrac{x-\mathrm{a}}{\mathrm{b}-\mathrm{a}}, & \mathrm{a}\le x\le \mathrm{b}\\ 1, & \mathrm{b}\le x\le c\\ \dfrac{d-x}{d-c}, & \mathrm{c}\le x\le d\\ 0, & \mathrm{d}\le x\end{cases} \tag{3.8}$$

where the shape of the trapezoidal MF is decided by the parameters a, b, c and d .

For the Gaussian MF, the expression is:

$$f(x; c, \sigma) = \mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}} \tag{3.9}$$

where the parameters c and σ decide the shape of the Gaussian MF.

The two-sided Gaussian MF is expressed as:

$$f(x; c_1, c_2, \sigma_1, \sigma_2) = \begin{cases} \mathrm{e}^{-\frac{(x-c_1)^2}{2\sigma_1^2}}, & x \le c_1 \\ 1, & c_1 < x < c_2 \\ \mathrm{e}^{-\frac{(x-c_2)^2}{2\sigma_2^2}}, & c_2 \le x \end{cases} \tag{3.10}$$

where the shape is decided by the parameters σ_1, c_1 and σ_2, c_2 which correspond to the widths and centres of the left and right half Gaussian functions.

The bell-shaped MF, is expressed as:

$$f(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}, \tag{3.11}$$

where the parameters *a*, *b*, and *c* describe the shape of bell-shaped MF.

The product of two sigmoidal MFs is expressed as:

$$f(x; a_1, c_1, a_2, c_2) = \frac{1}{(1 + \mathrm{e}^{-a_1(x-c_1)})(1 + \mathrm{e}^{-a_2(x-c_2)})} \tag{3.12}$$

where the parameters a_1, c_1, a_2 and c_2 describe the shapes of two sigmoid MFs.

The difference between two sigmoidal MFs is expressed as:

$$f(x;a_1,c_1,a_2,c_2)=\left|\frac{1}{(1+\mathrm{e}^{-a_1(x-c_1)})}-\frac{1}{(1+\mathrm{e}^{-a_2(x-c_2)})}\right| \tag{3.13}$$

where the parameters a_1, c_1, a_2 and c_2 describe the shapes of two sigmoid MFs

The pi-shaped MF is the product of Z shape and S shape functions. It is expressed as:

$$f(x;a,c)=\begin{cases} S(x;c-a,c), & x \le c \\ Z(x;c,c+a), & x > c \end{cases} \tag{3.14}$$

where c is the centre and a (>0) is the spread on each side of the MF.

3.3.4 Fuzzy *if-then* Rules

The goal of fuzzy systems is to mimic a human operator's action or to make humanlike decisions by using the knowledge about a target system (without knowing its model). This is achieved with *fuzzy if-then rules* (also known as *fuzzy rules, fuzzy implications,* or *fuzzy conditional statements*). For example

$$\textbf{\textit{If}}\ x \text{ is A } \textbf{\textit{then}}\ y \text{ is B} \tag{3.15}$$

where *x* and *y* are linguistic variables; A and B are linguistic values determined by fuzzy sets on (ranges of possible values) universe of discourses X and Y, respectively. Often the ***if*** part of the rule "x is A" is called the *antecedent* or *premise* part and the ***then*** part of the rule "*y* is B" is called the *consequence* or *conclusion* part.

Due to their concise form, fuzzy if-then rules are often employed to capture the imprecise modes of reasoning that play an important role in the human ability to make decision in an environment of uncertainty and imprecision.

The following examples describe the difference between classical and fuzzy rules

A classical ***if-then*** rule uses binary logic, for example,

Rule: 1
if the temperature is $\geq 30^{\circ}\mathrm{C}$ ***then*** the weather is hot
Rule: 2
if the temperature is $< 20^{\circ}\mathrm{C}$ ***then*** the weather is cold

In this example the linguistic variable temperature can have any numerical value between 0 and $50^{\circ}\mathrm{C}$, but the linguistic variable weather can only take either value *hot* or *cold*. In other words, classical rules are expressed in the ***true*** or ***false*** form.

A fuzzy ***if-then*** rule of the above example can be expressed as:

Rule: 1
if the temperature is high ***then*** the weather is hot
Rule: 2
if the temperature is low ***then*** the weather is cold

Here the linguistic variable *temperature* also has the range (universe of discourse) between 0 and $50^{\circ}\mathrm{C}$, but this range includes fuzzy sets, such as *low*, *medium*, and *high*. The linguistic variable *weather* may include fuzzy sets as *cold*, *warm* and *hot*. Thus fuzzy rules relate to fuzzy sets.

Another form of fuzzy if-then rule, proposed by (Takagi, 1983), has fuzzy sets involved only in the premise part. The consequent part of Takagi-Sugeno (TS for short) fuzzy if-then rule is a real-valued function of the input variables instead of a fuzzy set. A TS fuzzy if-then can be expressed in the following general form:

$$\textit{if } x_1 \text{ is } A_1, \text{ and} \ldots, \text{ and } x_n \text{ is } A_n, \textit{ then } y = f(x_1, x_2, \ldots, x_n), \tag{3.16}$$

where $f(\cdot)$ is a real-valued function.

3.3.5 Fuzzy Reasoning

Fuzzy reasoning, also known as approximate reasoning, is an inference (inferencing is the process of reasoning about a particular state of the underlying system, using all available knowledge to produce a best estimate of the output) procedure that derives conclusions from a set of fuzzy if-then rules and known facts. Before describing fuzzy reasoning, we need to understand the concept behind the *compositional rule of inference*, which can be found in (Zadeh, 1973).

Several types of fuzzy reasoning have been used in the literature, a sample of these can be found in (Lee, 1990). In general, the process of fuzzy reasoning can be divided into four steps:

1. **Degree of compatibility:** Compare the known facts with the antecedents of fuzzy rules to find the degrees of compatibility with respect to each antecedent MF.

2. **Firing Strength:** Combine degrees of compatibility with respect to antecedent MFs in a rule using fuzzy AND (intersection) or OR (union) operators to form a firing strength that indicates the degree to which the antecedent part of the rule is satisfied.

3. **Qualified (induced) consequent MFs:** Apply the firing strength to the consequent MF of a rule to generate a qualified consequent MF.

4. **Overall output MF:** Aggregate all the qualified consequent MFs to obtain an overall output MF.

3.3.6 Fuzzy Inference Systems

The *fuzzy inference system* is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning (Jang, 1997). It has found successful applications in a wide variety of fields such as automatic control, data classification, decision analysis, expert systems, time series prediction, robotics, and pattern recognition (Jamshidi, 1997). Because of its multidisciplinary nature, the fuzzy inference system is known by numerous other names, such as *fuzzy expert system* (Kandel, 1992), *fuzzy model* (Sugeno, 1988), *fuzzy associative memory* (Kosko, 1991), and simply *fuzzy system*.

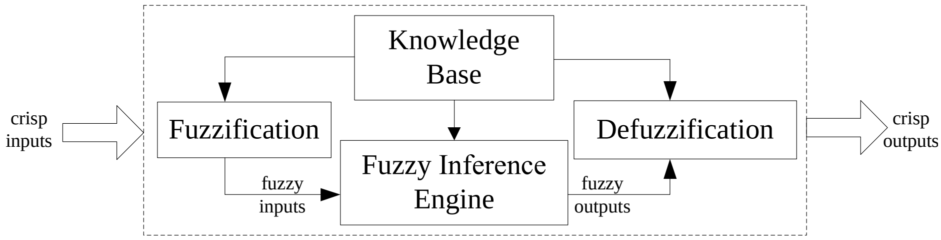

Figure 3.11: The basic components of a fuzzy inference system

A fuzzy inference system (FIS) consists of four functional blocks as shown in Figure 3.11.

1. **<u>Fuzzification:</u>** transforms the crisp inputs into degrees of match with linguistic values.

2. <u>Knowledge base:</u> consists of a rule base and a database. A rule base contains a number of fuzzy if-then rules. A database defines the MFs of the fuzzy sets used in the fuzzy rules.

3. **Fuzzy inference engine:** performs the inference operations on the rules.

4. **Defuzzification:** transforms the fuzzy results of the inference into a crisp output.

As indicated in Figure 3.11, the FIS can be envisioned as involving a knowledge base and a *processing stage* (consisting of fuzzification, fuzzy inference engine and defuzzication stages). The knowledge base provides MFs and fuzzy rules needed for the process. In the processing stage, numerical crisp variables are the input of the system. These variables are passed through a fuzzification stage where they are transformed to linguistic variables, which become the fuzzy input for the fuzzy inference engine. This fuzzy input is transformed by the rules of the fuzzy inference engine to fuzzy output. The linguistic results are then changed by a defuzzication stage into numerical values that become the output of the system.

Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, a fuzzy inference system can be classified into three types: The Tsukamoto-type FIS (Tsukamoto, 1979), Mamdani-type FIS (Mamdani, 1975) and Takagi-Sugeno-type FIS (Sugeno, 1988). An in depth analysis of each of these fuzzy inference systems can be found in (Jang, 1997).

CHAPTER 4 : NEURAL FUZZY MODELING

4.1 Introduction

Modeling, in a general sense, refers to the establishment of a description of a physical system (a plant, a process, etc.) in mathematical terms, which characterizes the input-output behavior of the underlying system. The real system can then be explained, controlled, simulated, predicted or even improved. The development of reliable and comprehensible models is the main objective in systems modeling.

To describe a physical system, such as a circuit or a microprocessor, one has to use a mathematical formula or equation that can represent both qualitatively (patterns) and quantitatively (numbers). Such a formulation is, by nature, a mathematical representation, called a *mathematical model*, of the physical system in interest.

Most physical systems are extremely difficult to model by an accurate and precise mathematical formula or equation due to the complexity of the system structure, nonlinearity, uncertainty, randomness, etc. Therefore, *approximate modeling* is often necessary and indeed practical in real world applications.

Intuitively, approximate modeling is always possible. However, the key questions are what kind of approximation is good (where the sense of "goodness" has to first be defined) and how to formulate such a good approximation in modeling a system such that it is mathematically rigorous and can produce satisfactory results in both theory and application.

From the decriptions given in the last two chapters, it is clear that fuzzy logic can provide a promising alternative to mathematical modeling for many physical systems particularly those that are too vague or too complicated to be described by simple mathematical equations. When fuzzy logic is employed, the fuzzy membership functions are used as approximation measures, leading to the so-called *fuzzy systems modeling*.

Ever since fuzzy systems were applied in industrial applications, developers have realized that the construction of a well performing fuzzy system is not always easy. The problem of finding appropriate membership functions and fuzzy rules is often a tiring process of trial and error. Therefore the idea of applying learning algorithms to fuzzy systems was considered early.

Fuzzy systems can easily handle imprecise data, and explain their decision in the context of the available facts in linguistic form but they are not capable of learning from the environment. On the other hand, neural networks are capable of learning but they cannot interpret imprecise data that can be helpful in making decisions. Therefore, fuzzy systems and neural networks are often considered as two technologies that complement each other.

Interest in *hybrid intelligent systems* that combine neural networks and fuzzy systems has grown in the last few years and a summary may be obtained from (Abraham, 2001). Among these hybrid intelligent systems, there is one important system called the *neural fuzzy* (or sometimes called *neuro-fuzzy*) system that can learn from its environment and then make decisions. A neural fuzzy system is based on fuzzy inference system, which is trained by a learning algorithm derived from artificial neural network theory. A detailed treatment of neuro-fuzzy systems can be found in (Yager, 1994) and (Ying, 2000).

4.2 Modeling the Transformer's Top-oil Temperature Using an Adaptive Network-based Fuzzy Inference System

An adaptive network here refers to as a special kind of feedforward neural network with supervised learning capability. It is a network structure consisting of nodes and directional links through which the nodes are connected. In this network, part or all of the nodes are adaptive, which means that the output of each node depends on the parameter(s) pertaining to it. The parameter(s) should be changed to minimize a prescribed error measure according to the network's learning algorithm.

This Section will explain the modeling aspects of the transformer's top-oil temperature prediction using the hybrid neural fuzzy learning technique ANFIS (Nguyen, 2002). The ANFIS stands for Adaptive Network based Fuzzy Inference System, which was developed by Jang (Jang, 1993). The ANFIS is based on the architecture of the Takagi-Sugeno-type fuzzy inference system. ANFIS is one of the most popular and well documented neural fuzzy systems, which has a good software support (The MathWorks, 1999). Jang (Jang, 1997) presented the ANFIS architecture and application examples in modeling a nonlinear function, a dynamic system identification and a chaotic time series prediction. Given its potential in building neural fuzzy models with good prediction capabilities (Cai, 2003) and (Mastacam, 2005), the ANFIS architecture was chosen for modeling of this work.

ANFIS has a similar structure to a real multilayer feed forward neural network as shown in Figure 4.1. Each node performs a particular function (node function) on incoming signals as well as providing a set of parameters pertaining to this node. The nature of node functions may vary from node to node, and the choice of each node function depends on the overall input-output function that the adaptive network is required to carry out. Note that unlike the real multilayer feedforward neural network, the links in an ANFIS only indicate the flow direction of signals between nodes and no weights are associated with the links.

To reflect different adaptive capabilities, both circle and square nodes are used in an ANFIS. A square node (adaptive node) has parameters while a circle node (fixed) has none.

As for the transformer's top-oil temperature prediction, the ANFIS has two input variables $\boldsymbol{\theta}_{amb}$ and $\boldsymbol{I}_{load}$; and each input variable has three membership functions (MF)

A_1, A_2, and A_3 and B_1, B_2, and B_3 respectively, then a Takagi-Sugeno-type fuzzy *if-then* rule is set up as:

$$Rule_n = if\ \theta_{amb} \text{ is } A_i \text{ and } I_{load} \text{ is } B_i\ then\ f_n = p_n\theta_{amb} + q_n I_{load} + r_n \tag{4.1}$$

where, I is an index I = 1, 2, 3 and n is the rule number; and p_n, q_n and r_n are the linear parameters of function f_n. Using the grid partitioning method (Jang, 1993), two input variables and three MFs for each input variable, the ANFIS for the transformer's top-oil temperature prediction will have 9 ($3^2 = 9$) Takagi-Sugeno fuzzy rules. Some layers of ANFIS have the same number of nodes and the nodes in the same layer have similar functions. For convenience, the output of the ith and nth node in layer l is denoted as $O_{l,i}$ and $O_{l,n}$ respectively. The function of each layer is described as follows.

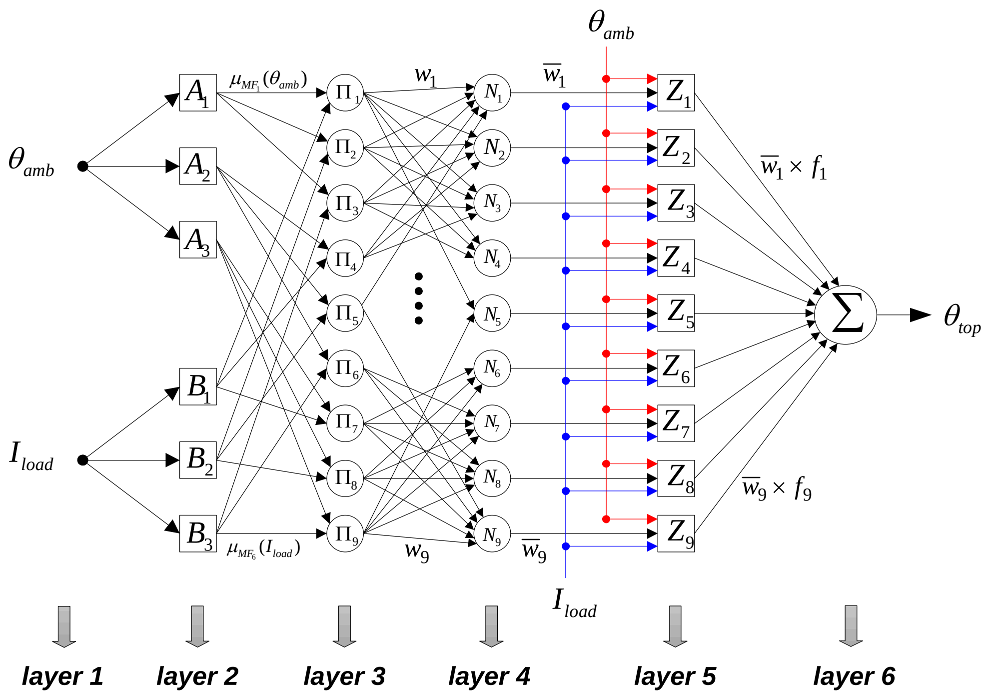

Figure 4.1: An ANFIS model for the transformer's top-oil temperature prediction

Layer 1 is the **input layer**. Nodes in this layer simply pass incoming signals (crisp values) to Layer 2. That is,

$$O_{1,\theta_{amb}} = \theta_{amb} \qquad (4.2)$$

$$O_{1,I_{load}} = I_{load}$$

Layer 2 is the **fuzzification layer**. Every node *i* in this layer is an adaptive node. The output of nodes in this layer are presented as,

$$O_{2,i} = \mu_{A_i}(\theta_{amb}), \text{ for } i = 1, 2, 3, \text{ or} \qquad (4.3)$$

$$O_{2,i} = \mu_{B_i}(I_{load}), \text{ for } i = 1, 2, 3.$$

Where θ_{amb} (or I_{load}) is the input to node *i* and A_i (or B_i) is a linguistic label (such as "small", "medium", or "large") associated with this node. In other words, $O_{2,i}$ is the membership grade of a fuzzy set *M* (= A_1, A_2, A_3, B_1, B_2, or B_3) and it specifies the degree to which the given input θ_{amb} (or I_{load}) satisfies the quantifier *M*. Here the membership function for *M* can be any appropriate parameterized membership function introduced in Section 3.3.3, for instance, the membership function is the generalized bell function:

$$\mu_M(\theta_{amb}) = \frac{1}{1 + \left[\left(\dfrac{\theta_{amb} - c_i}{a_i}\right)^2\right]^{b_i}}, \text{ or} \qquad (4.4)$$

$$\mu_M(I_{load}) = \frac{1}{1 + \left[\left(\dfrac{I_{load} - c_i}{a_i}\right)^2\right]^{b_i}}.$$

Where a_i, b_i and c_i are the parameters that control, respectively, the centre, width and slope of the bell-shaped function of node *i*. As the values of these parameters change the bell-shaped function varies accordingly, thus exhibiting various forms of membership functions for fuzzy set *M*. Parameters in this layer are referred to as *premise parameters.*

Layer 3 is the **rule layer**. Every node in this layer is a fixed node and labeled as Π_n. Each node in this layer corresponds to a single Takagi-Sugeno fuzzy rule. A rule node receives inputs from the respective fuzzification nodes and calculates the firing strength of the rule it represents. Each node output is the product of all incoming signals:

$$O_{3,n} = w_n = \mu_{A_i}(\theta_{amb}) \times \mu_{B_i}(I_{load}),\; i = 1, 2, 3. \tag{4.5}$$

where w_n represents the firing strength, or the truth value, of the *n*th rule; and $n = 1, 2, \cdots, 9$ is the number of Takagi-Sugeno fuzzy rules.

Layer 4 is the **normalization layer.** Every node in this layer is a fixed node and labeled as N_n. Each node in this layer receives inputs from all nodes in the rule layer, and calculates the *normalized firing strength* of a given rule. The normalized firing strength of the *n*th node is the ratio the *n*th rule's firing strength to the sum of all rule's firing strengths:

$$O_{4,n} = \overline{w}_n = \frac{w_n}{\sum_{n=1}^{9} w_n} \quad n = 1, 2, 3, \cdots, 9 \tag{4.6}$$

The number of nodes in this layer is the same the number nodes in layer 3. That is 9 nodes. The outputs of this layer are called normalized firing strengths.

Layer 5 is the **defuzzification layer.** Every node in this layer is an adaptive node and labeled as Z_i. Each node in this layer is connected to the respective normalization node, and also receives initial inputs, θ_{amb} and I_{load}. A defuzzification node calculates the weighted consequent value of a given rule as,

$$O_{5,n} = \overline{w}_n f_n = \overline{w}_n (p_n \theta_{amb} + q_n I_{load} + r_n) \tag{4.7}$$

where $\overline{w}_n$ is a normalized firing strength from layer 3 and { p_n, q_n, r_n } is the parameter set of this node. These parameters are also called *consequent parameters.*

Layer 6 is represented by a single **summation node.** This single node is a fixed node and labeled as Σ. This node calculates the sum of outputs of all defuzzification nodes and produces the overall ANFIS output, θ_{top},

$$\theta_{top} = O_{6,1} = \sum_{n=1}^{9} \overline{w}_n f_n = \frac{\sum_{n=1}^{9} w_n f_n}{\sum_{n=1}^{9} w_n} \tag{4.8}$$

Thus, the ANFIS model for the transformer's top-oil temperature as shown in Figure 4.1 is functionally equivalent to a first order polynomial Takagi-Sugeno fuzzy model. It is not necessary to have any prior knowledge of rule consequent parameters for an ANFIS. An ANFIS uses a hybrid-learning rule with a combination of a gradient descent (to tune the membership function parameters) and a least squares estimate algorithm (to learn the rule consequent parameters) (Jang, 1993).

ANFIS only supports Sugeno-type systems, and ANFIS must have the following properties (Demuth, 1998):

- Be first or zeroth order Sugeno-type systems.

- Have a single output, obtained using weighted average defuzzification. All output MFs must be the same type and either be linear or constant.

- Have no rule sharing. Different rules cannot share the same output MF, namely the output MFs must be equal to the number of rules.

- Have unity weight for each rule.

4.3 Hybrid Learning of an ANFIS

The hybrid learning algorithm of ANFIS in Matlab can be explained as follows: each epoch is composed from a forward pass and a backward pass.

Forward pass

In the forward pass, a training set of input patterns [as input vectors i.e. ambient temperature ($\boldsymbol{\theta}_{amb}$) and load current ($\boldsymbol{I}_{load}$)] is presented to the ANFIS, node outputs are calculated on layer by layer basis, and rule consequent parameters are identified by the least-squares estimator. In the TS type fuzzy inference, an output vector, top-oil temperature ($\boldsymbol{\theta}_{top}$), is a linear function. Thus, given the values of the membership parameters (for example triangular MF which has 3 parameters a, b and c) and a training set of 1702 input [$\boldsymbol{\theta}_{amb}$ $\boldsymbol{I}_{load}$] and output [$\boldsymbol{\theta}_{top}$] patterns, one can form 1702 linear equations in terms of the consequent parameters (p, q and r) as:

$$\begin{cases} \theta_{top\_p}(1) = \overline{w}_1(1) f_1(1) + \overline{w}_2(1) f_2(1) + \cdots + \overline{w}_n(1) f_n(1) \\ \theta_{top\_p}(2) = \overline{w}_1(2) f_1(2) + \overline{w}_2(2) f_2(2) + \cdots + \overline{w}_n(2) f_n(2) \\ \qquad \vdots \\ \theta_{top\_p}(m) = \overline{w}_1(m) f_1(m) + \overline{w}_2(m) f_2(m) + \cdots + \overline{w}_n(m) f_n(m) \end{cases} \tag{4.9}$$

or

$$\begin{cases} \theta_{top\_p}(1) = \overline{w}_1(1)[p_1\theta_{amb}(1) + q_1 I_{load}(1) + r_1] + \\ \qquad\qquad \overline{w}_2(1)[p_2\theta_{amb}(1) + q_2 I_{load}(1) + r_2] + \cdots + \\ \qquad\qquad \overline{w}_n(1)[p_n\theta_{amb}(1) + q_n I_{load}(1) + r_n] \\ \theta_{top\_p}(2) = \overline{w}_1(2)[p_1\theta_{amb}(2) + q_1 I_{load}(2) + r_1] + \\ \qquad\qquad \overline{w}_2(2)[p_2\theta_{amb}(2) + q_2 I_{load}(2) + r_2] + \cdots + \\ \qquad\qquad \overline{w}_n(2)[p_n\theta_{amb}(2) + q_n I_{load}(2) + r_n] \\ \qquad\qquad\qquad \vdots \\ \theta_{top\_p}(m) = \overline{w}_1(m)[p_1\theta_{amb}(m) + q_1 I_{load}(m) + r_1] + \\ \qquad\qquad \overline{w}_2(m)[p_2\theta_{amb}(m) + q_2 I_{load}(m) + r_2] + \cdots + \\ \qquad\qquad \overline{w}_n(m)[p_n\theta_{amb}(m) + q_n I_{load}(m) + r_n] \end{cases} \tag{4.10}$$

where m = input/output patterns = 1702; n = number of nodes in the rule layer = 9 and θ_{top_p} is the predicted top-oil temperature of the ANFIS when inputs θ_{amb} and I_{load} are presented to it.

Equation (4.10) can be written in a matrix form, such as

$$\boldsymbol{\theta}_{top\_p} = \mathbf{A}\,\boldsymbol{k} \tag{4.11}$$

where $\boldsymbol{\theta}_{top\_p}$ is a $m \times 1$ = 1702×1 predicted top-oil temperature vector,

$$\boldsymbol{\theta}_{\boldsymbol{top\_p}} = \begin{bmatrix} \theta_{top\_p}(1) \\ \theta_{top\_p}(2) \\ \vdots \\ \theta_{top\_p}(m) \end{bmatrix} \tag{4.12}$$

$\mathbf{A}$ is a $m \times n$ (1+ number of input variables) = 1702×27 matrix,

$\mathbf{A}$ =

$$\begin{bmatrix} \overline{w}_1(1) & \overline{w}_1(1)\theta_{amb}(1) & \overline{w}_1(1)I_{load}(1) & \cdots & \overline{w}_n(1) & \overline{w}_n(1)\theta_{amb}(1) & \overline{w}_n(1)I_{load}(1) \\ \overline{w}_1(2) & \overline{w}_1(2)\theta_{amb}(2) & \overline{w}_1(2)I_{load}(2) & \cdots & \overline{w}_n(2) & \overline{w}_n(2)\theta_{amb}(2) & \overline{w}_n(2)I_{load}(2) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \overline{w}_1(m) & \overline{w}_1(m)\theta_{amb}(m) & \overline{w}_1(m)I_{load}(m) & \cdots & \overline{w}_n(m) & \overline{w}_n(m)\theta_{amb}(m) & \overline{w}_n(m)I_{load}(m) \end{bmatrix}$$

(4.13)

And $\boldsymbol{k}$ is an $n(1+\text{number of input variables})\times 1$ = 27×1 vector of unknown consequent parameters,

$$\boldsymbol{k} = [\,p_1 \quad q_1 \quad r_1 \quad p_2 \quad q_2 \quad r_2 \quad \cdots \quad p_n \quad q_n \quad r_{\mathrm{n}}\,]^T \tag{4.14}$$

In this case, the number of input – output patterns $m = 1702$ used in training is greater than the number of consequent parameters $n(1+\text{number of input variables})$ = 27. It means that we are dealing with an over-determined problem, and thus an exact solution to Equation (5.20) may not even exist. Instead, one should find a least-square estimate of $\boldsymbol{k}$, $\boldsymbol{k}^*$, that minimizes the squared error $\left\|\boldsymbol{Ak}-\boldsymbol{\theta}_{\boldsymbol{top\_p}}\right\|^2$. This is achieved using the pseudo-inverse technique:

$$\boldsymbol{k}^* = (\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{\theta}_{\boldsymbol{top\_p}}, \tag{4.15}$$

where $\boldsymbol{A}^{\boldsymbol{T}}$ is the transpose of $\boldsymbol{A}$, and $(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}$ is the pseudo-inverse of $\boldsymbol{A}$ if $(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})$ is non-singular.

As soon as the rule consequent parameters are established, we can compute an actual network output vector $\boldsymbol{\theta}_{\boldsymbol{top}}$, and the error vector $\boldsymbol{e}$, can be determined as,

$$\boldsymbol{e} = \boldsymbol{\theta}_{top\_p} - \boldsymbol{\theta}_{\boldsymbol{top}} \tag{4.16}$$

Backward pass

In the backward pass, the back-propagation algorithm is applied. The error signals are propagated back, and the antecedent parameters are updated according to the chain rule.

For instance, consider a correction applied to parameter a of the bell-shaped membership function used in node A_1 of Figure 5.6. The chain rule can be expressed from Equation (5.12) as,

$$\Delta a = -\eta \frac{\partial E}{\partial a} = -\eta \frac{\partial E}{\partial e} \times \frac{\partial e}{\partial \theta_{top}} \times \frac{\partial \theta_{top}}{\partial (\overline{w}_i f_i)} \times \frac{\partial (\overline{w}_i f_i)}{\partial \overline{w}_i} \times \frac{\partial \overline{w}_i}{\partial w_i} \times \frac{\partial w_i}{\partial w_{A_1}} \times \frac{\partial w_{A_1}}{\partial a}, \tag{4.17}$$

where η is the learning rate, and E is the instantaneous value of the squared error for the ANFIS output neuron, i.e.,

$$E = \frac{1}{2} e^2 = \frac{1}{2} (\theta_{top\_p} - \theta_{top})^2 \tag{4.18}$$

Thus,

$$\Delta a = -\, \eta (\theta_{top\_p} - \theta_{top})(-1)\, f_i \times \frac{\overline{w}_i (1 - w_i)}{w_i} \times \frac{w_i}{w_{A_1}} \times \frac{\partial w_{A_1}}{\partial a} \tag{4.19}$$

or

$$\Delta a = \eta (\theta_{top\_p} - \theta_{top})\, f_i \overline{w}_i (1 - \overline{w}_i) \times \frac{1}{w_{A_1}} \times \frac{\partial w_{A_1}}{\partial a}$$

where

$$\frac{\partial w_{A_1}}{\partial a} = \frac{1}{\left[1 + \left(\frac{\theta_{amb} - a}{c}\right)^{2b}\right]^2} \times \frac{1}{c^{2b}} \times 2b \times (\theta_{amb} - a)^{2b-1} \times (-1) = w_{A_1}^2 \times \frac{2b}{c} \times \left(\frac{\theta_{amb} - a}{c}\right)^{2b-1}$$

Similarly, the corrections applied to parameters *b* and *c* can also be obtained.

Root Mean Square Error (RMSE)

In order to evaluate the predicting accuracy of different models, the transformer's top-oil temperature prediction uses the root mean square error (RMSE) to measure the difference between the predicted and measured temperature values.

For example with top-oil temperature,

$$\theta_{top\_p} = \text{predicted top-oil temperature}$$

and

$$\theta_{top\_m} = \text{measured top-oil temperature}$$

and the RMSE is given by the following equation,

$$\text{RMSE} = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(\theta_{top\_m} - \theta_{top\_p})^2} \tag{4.20}$$

Where P is the total number of temperature samples.

Although RMSE is a good tool to measure the prediction accuracy but sometimes it cannot give a true meaning of how well a model performs. The RMSE value can be small but the model does not work properly. The model comparison in this thesis is conducted on the basis of both peaks of error and RMSE results.

CHAPTER 5 : TESTING AND COMPARISON

5.1 Introduction

This Chapter presents in detail the calculation and simulation of: ANFIS, IEEE, multilayer feedforward and Elman neural networks. This chapter also compares the results of all four models and identifies the most successful model.

The parameters that are routinely monitored by staff of the Power Engineering Department at Monash University include the: top-oil temperature, bottom oil temperature, ambient temperature and load current of the distribution transformers. These parameters are measured every 30 minutes for the 8 MVA OA-cooled transformer (transformer 1) and every 15 minutes for the 27 MVA FA-cooled transformer (transformer 2). Load current figures were rounded to their closest integer value in Amperes. The top-oil, bottom-oil and ambient temperature measurements were rounded to one tenth of a degree, Celsius. For transformer 1, the measured top-oil temperature, bottom-oil temperature, ambient temperature, and load current are labeled as $\theta_{\mathrm{T1\_top\_m}}$, $\theta_{\mathrm{T1\_bot\_m}}$, $\theta_{\mathrm{T1\_amb\_m}}$ and $I_{\mathrm{T1\_load\_m}}$ respectively, whereas for transformer 2 these parameters are labeled as $\theta_{\mathrm{T2\_top\_m}}$, $\theta_{\mathrm{T2\_bot\_m}}$, $\theta_{\mathrm{T2\_amb\_m}}$ and $I_{\mathrm{T2\_load\_m}}$ respectively. For more detail of these parameters refer to Figure A.1 to A.8. in Appendix A.

The most important aspect in building a successful neural fuzzy or neural network based transformer temperature model is the selection of the input variables. There is no guaranteed rule that one could follow in this process. The selection of input variables for this project has been carried out almost entirely by trail and error. However, the ambient temperature and load current are understood to be the main factors that affect the top-oil temperature of transformers (Lesieutre, 1997). Therefore, in all the models considered herein, the inputs include the ambient temperature and load current signals while the output will be the top-oil or bottom-oil temperature signal unless stated otherwise. In order to evaluate the prediction ability of each model, the available transformer data sets were divided into two halves, the first half was used as the training set, and the remaining half as the test set.

5.2 The Transformer Temperatures Prediction Using the ANFIS Model

This section presents the training and testing of the ANFIS model for transformer temperature prediction with batch (off line) learning. In a conventional fuzzy inference system, an expert who is familiar with the system to be modeled decides the number of rules. In the following simulation, however, no expert is available and as a consequence the parameters assigned to each input and output variable were determined by trial and error. For the 8 MVA transformer, the training data set covers from day 1 to 35.5, whereas the test set covers from day 35.5 to 71. For the 27 MVA transformer, the training data set was from day 1 to 4.5 and from day 4.5 to 9 was for test data set.

There are many parameters one can select to obtain better results in ANFIS. For the most common case, these parameters are: the number and type of membership function for each input, the output membership function type (either ‘linear’ or ‘constant’), the training epoch number, the training error goal, the initial step size, the step size decrease rate and the step size increase rate. In addition to the parameter selection one can also ensure that appropriate test data are used to detect overfitting of the training data set. The test data have the same format as the training data. Overfitting can be detected when the test error (difference between the measured and predicted outputs) starts increasing while the training error is still decreasing.

The ANFIS model used in the preparation of this thesis is shown in Figure 5.1.

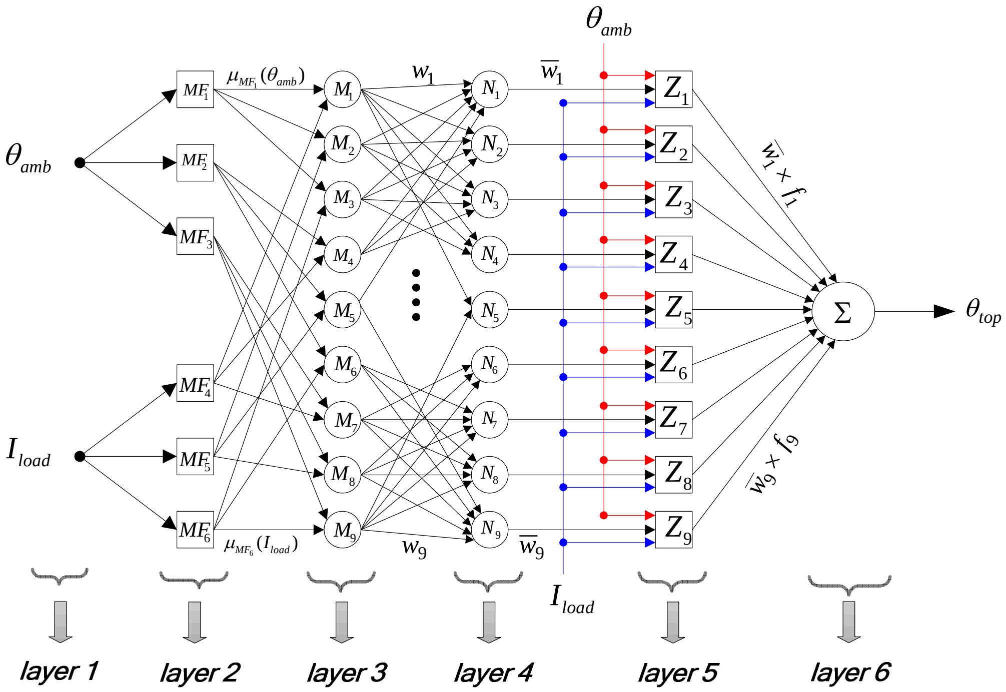

Figure 5.1: The ANFIS model used to predict top-oil temperature for the case of three MFs.

5.2.1 The ANFIS Top-oil Temperature Prediction with Two Input Variables (8 MVA Transformer)

A practical approach is to choose the smallest number of membership functions (MFs). Thus the number of MFs of each input is assigned to two. The effect of different MFs of input for predicting the top-oil temperature will be discussed in this section. The names of these MFs are bell MF, Gaussian MF, two-sided Gaussian MF, triangular MF, trapezoidal MF, product of two sigmoid MFs, difference between two sigmoidal MFs, and pi-shaped MF.

5.2.1.1 Standard Data Set – Bell-shaped Membership Function

Prediction with the bell-shaped MF

First the bell MF was chosen as MFs for this ANFIS. There were two inputs, θ_{amb} and I_{load}. Each input has two MFs which were named as *low* and *high*. The initial step size was 0.01. The comparison of the predicted and measured top-oil temperature is shown in Figure 5.2. The comparison of peak error, RMSE (root mean square error) and the training step sizes for the bell MFs using the ANFIS model are illustrated in Figure 5.3 and 5.4. The changes in the bell MFs before and after training are shown in Figure 5.5(a) to (d).

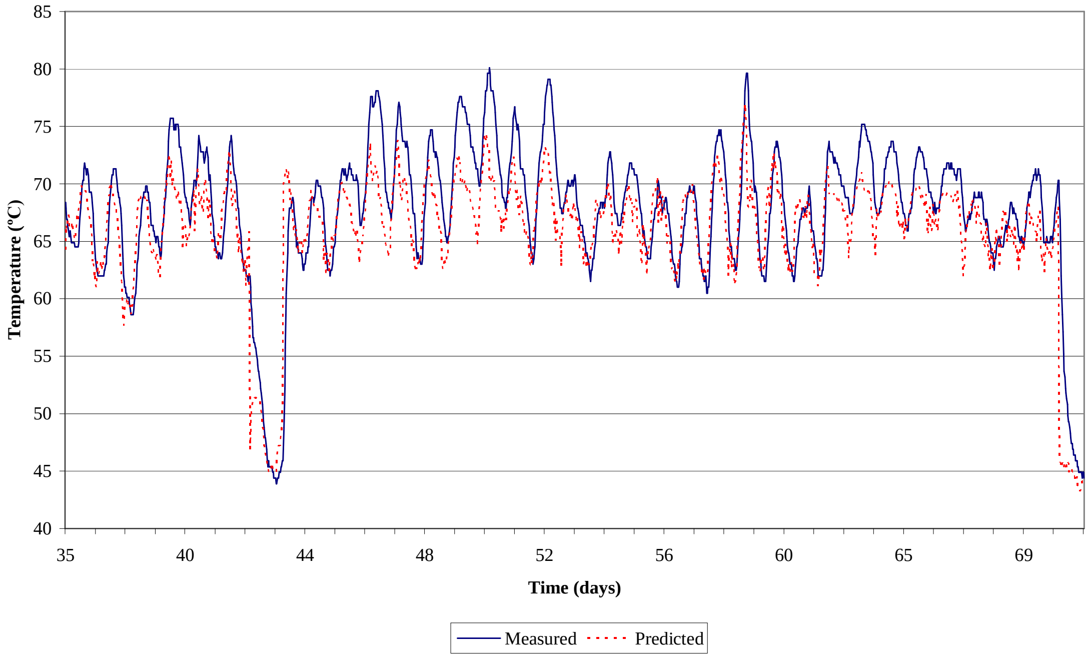

Figure 5.2: Waveform of the measured and predicted top-oil temperature by using the ANFIS model with the bell MFs.

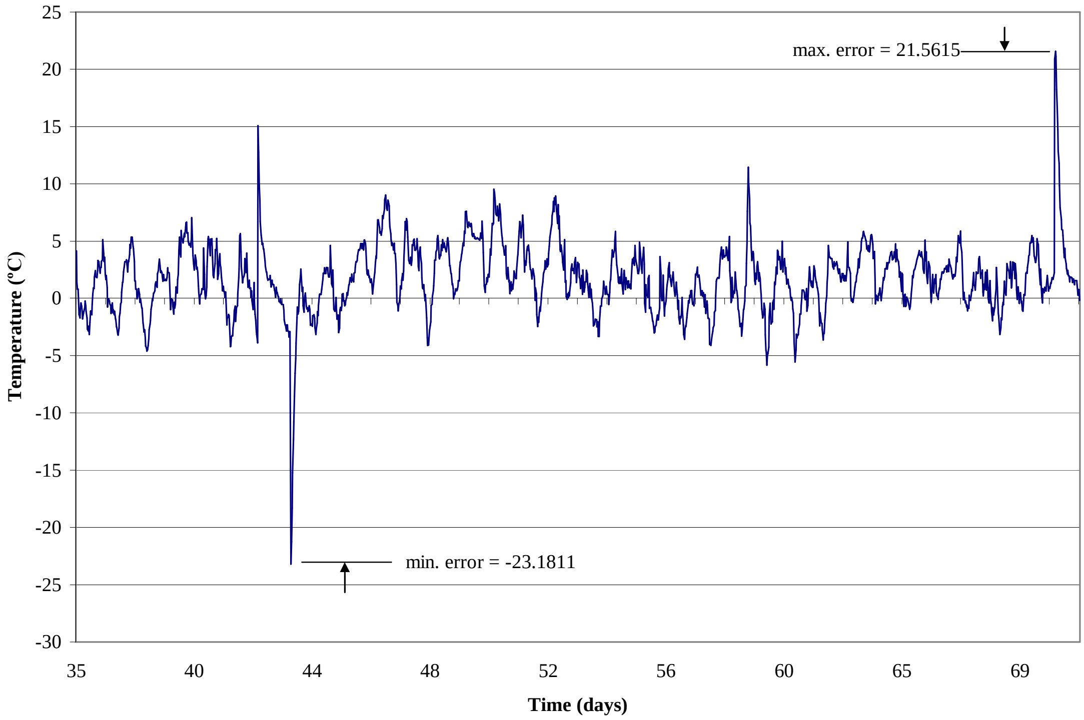

Figure 5.3: An error waveform of the measured and predicted top-oil temperature (transformer 1) using the ANFIS model with the bell MFs.

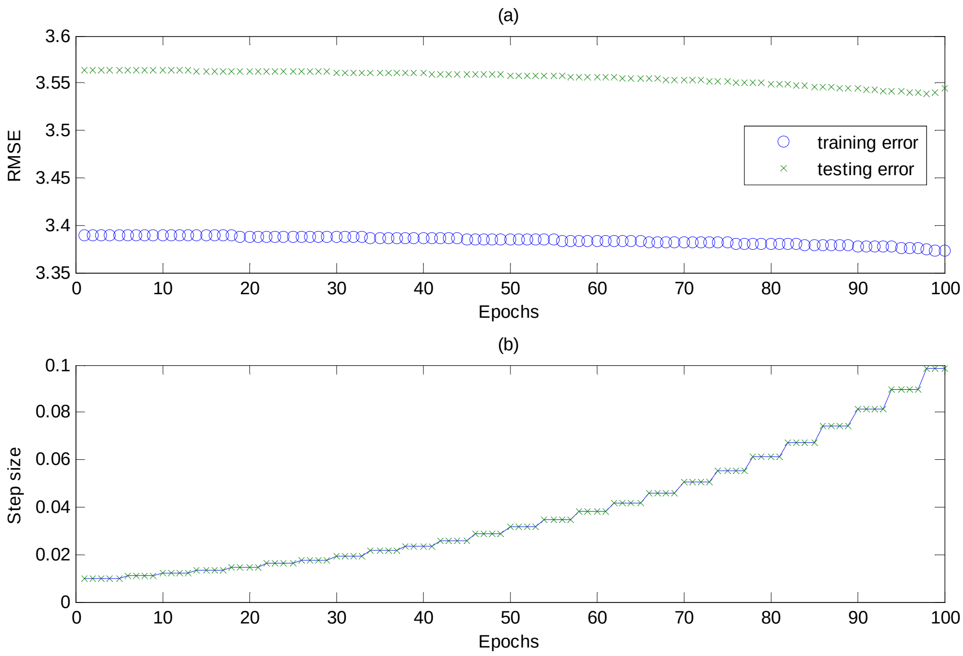

Figure 5.4: (a) Training and testing RMSE curves; (b) Training step sizes of the bell MFs.

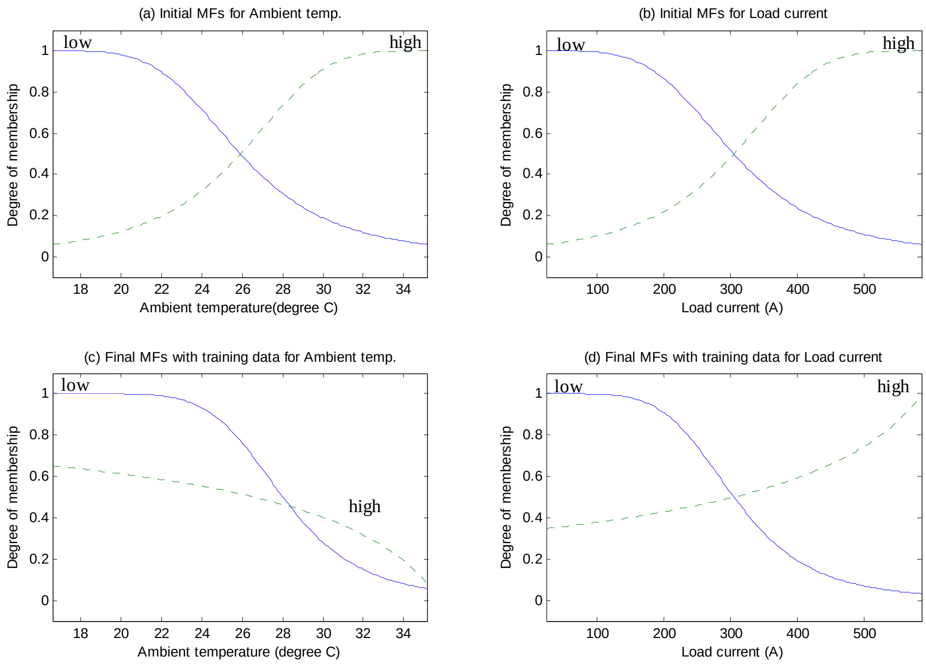

Figure 5.5: The changes in the bell MFs before and after training.

The above presentation, for the bell MF, has shown a number of graphs detailing features of the simulation conducted. The critical parameters (root-mean-square error, largest positive and negative difference) may be readily tabulated (table 5.1 on page 59) so that for other MFs only the changes in MFs before and after training graphs are presented.

5.1.1.2 Comparison between Membership Functions

Prediction with the Triangular MF

The triangular MF was chosen to predict the top-oil temperature of the 8 MVA transformer. The changes in triangular MFs before and after training are shown in Figure 5.6(a) to (d).

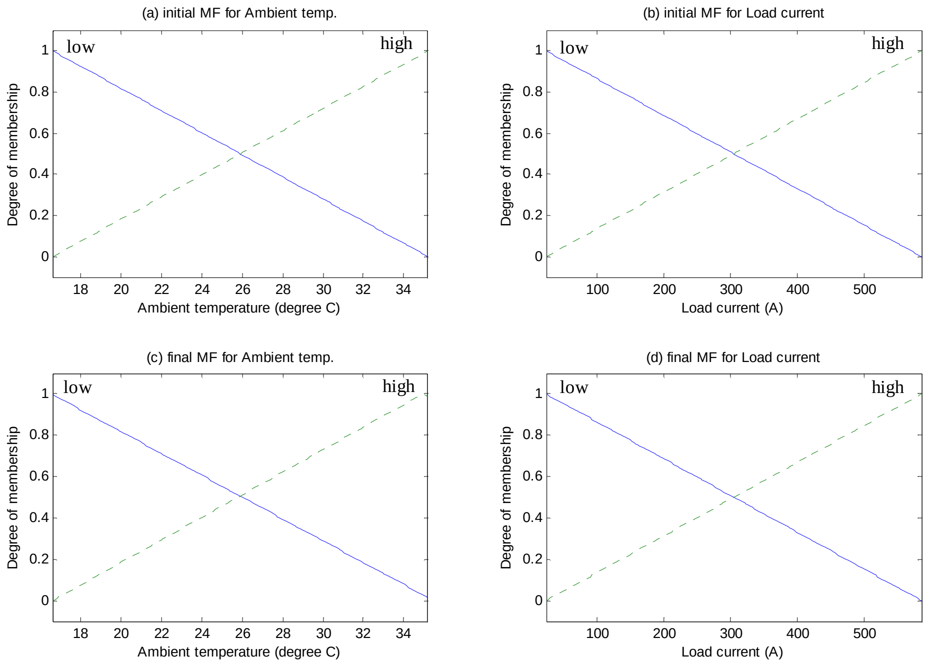

Figure 5.6: The changes in the triangular MFs before and after training.

Prediction with the Gaussian MF

The Gaussian MF was used to predict the top-oil temperature of the 8MVA transformer. The MFs before and after training are shown in Figure 5.7(a) to (d).

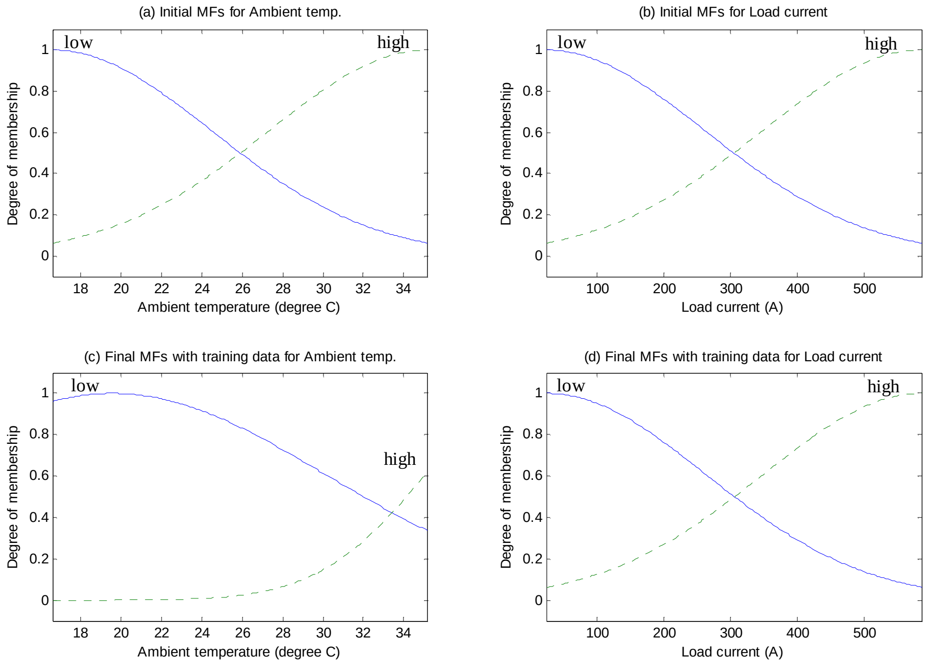

Figure 5.7: The changes in the Gaussian MFs before and after training.

Prediction with the Two-Sided Gaussian MF

The two-sided Gaussian membership function was used to predict top-oil temperature of the 8 MVA transformer. Figure 5.8(a) to (d) show the two-sided Gaussian MFs before and after training.

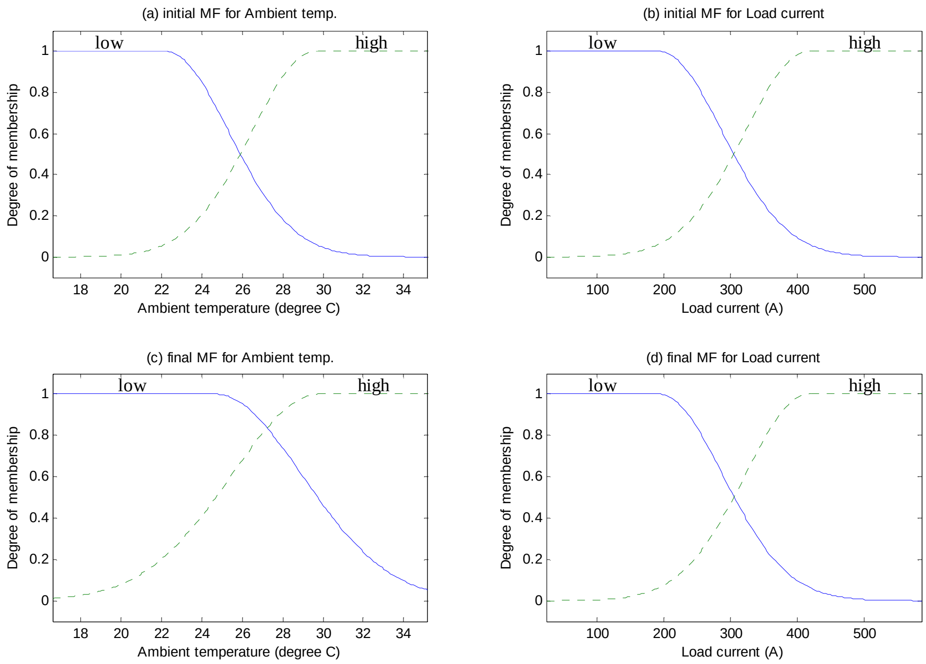

Figure 5.8: The changes in the two-sided Gaussian MFs before and after training.

Prediction with the product of Two Sigmoid MFs

The product of the two sigmoid membership functions was introduced to predict the top-oil temperature of the 8MVA transformer. Figure 5.9(a) to (d) show the product of two sigmoid MFs before and after training.

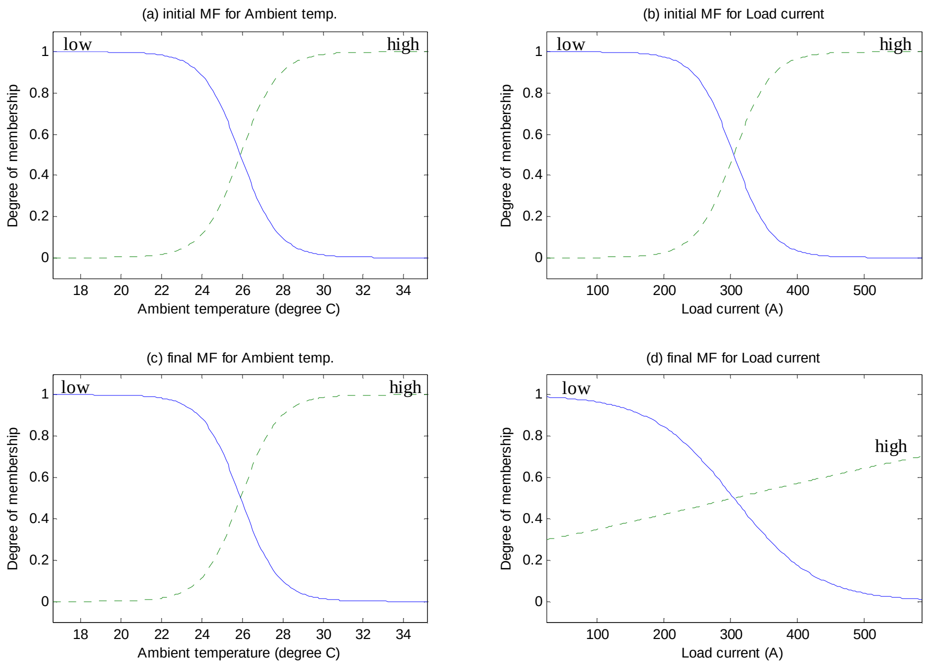

Figure 5.9: The changes in the product of the two sigmoid MFs before and after training.

Prediction with the Trapezoidal MF

The trapezoidal membership function was used to predict the top-oil temperature of the 8MVA transformer. Figure 5.10(a) to (d) show the trapezoidal MFs before and after training.

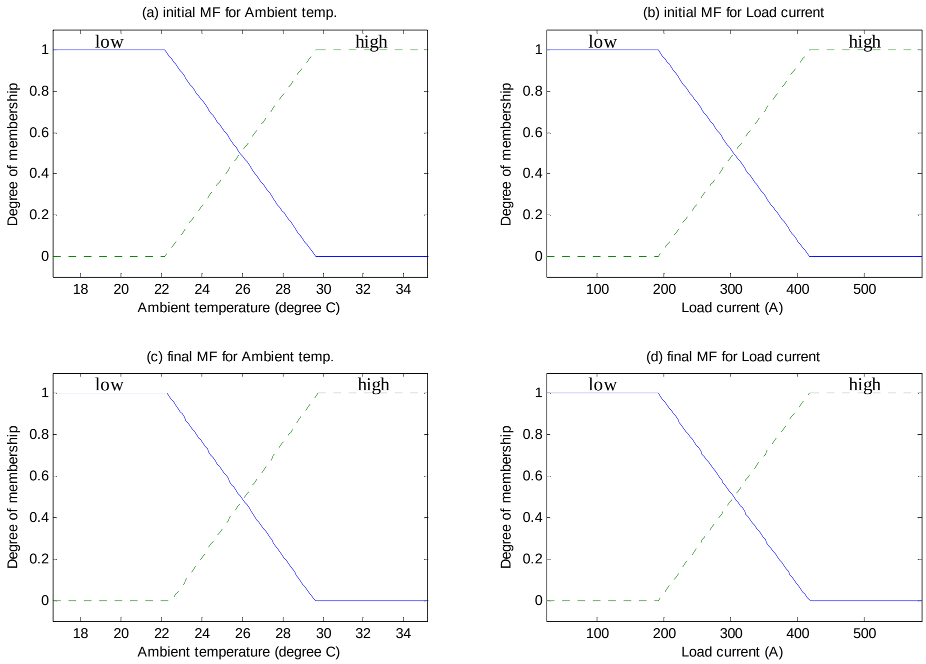

Figure 5.10: The changes in the trapezoidal MFs before and after training.

Prediction with the Difference between Two Sigmoidal MFs

The difference between two sigmoidal membership functions was used to predict to top-oil temperature of the 8MVA transformer. Figure 5.11(a) to (d) show the changes in the two sigmoidal MFs before and after training.

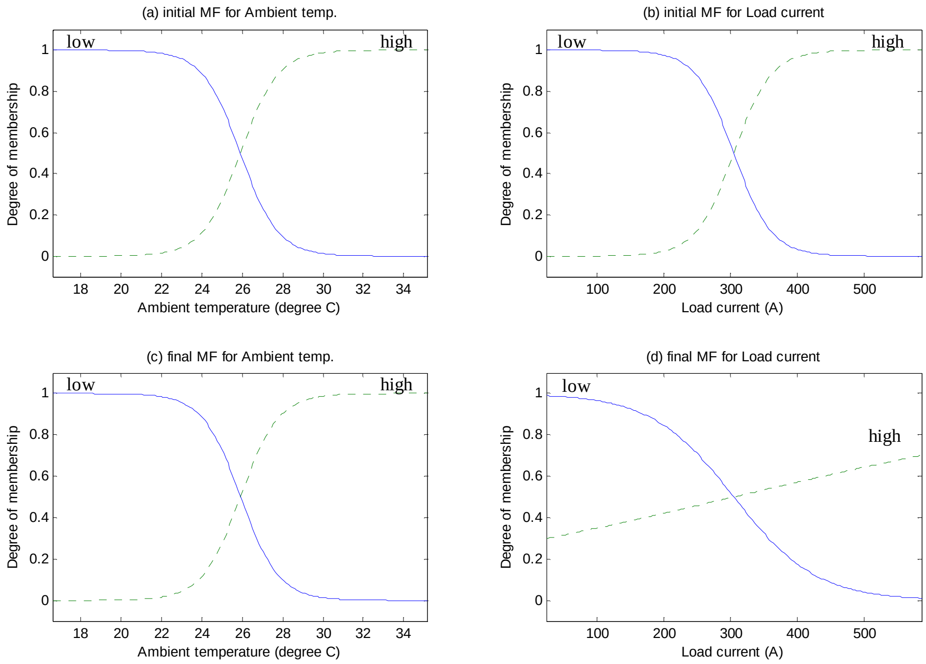

Figure 5.11: The changes in the difference between two sigmoidal MFs before and after training.

Prediction with the Pi-Shaped MFs

The pi-shaped membership function was used to predict the top-oil temperature of the 8MVA transformer. Figure 5.12(a) to (d) show the pi-shaped MFs before and after training.

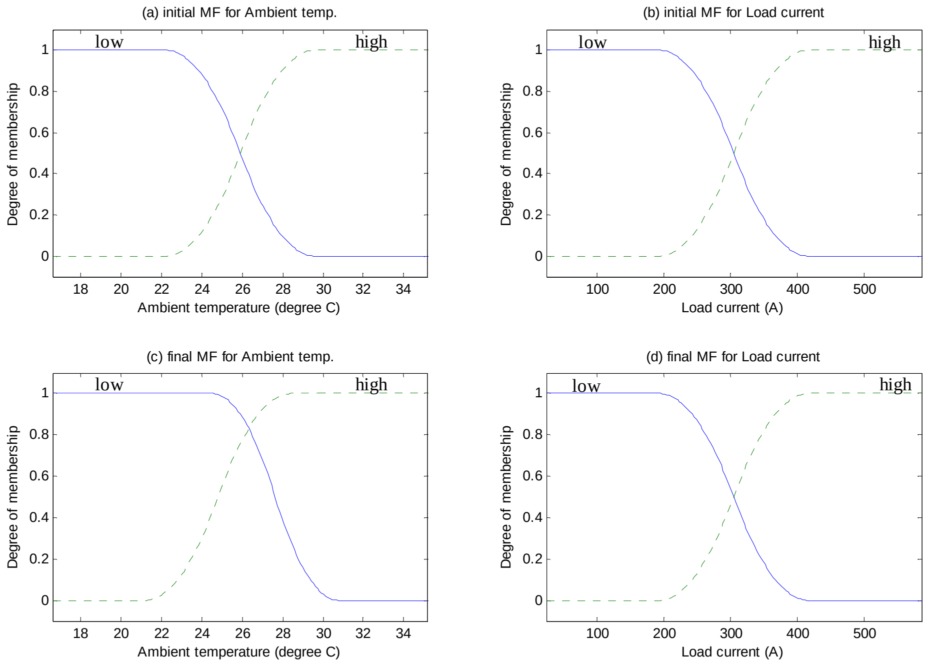

Figure 5.12: The changes in the pi-shaped MFs before and after training.

Summary of All Membership Functions

All RMSE and peak errors (minimum and maximum) results, processed with different type of membership functions, are listed in descending order in Table 5.1 for comparison purposes. It was found that the top-oil temperature prediction of the 8 MVA transformer do not vary to any great extent between the various MFs however the best prediction was obtained when using the Gaussian MF. The bell MF, the triangular MF, and the two-sided Gaussian MF are only slightly poorer than the Gaussian MF. The trapezoidal and the pi-shaped are poorer still than the above-mentioned MFs. The poorest MFs are the difference between two sigmoidal and the product of two sigmoid. From Figure 5.9(d) and 5.11(d), the (*high*) MFs for the load current have led to values (degree of membership) away from 1, the results appear worse. In contrast to Figure 5.5(c) and 5.7(c) where the (*high*) MFs for the ambient temperature have led to values away from 1, but the results appear to be better. This indicated that the load current is the most important predictor than the ambient temperature.

Name of MFs	RMSE	Min	Max
Product of two sigmoid	3.5730	-23.4103	21.5172
Difference between two sigmoidal	3.5730	-23.4103	21.5172
Pi-shaped	3.5727	-23.7039	21.4434
Trapezoidal	3.5698	-23.6164	21.4316
Two-sided Gaussian	3.5606	-23.3889	22.0064
Triangular	3.5443	-23.7103	21.5504
Bell	3.5390	-23.1811	21.5615
Gaussian	3.5376	-23.2740	21.5182

Table 5.1: The RMSE between the measured and predicted top-oil temperature of the 8 MVA transformer with all types of MFs.

5.2.1.3 Step Size

As shown in Table 5.1, the most of the MFs give a satisfactory result under the following conditions: initial step size = 0.01, step size decrease rate = 0.9, step size increase rate = 1.1; epochs = 100. The step size decrease of 0.9 and increase rate of 1.1 were chosen according to the two heuristic rules (Jang, 1997).

By varying the initial step size it may be possible to achieve a better RMSE. However, if the initial step size is large, convergence will initially be very fast, but the RMSE curve will oscillate about the optimum as shown in Figure 5.13 (a) and (b). Though an initial step size of 0.01 was chosen more or less arbitrarily, the results shown in Figure 5.13(a) indicate that an initial step size of 0.04 will achieve a smaller RMSE and does not lead to oscillation about the optimum point; therefore 0.04 is considered a better choice for this case.

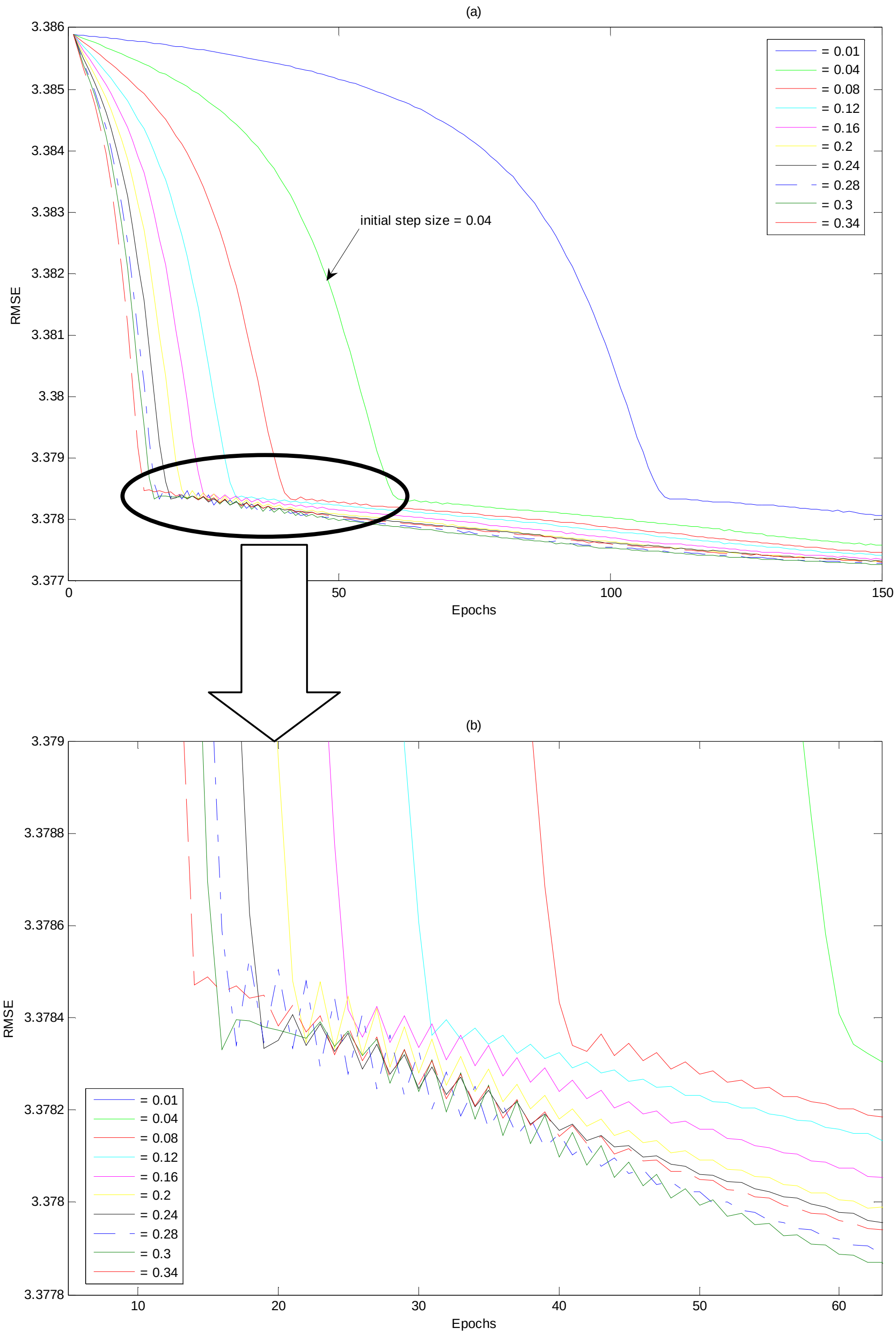

Figure 5.13: (a) RMSE curves with different initial step sizes (from 0.01 to 0.34) of Gaussian MFs; (b) Osciallation occured when initial step size is increased.

5.2.1.4 Epoch Numbers

The epoch number was set to 150 for the comparison between different. To test the validity of this, the Gaussian MF has been further studied using various epoch numbers. Figure 5.14(a) shows the waveforms of training and testing error. It is obvious that RMSE waveforms start descending before 50 epochs. After 70 epochs, the curves of RMSE tend to stabilize in value and any improvement in RMSE is small. As a practical consideration, increasing the epoch number simply consumes more time for computation: thus it is enough to train within 70 epochs. It can be observed that when the step size increases, RMSE decreases consistently as can be appreciated by comparing Figure 5.14(a) with (b).

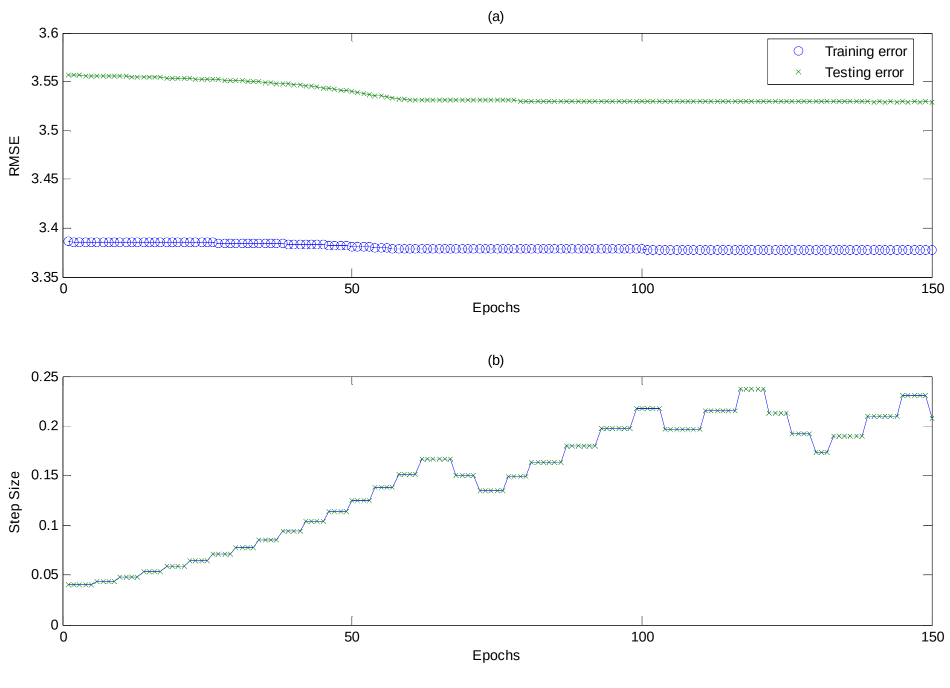

Figure 5.14: (a) The RMSE training and testing curves; (b) Training step sizes of the Gaussian MFs.

A list of the RMSE values between the measured and predicted of top-oil temperature with various training epoch number is shown in Table 5.2. As the training epoch number increases, the forecast of top-oil temperature with ANFIS becomes better, however after 70 epochs there is little improvement in the predicted values. It is therefore concluded that 70 epochs should be sufficient for training purposes and that the use of 150 epochs for the comparison of MFs is valid.

Epoch number	**RMSE**	**Min**	**Max**
20	3.5536	-23.2140	21.4800
50	3.5398	-23.2695	21.5179
70	3.5299	-23.2437	21.4646
150	3.5293	-23.2176	21.4588
200	3.5293	-23.2110	21.4547
400	3.5293	-23.2110	21.4547
600	3.5293	-23.2110	21.4547

Table 5.2: RMSE and peaks values of the predicted top-oil temperature with the different number of training epochs for 2 Gaussian MFs in each input case.

5.2.1.5 Number of Membership Functions for Each Input

The effect of the number of Gaussian MFs for each input is discussed in this section. The number of Gaussian MFs for each input was set to 2 (see Figure 5.7), 3, 4, 5 and 6 Gaussian MFs separately. Figure 5.15 to 5.18 show the changes of MFs before and after training for 3 (*low*, *medium* and *high*), 4 (*low*, *medium*, *high* and *very high*), 5 (*very low*, *low*, *medium*, *high* and *very high*) and 6 (*very low, low*, *medium*, *high, very high* and *extremely high*) Gaussian MFs respectively. For more details on the use of linguistic values (*low*, *high*, etc) refer to Section 3.3.2.

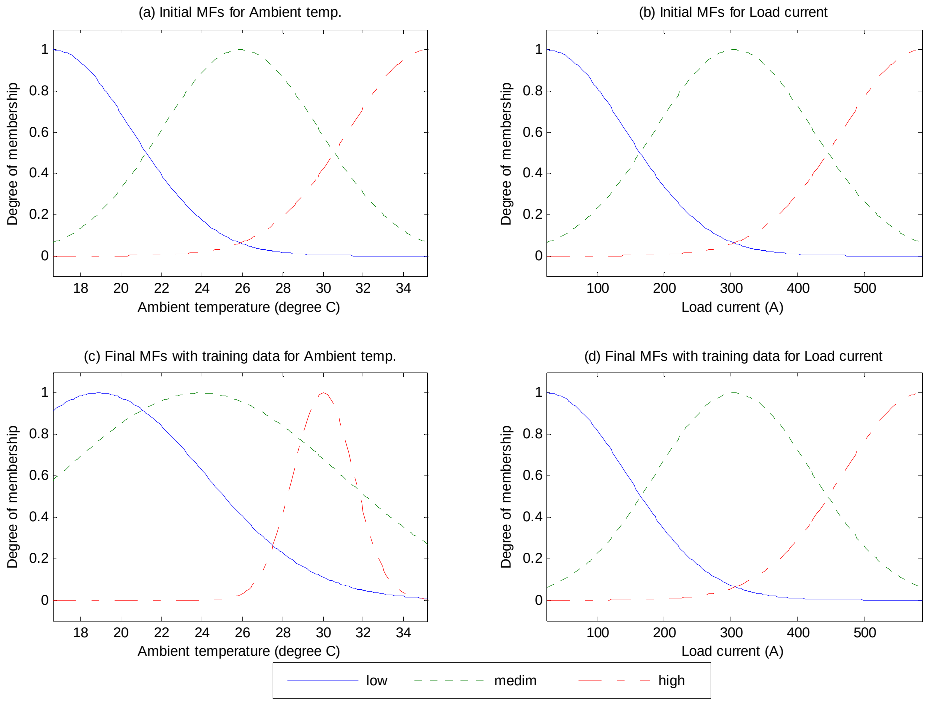

Figure 5.15: The changes in the 3 Gaussian MFs for each input before and after training.

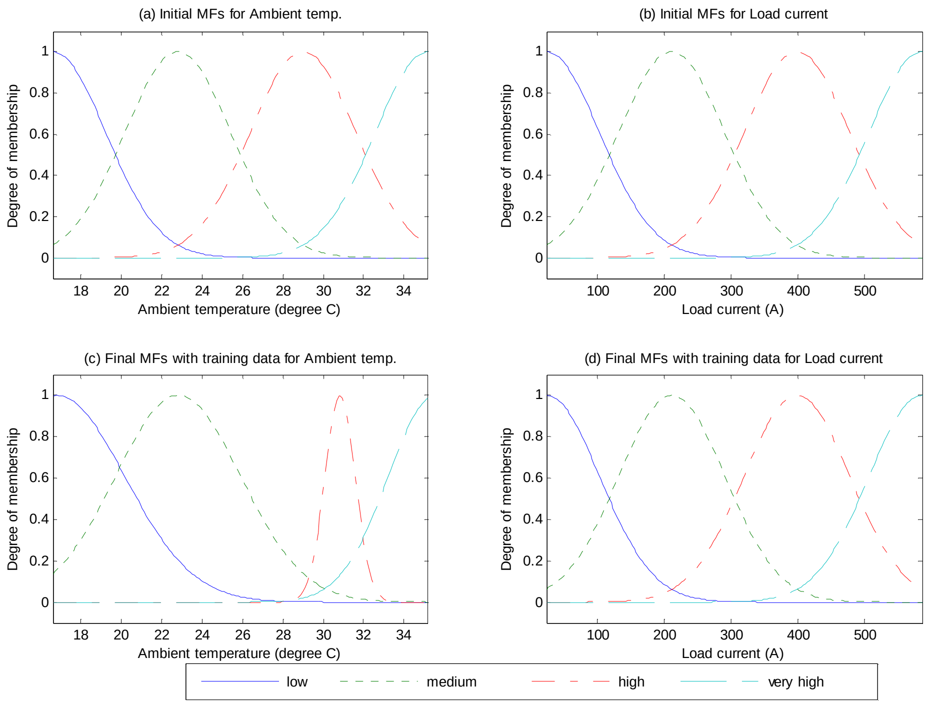

Figure 5.16: The changes in the 4 Gaussian MFs for each input before and after training.

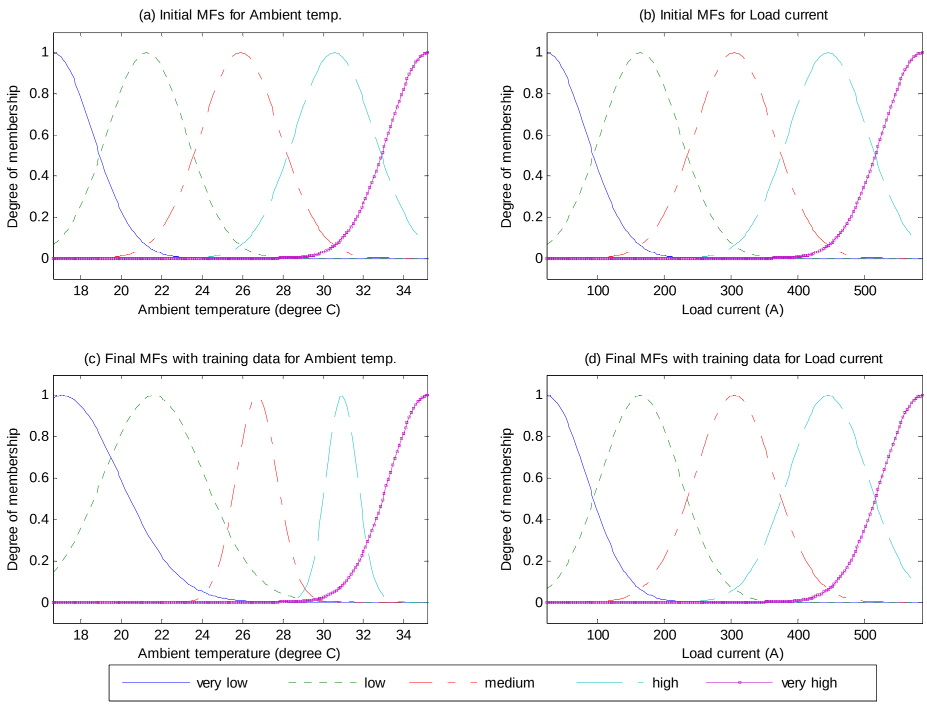

Figure 5.17: The changes in the 5 Gaussian MFs for each input before and after training.

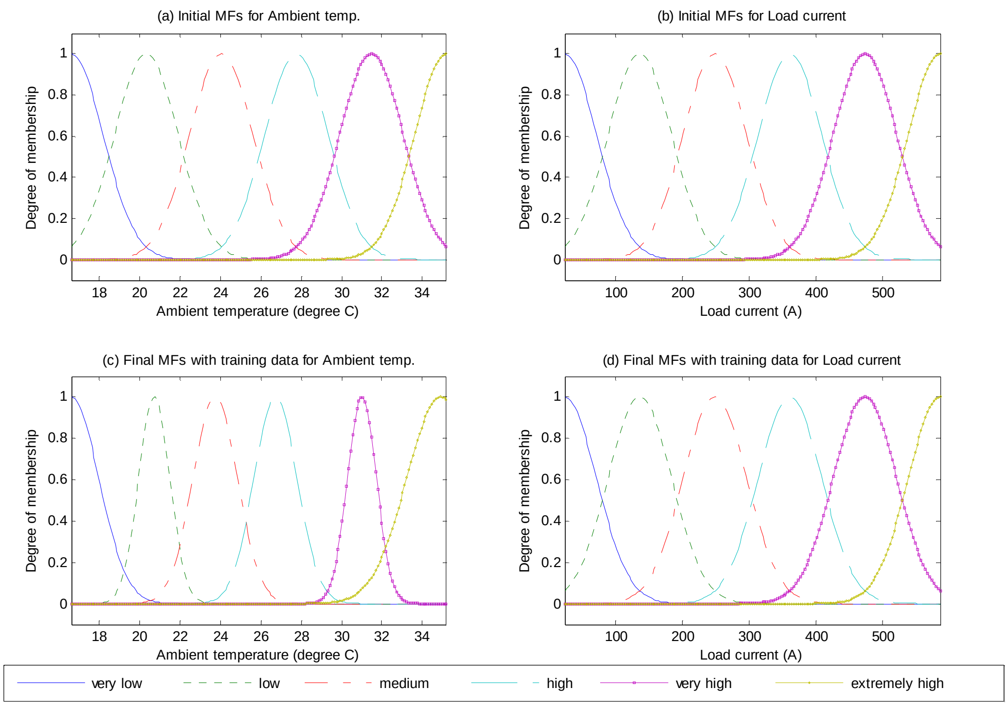

Figure 5.18: The changes in 6 Gaussian MFs for each input before and after training.

A list of RMSE of the predicted top-oil temperature with the different number of Gaussian MFs for each input can be seen in Table 5.3. Intuitively, one would expect more parameters would result in greater accuracy. But in this case when the number of MFs for each input increases from 3 to 6, the ANFIS model produces redundancy for the structure of data, hence the RMSE and peaks of error values increase.

Membership Number	RMSE	Min	Max
2	3.5299	-23.2437	21.4646
3	3.6033	-22.9690	28.6498
4	3.6881	-22.7558	21.3954
5	3.9091	-43.6313	21.5300
6	8.8467	-312.5036	21.2901

Table 5.3: The RMSE and peaks error values with the different number of Gaussian MFs for each.

5.2.1.6 Optimization Method Used in ANFIS Training

ANFIS uses either a hybrid learning algorithm or the back-propagation method to identify the MF parameters of the output. A combination of least-squares and back-propagation gradient descent methods can be used for training fuzzy inference system (Sugeno type FIS) membership function parameters to model a given set of input/output data or just the back-propagation method.

The RMSE of top-oil temperature of the 8 MVA transformer as predicted by using the back-propagation method and by using the hybrid learning algorithm are shown in Table 5.4. From this table, the capability of the ANFIS model in predicting top-oil temperature (8 MVA transformer) using the hybrid learning algorithm is much better than when using the back-propagation method. This is because the hybrid method comprises of back-propagation and Least-Square methods.

Method	RMSE	Min	Max
Back-propagation	9.6917	-41.8128	46.9765
Hybrid	3.5299	-23.2437	21.4646

Table 5.4: The RMSE and peaks error values by the back-propagation method and the hybrid learning algorithm.

5.2.1.7 Output Membership Function Type

ANFIS usually uses *linear* or *constant* to identify the output MF type. The *linear* or the *constant* can be used for training fuzzy inference system MF parameters to model a given set of input/output data.

In the *constant* output MF, a typical fuzzy rule has the following form:

if x is A and y is B then $z = k$

where A and B are the fuzzy sets while k is an exact constant.

In the *linear* output MF, it has the following format:

If x is A and y is B then $z = px + qy + r$

where p, q and r are constants.

The RMSE values of the predicted top-oil temperature by *linear* and *constant* output MF are shown in Table 5.5.

The types of output MF	**RMSE**	**Min**	**Max**
The *constant* output MF	3.5713	-23.5180	21.2769
The *linear* output MF	3.5299	-23.2437	21.4646

Table 5.5: RMSE of the predicted top-oil temperature by the *linear* and *constant* output.

From Table 5.5, it is evident that the effect of predicting top-oil temperature by the output MF type as *linear* is better than by the output MF type as *constant*.

5.2.1.8 ANFIS Model Validation

As mentioned in Section 5.1, one half of the total samples are used as the training data and the other half the test (validation) data. The test data are to detect any overfitting of the training data set. The test data has the same format as the training data. Overfitting can be detected when the testing error starts increasing while the training error is still decreasing.

Figure 5.19(a) shows the RMSE curves of the test and training data. Overfitting cannot be detected because when the training error slightly decreases from 3.39 to 3.38, the test error also descends slightly from 3.56 to 3.55. The changes in the Gaussian MFs of the training and test data before and after training are shown in Figures 5.20(a) to (f).

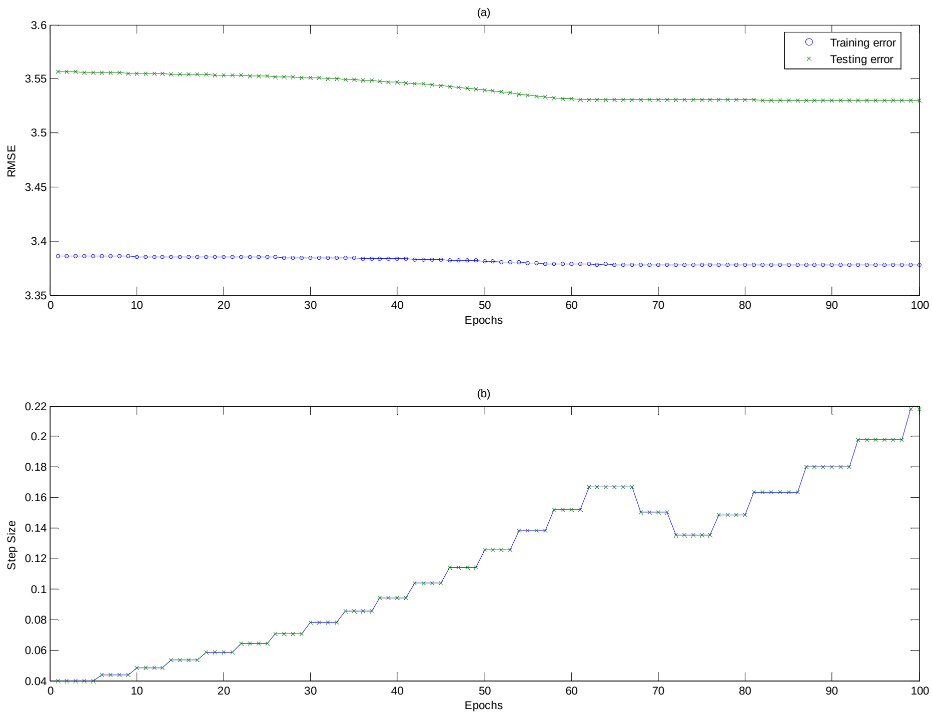

Figure 5.19: (a) RMSE curves; (b) Training step size of Gaussian MFs

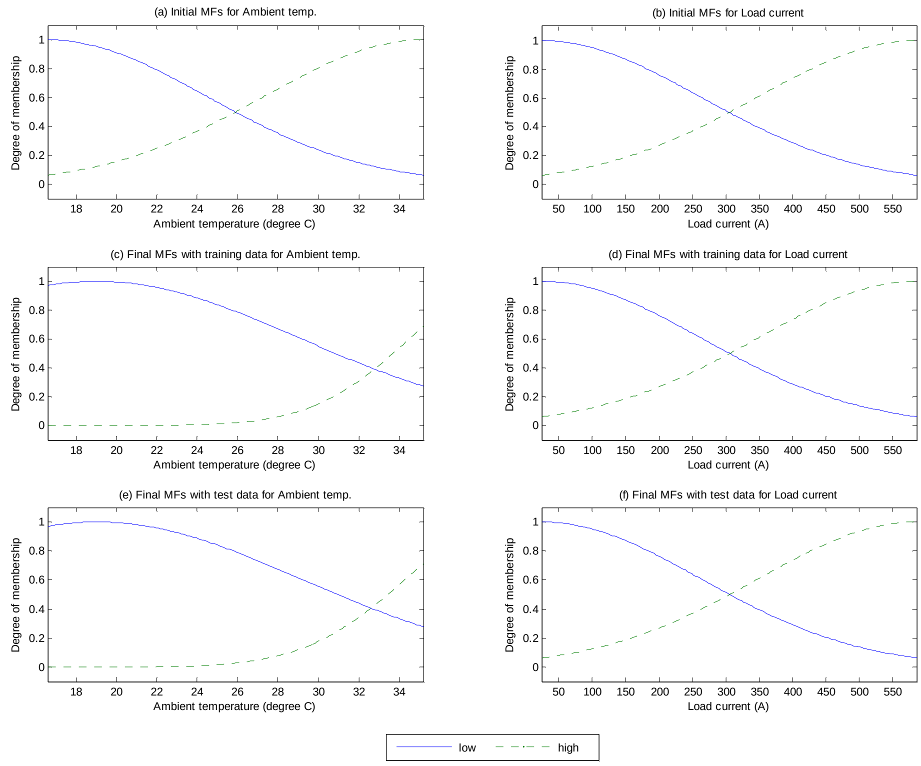

Figure 5.20: The changes in the Gaussian MFs of the training data and test data before and after training.

5.2.1.9 Summary

This Chapter initially illustrated the range of data generated by the standard model and then produced selected data for variations of that model thereby demonstrating that the model chosen (2 Gaussian MFs, initial step size of 0.01, 70 epochs, hybrid learning type, linear output) was the one best suited to predict the top-oil temperature. A summary of particular conclusions reached in this section is listed below:

- The Gaussian MF type is the best choice of MF for top-oil temperature prediction as shown in Table 5.1.

- Increasing the number of training epochs, improves the RMSE performance. Even though error is reduced further after 70 epochs, the improvement in the prediction of the top-oil temperature is small. On the other hand, increasing the number of epochs consumes more time. Therefore, it was considered sufficient to train within 70 epochs.

- The number of MFs for each input that gave the best match for the top-oil temperature data was 2 as shown in Table 5.3.

- A comparison of the result of varying the number of MFs for each input reflects the complexity of ANFIS for choosing parameters. When the number of MFs for each input is 3, 4, 5 or 6, it produces redundancy for the structure of data, therefore the RMSE increases.

It is important for the structure of the system used to match the available data. Therefore, to build a model with ANFIS, the choice of data must express all the properties of the system. The choice for the system structure should have enough parameters to reflect all the characteristics. However, the number of parameters should also be restrained. It is by no means true that the more complex the structure, the better the effect. The structure should match the data. A decision needs to be made on the structure based on experience or by changing it to observe the effect in a special application. For the optimization method, the effect of choosing a combination of least-squares and back-propagation gradient descent methods is much better than that of the back-propagation method alone. Choosing the linear output MF type is much better than choosing the constant output MF type. It can be observed from Figure 5.19(a) that the changes in RMSE between the test data and the training data all decrease and match well after training. This establishes that no overfitting has occurred in the ANFIS training process. A number of large peaks error values occurred in the predicted top-oil temperature values because the load current values have dropped unexpectedly.

The ANFIS prediction of top-oil temperature of the 8 MVA transformer has produced the following results: RMSE = 3.5299, minimum peak error = -23.2437 and maximum peak error = 21.4646 under the following conditions: number of MFs = 2 Gaussian MFs (*low* and *high*), initial step size = 0.04, step size decrease rate = 0.9, step size increase rate =

1.1, epoch number = 100 epochs, learning type = hybrid and linear type output. However, the peaks error values are high when the top-oil temperature patterns vary rapidly as in the case on days 13, 21, 28, 35, 43 and 71 due to the sudden decrease of load (Figure A.1 in appendix A). In order to overcome this problem, one can increase the number of input variables (Jang, 1993).

In the next section an extra input variable (bottom-oil temperature)[1] (refer to Figure A.2 in appendix A) has been added to the input vector (input data set) to investigate whether an improved RMSE can be achieved.

5.2.2 The ANFIS Top-oil Temperature Prediction with Three Input Variables (8 MVA Transformer)

The ANFIS model now has three inputs, θ_{amb} , θ_{bot} and I_{load}. The input and output data sets have been divided into two halves similar to the ANFIS with two input variables case. Each input has two MFs designated *low* and *high*. As a result of the findings for the two input case the following parameters were set: initial step size 0.04, step size decrease rate = 0.9, step size increase rate = 1.1, epoch number = 100 epochs, the learning type is hybrid with linear type output.

The measured and predicted top-oil temperature of the 8 MVA transformer is shown in Figure 5.21 (see Appendix B for the Matlab source codes).

The comparison of peak error, RMSEs (root mean square error) and the training step sizes are illustrated in Figure 5.22 and Figure 5.23(a) and (b). From Figure 5.22, one can clearly see that the RMSE decreases significantly from 3.5299 (two input variables) to 0.8915 (three input variables). The changes in the Gaussian MFs of the training and test data before and after training are shown in Figures 5.24(a) to 5.24(i).

[1] Using the bottom-oil temperature to predict the top-oil temperature is not practical for a real system as it is just as simple to measure the top-oil temperature as it is to measure the bottom-oil temperature. However the success of the bottom-oil temperature in predicting the top-oil temperature demonstrates the value of the technique where a strong link exists between the predictor and the parameter being predicted.

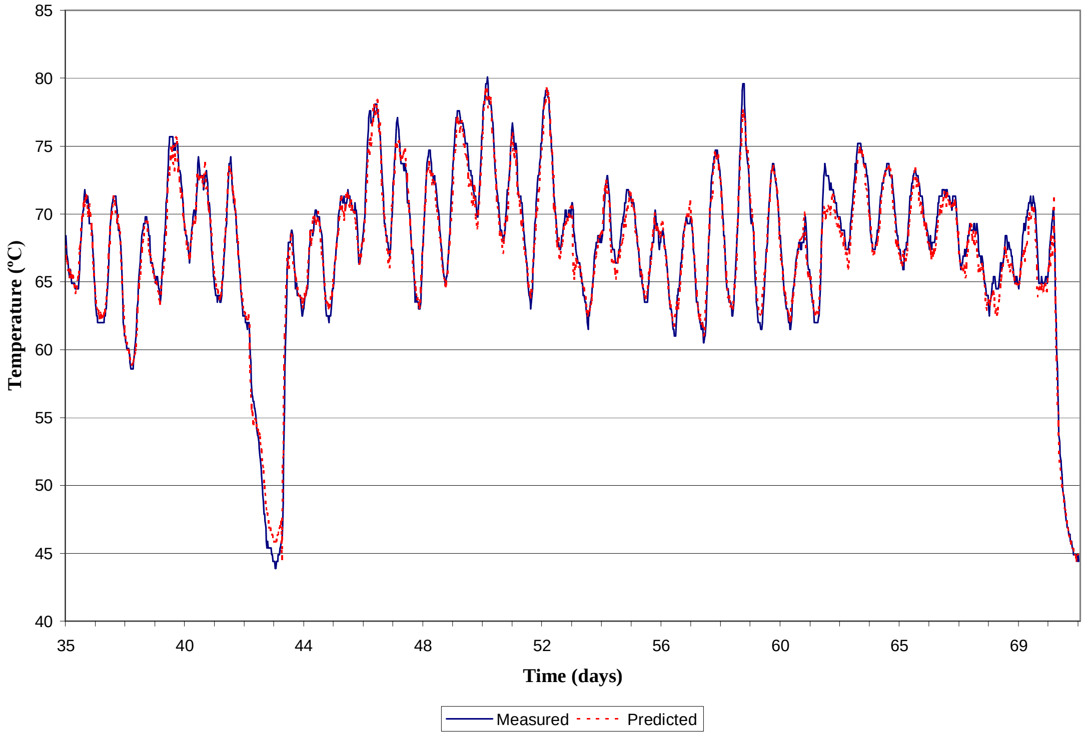

Figure 5.21: The measured and predicted of the top-oil temperature (8 MVA transformer) by using the ANFIS model with 3 inputs.

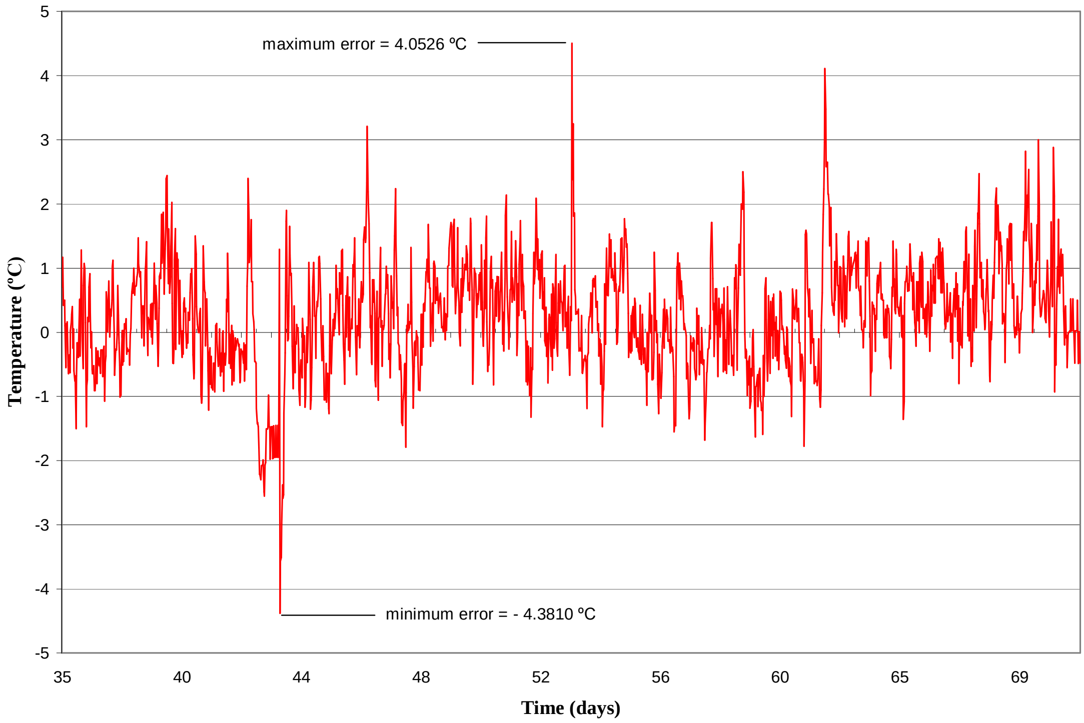

Figure 5.22: The difference between the measured and predicted top-oil temperature (8 MVA) using the ANFIS model with 3 inputs.

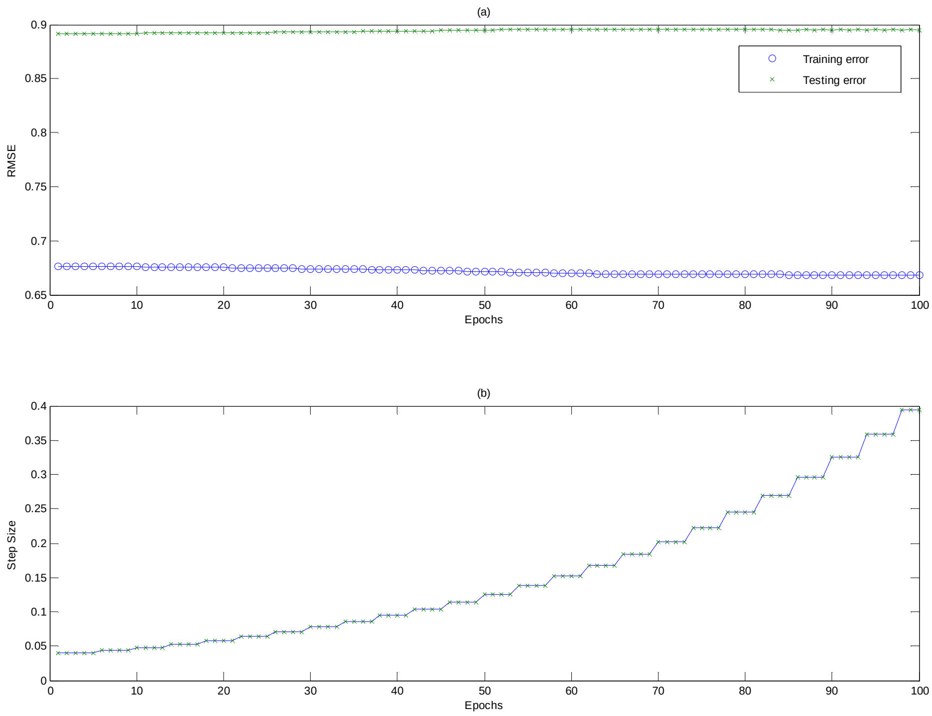

Figure 5.23: (a) RMSE curves; (b) Training step size of Gaussian MFs.

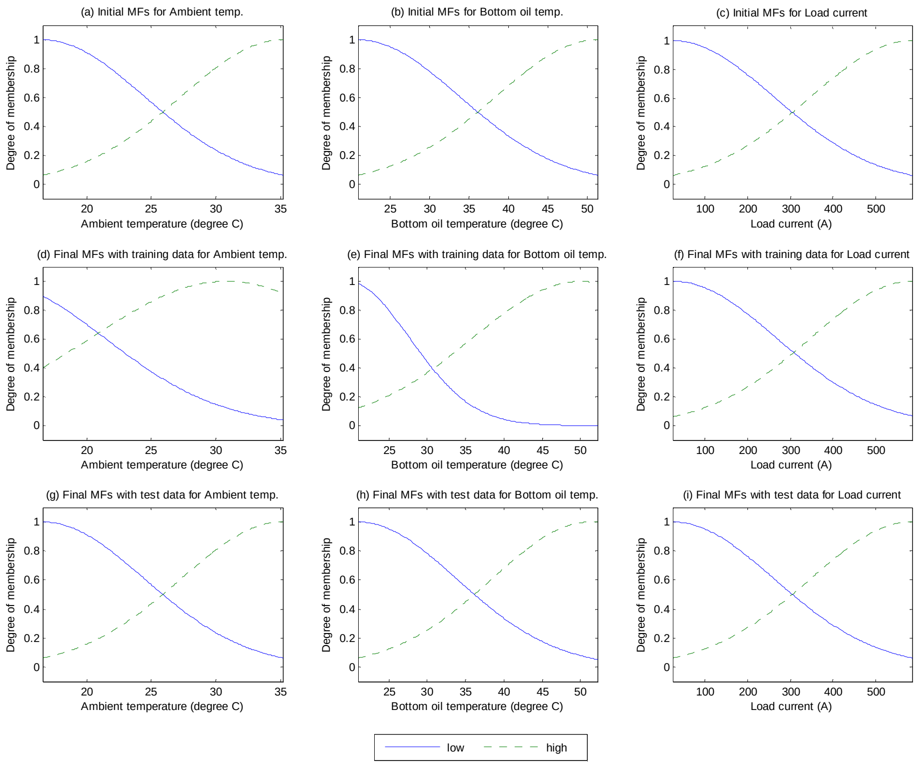

Figure 5.24: The changes in the Gaussian MFs of the training data and test data before and after training.

From Figure 5.21, the 3 input ANFIS model demonstrates that by using more input variables, the performance is improved dramatically. Although, a small overfitting occurred between the training and testing RMSE curves this can be considered insignificant because the results shown in Table 5.6 confirmed that in this case 3 inputs is better than 2. This positive result does not indicate that including more input variables necessarily leads to an improved performance. Inputs should be carefully selected by using experience from the expert, or by trial and error. For the 8 MVA transformer, the bottom oil temperature has a similar pattern to that of the top-oil temperature and therefore, was a logical choice for the input selection process for the ANFIS.

The final results for both ANFIS with 2 and 3 input variables are summarized in Table 5.6.

Number of inputs	RMSE	Min	Max
2	3.5299	-23.2437	21.4646
3	0.8915	-4.381	4.5026

Table 5.6: The RMSE values for the top-oil temperature (8 MVA transformer) using the ANFIS model with the different number of input variables.

5.2.3 The ANFIS Bottom-oil Temperature Prediction (8 MVA Transformer)

A further test of the capability of ANFIS model used in predicting the top-oil temperature was achieved by application to the bottom oil temperature data. The training and test data sets remain the same as in the top-oil temperature prediction for the ANFIS model with 2 variables. An extra variable (top-oil temperature) was added to the ANFIS model with 3 input variables.

Two Input Variables

Following the same methodology as in Section 5.1.1, the prediction of bottom oil temperature of the 8 MVA transformer produced the following results: RMSE = 2.9099, minimum peak error = -15.6571 and maximum peak error = 14.8042. These results were obtained under the following conditions: number of MFs = 2 Gaussian MFs (*low* and *high*), initial step size = 0.04, step size decrease rate = 0.9, step increase rate = 1.1, epoch number = 50 epochs, learning type = hybrid and linear type output. Similar to the top-oil temperature prediction case, the ANFIS prediction of bottom oil temperature does show poor performance where the load current has a sudden change in its value. Figure 5.25(a) shows that no overfitting has occurred during the ANFIS training process.

Each of the RMSE, minimum and maximum peak error values show improvement over those for the top temperature predictions as discussed in Section 5.1.1.1. This is not a surprising result since the temperature at the bottom of the transformer is less extreme than at the top, which is a direct consequence of heat rising. The bottom temperature therefore is both less extreme than the top and takes a longer period of time to vary: hence the model is better able to cope with temperature changes when they occur.

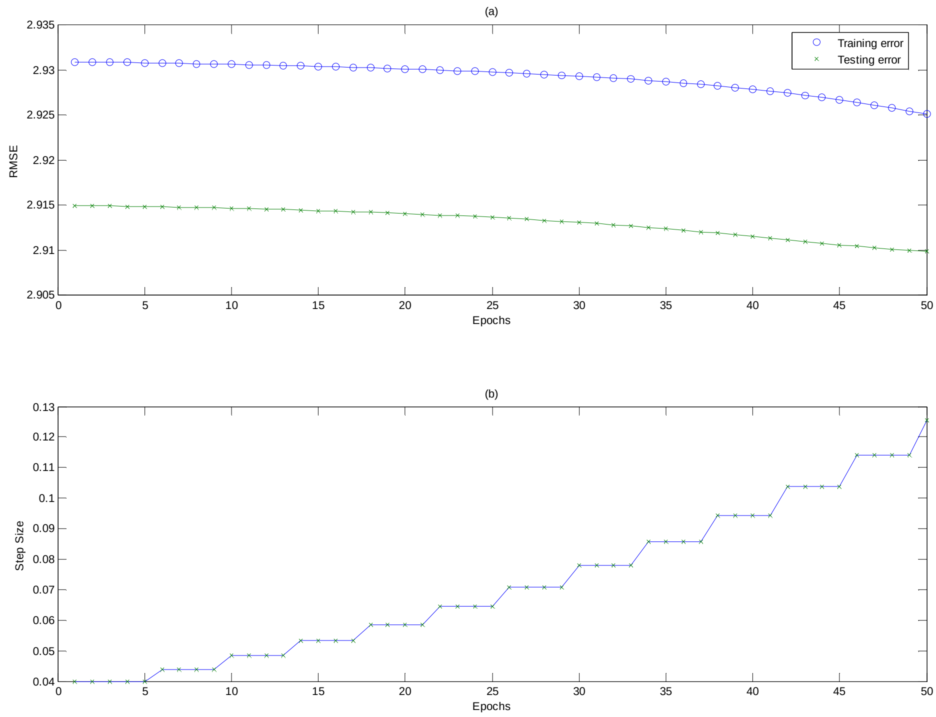

Figure 5.25: (a) RMSE curves; (b) Training step sizes of the Gaussian MFs.

<u>Three Input Variables</u>

Follow the same manner as in Section 5.1.2, an extra input variable (top-oil temperature)[2] (Figure A.1 in appendix A) has been added to the ANFIS model.

Now, with the same conditions as in the ANFIS model for bottom oil temperature (8 MVA transformer) prediction with 2 input variables. There was a small overfitting as indicated on graph in Figure 5.26(a) which once again can be considered insignificant by comparing the overall result in Table 5.7. The measured and predicted bottom oil temperature waveforms are shown in Figure 5.27, whereas the difference between these two waveforms is shown in Figure 5.38.

[2] Using the top-oil temperature to predict the bottom-oil temperature is not practical for a real system as it is just as simple to measure the bottom-oil temperature as it is to measure the top-oil temperature. However the success of the top-oil temperature in predicting the bottom-oil temperature demonstrates the value of the technique where a strong link exists between the predictor and the parameter being predicted.

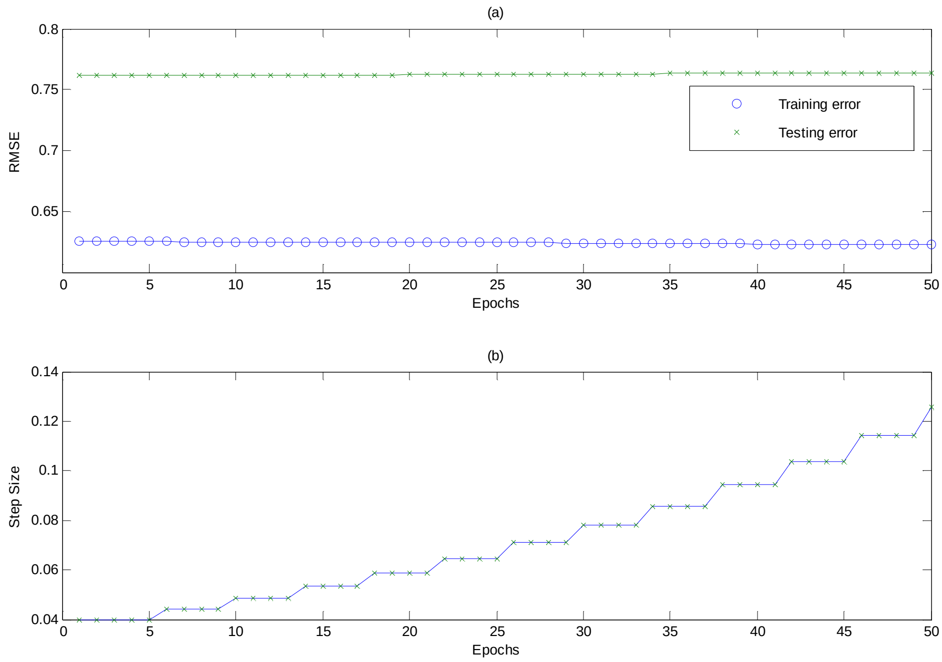

Figure 5.26: (a) The changes in RMSE of the training and test data; (b) Training step sizes.

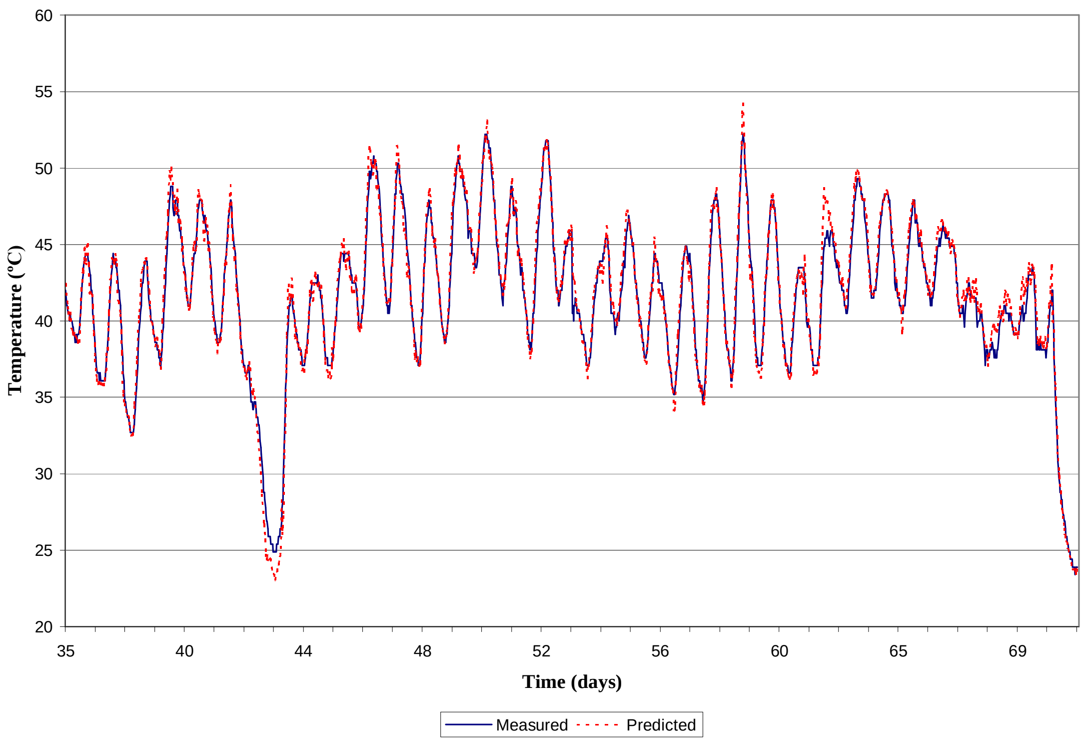

Figure 5.27: The measured and predicted bottom oil temperature (8 MVA transformer) by using the ANFIS model with 3 input variables.

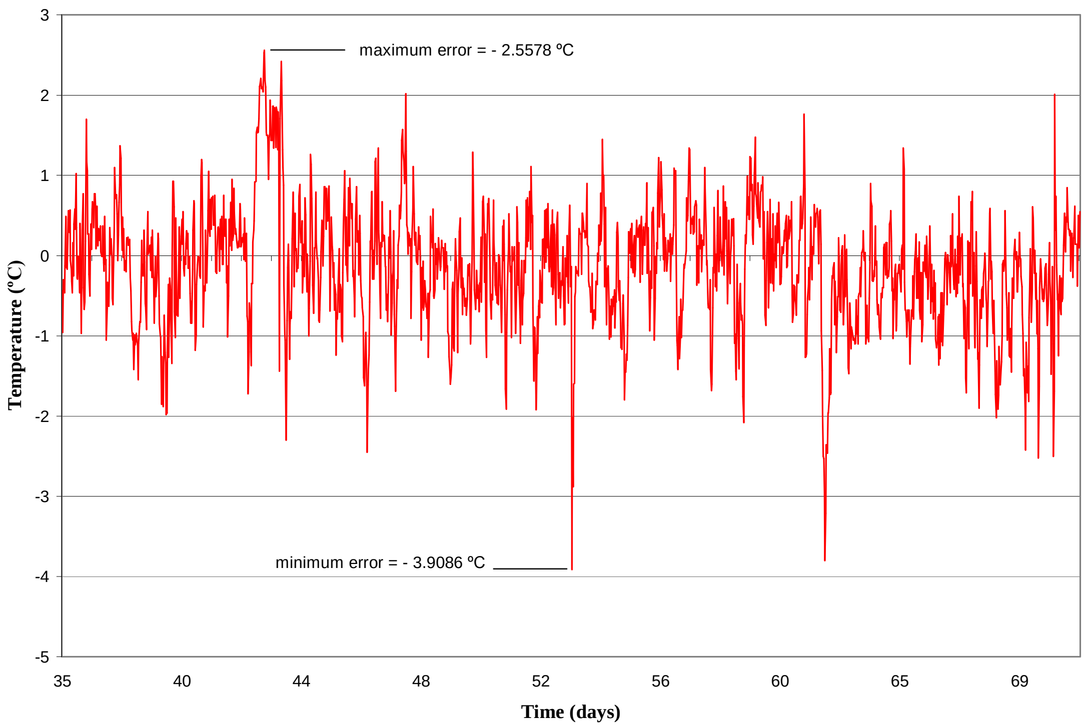

Figure 5.28: The difference between the measured and predicted bottom oil temperature (8 MVA transformer).

Number of inputs	RMSE	Min	Max
2	2.9099	-15.6571	14.8042
3	0.7617	-3.9086	2.5578

Table 5.7: The RMSE values for the bottom oil temperature (8 MVA transformer) prediction by using the ANFIS model with three input variables.

5.2.4 The ANFIS Top-oil Temperature Prediction (27 MVA Transformer)

In this Section, the data of the 27 MVA transformer were used to further examine the capability of the ANFIS model.

<u>Two Input Variables</u>

Following the same methodology as in Section 5.1.1, the ANFIS prediction of top-oil temperature for the 27 MVA transformer has produced the following results: RMSE = 2.9608, minimum peak error = -6.6134 and maximum peak error = 2.7499. These results were obtained when simulating ANFIS under the following conditions: number of MFs = 2, triangular MFs (*low* and *high*), initial step size = 0.04, step size decrease rate = 0.9, step increase rate = 1.1, epoch number = 20 epochs, learning type = hybrid and linear type output. It should be noted that some parameters have changed from those used for the 8 MVA transformer data set. These are: the triangular MF and an epoch number of 20. These were selected for the data set of the 27 MVA transformer from a trial and error process as was the case for the 8 MVA transformer.

The ANFIS prediction with 2 input variables has suffered a similar problem to that of the ANFIS with 2 input variable of the 8 MVA transformer, that is the peak error values are high in comparison to the maximum measured value of the top-oil temperature is $28.4^{\circ}C$.

Three Input Variables

Following the same methodology as in Section 5.1.2, an extra input variable (bottom oil temperature) (see Figure A.6) has been added to the input data set of the ANFIS model.

The same conditions as in the ANFIS prediction of 2 input variables case were used here. The measured and predicted waveforms of top-oil temperature of the 27 MVA transformer are shown in Figure 5.29 and the error wave form can be seen in Figure 5.30. The changes in the Gaussian MFs of the training and test data before and after training are shown in Figures 5.31(a) to (i). For comparison purposes, the final results for both ANFIS with 2 and 3 input variables are summarized in Table 5.12.

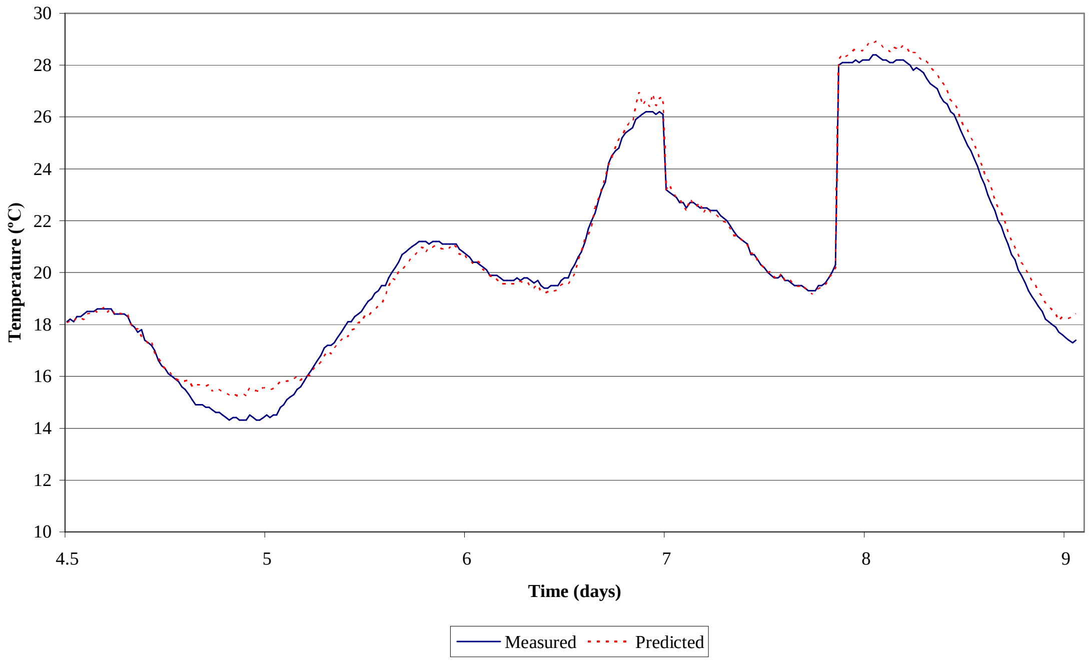

Figure 5.29: The measured and predicted waveforms of top-oil temperature (27 MVA transformer) with ANFIS (3 inputs) of triangular MFs.

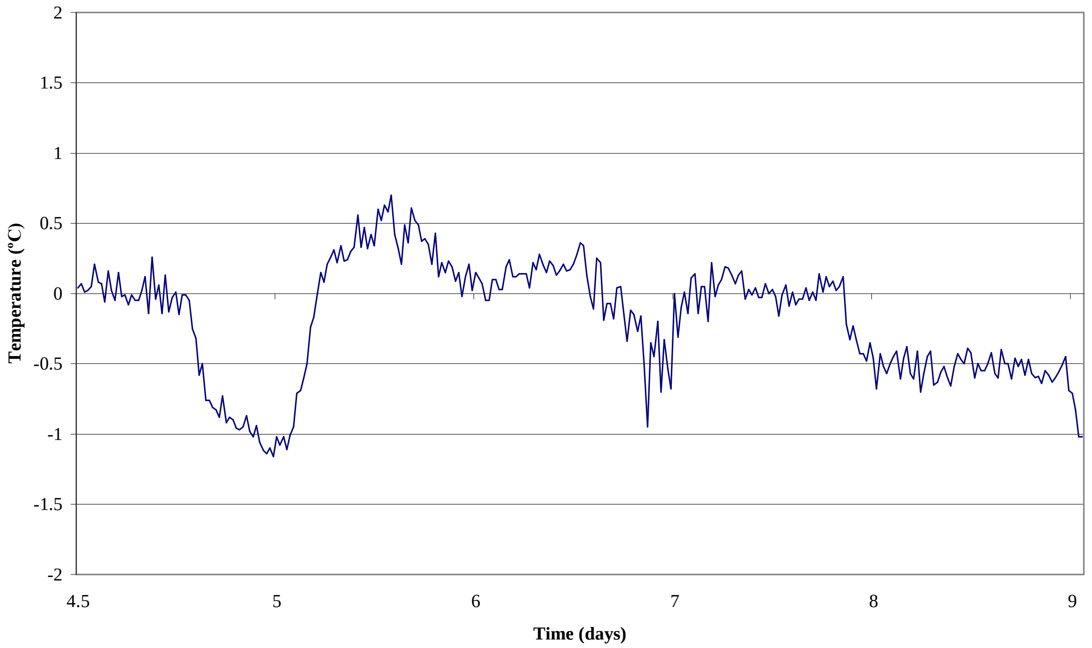

Figure 5.30: The error waveform between the measured and predicted top-oil temperature (27 MVA transformer) by using the ANFIS model with 3 input variables.

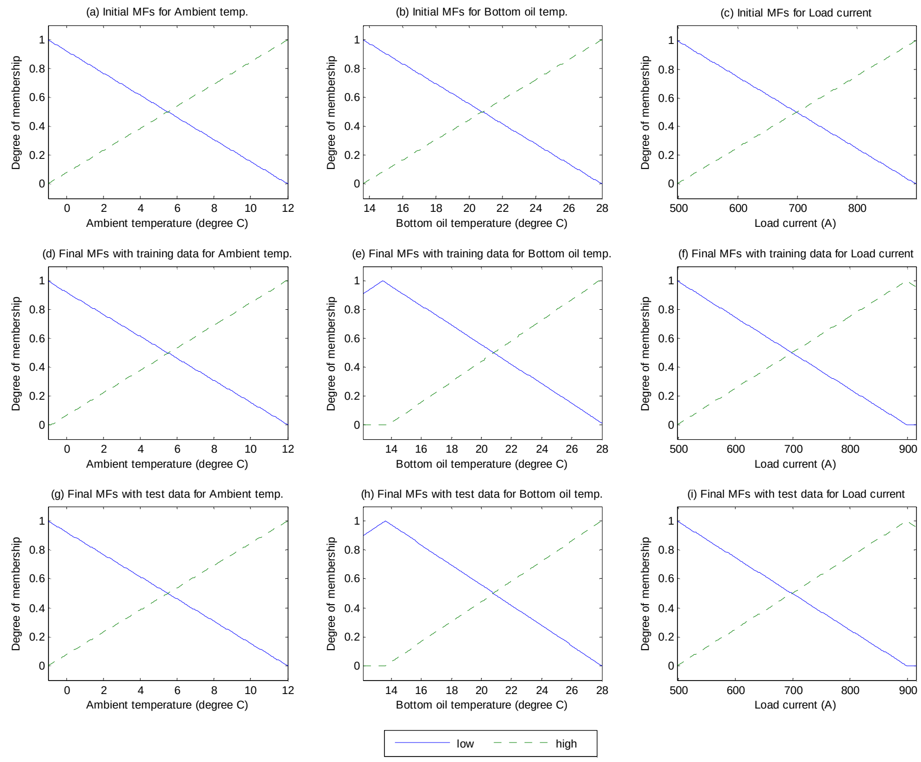

Figure 5.31: The changes in triangular MFs of training and test data before and after training.

Number of inputs	RMSE	Min	Max
2	2.9608	-6.6134	2.7499
3	0.4492	-1.1554	0.6953

Table 5.8: RMSE and peak error values of the measured and predicted top-oil temperature with ANFIS of 2 and 3 input variables.

5.2.5 The ANFIS Bottom-oil Temperature (27 MVA Transformer)

<u>Two Input Variables</u>

Following the same methodology as in Section 5.1.2 the ANFIS prediction of bottom oil temperature of transformer 27 MVA has produced these results: RMSE = 2.9222; minimum peak error = -6.6692 and maximum peak error = 2.5354. This was carried out under the following conditions: number of MFs = 2 triangular MFs (*low* and *high*); initial step size = 0.04; step size decrease rate = 0.9; step increase rate = 1.1; epoch number = 20 epochs; learning type = hybrid and linear type output. However, similar to the top-oil temperature (27 MVA transformer) the ANFIS prediction with 2 input variables has a poor performance in terms of peak errors.

Three Input Variables

An extra input variable (top-oil temperature) (see Figure 5.52) has been added to the input data set. The input and output data sets have been divided into two halves. One half of the data (from day 1 to day 4.5) is used for training, the other half (from day 4.5 to day 9) is used for testing.

The same conditions (number of MFs = 2 triangular MFs (*low* and *high*); initial step size = 0.04; step size decrease rate = 0.9; step increase rate = 1.1; epoch number = 20 epochs; learning type = hybrid and linear type output) as in the ANFIS prediction of 2 input variables case.

The result of predicting top-oil temperature of the 27 MVA with ANFIS (3 input variables) is shown Figure 5.32 and the error waveform between the measured and predicted can be seen in Figure 5.33.

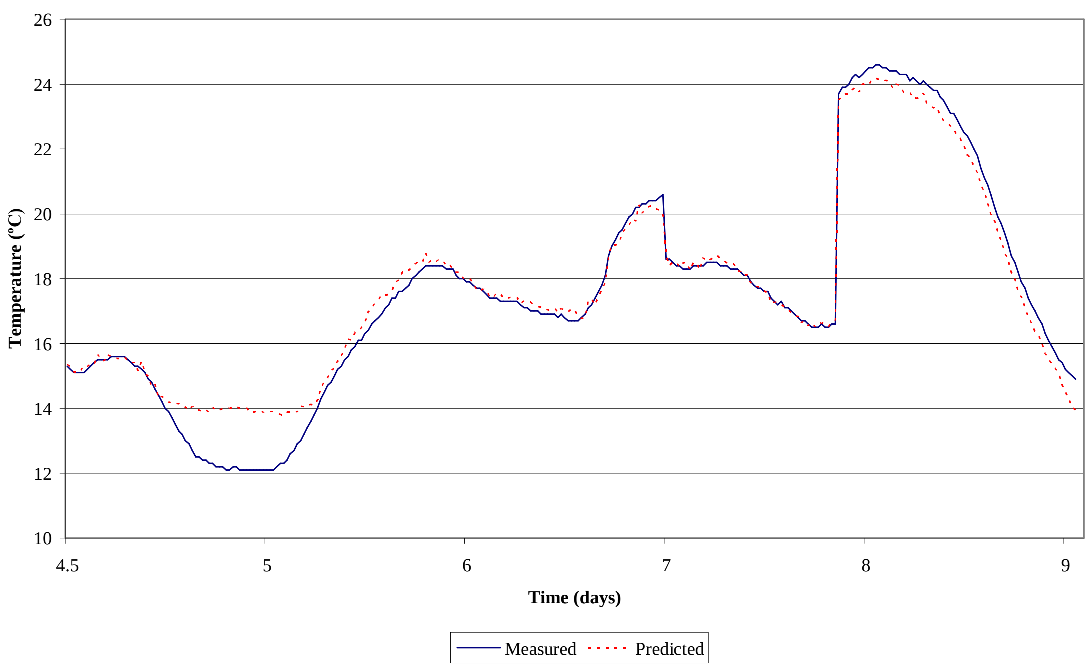

Figure 5.32: The measured and predicted of the bottom oil temperature using ANFIS with 3 input variables.

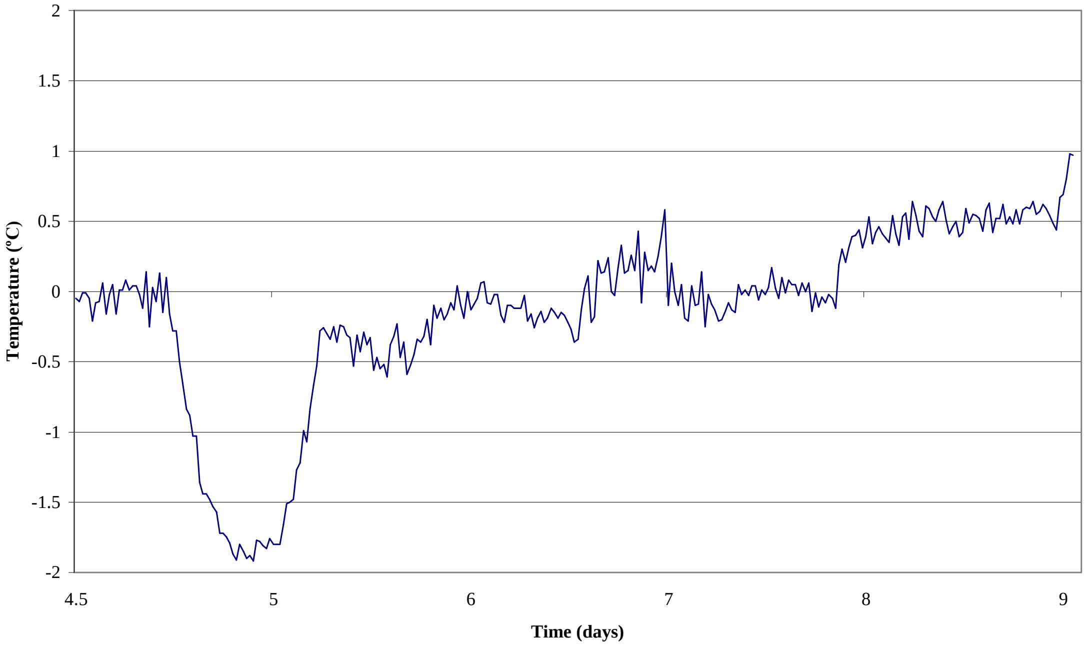

Figure 5.33: The difference between the measured and predicted bottom oil temperature (27 MVA transformer) using ANFIS with 3 inputs.

For comparison purposes, the results of using ANFIS with 2 and 3 input variables are summarized in Table 5.9.

Number of inputs	RMSE	Min	Max
2	2.9222	-6.6692	2.5354
3	0.6449	-1.9242	0.9824

Table 5.9: RMSE and peak error values of bottom oil temperature prediction using ANFIS with 2 and 3 inputs.

5.2.6 Summary and Analyses

From the above investigation and comparisons, the following summary and analyses can be generated:

- Eight different MFs were used to predict the transformer's temperatures. They were bell MFs, triangular MFs, Gaussian MFs, trapezoidal MFs, two-sided Gaussian MFs, product of two sigmoidal MFs, different between two sigmoidal

MFs, and pi-shaped MFs. In predicting top and bottom oil temperatures, the differences in these MFs are very small, as reflected in the RMSE values. In predicting the top and bottom oil temperatures of the 8 MVA transformer, the Gaussian MFs produce the smallest error. However, with the 27 MVA transformer, the triangular MFs is the best for predicting top and bottom oil temperatures. The Gaussian MFs have a continuous differential function whereas the triangular MFs do not. This indicates that the temperature patterns of the 27 MVA transformer are simpler than the 8 MVA transformer.

- When ANFIS is used to predict the top and bottom oil temperatures of the 8 MVA and 27 MVA transformers, the effect of choosing the combination of least-squares and back-propagation gradient descent method is much better than that of the back-propagation method alone.

- Through increasing the number of the training epochs, the temperature prediction improves but consumes more time. For the top-oil temperature (8 MVA transformer) prediction, after 100 epochs, the RMSE curve tended to stabilize. Even though the error is reduced after 100 epochs, the difference is insignificant. For bottom oil temperature prediction (8 MVA transformer) prediction, it is sufficient to train within 50 epochs. For the top and bottom oil temperatures (27 MVA transformer), 20 epochs have shown to be sufficient.

- For both transformers, the *linear* output MF produces much better results than the *constant* output MF.

- For both transformers, the hybrid learning algorithm is shown to be superior in predicting top and bottom oil temperatures.

- The step size of training data for all MFs in ANFIS is related to the RMSE. If the RMSE decreases, the step size increases. However, if the RMSE oscillates, the step size decreases. When the initial step size number is increased, the RMSE is likely to oscillate.

- Choosing the number of MFs for each input reflects the complexity of ANFIS for choosing parameters. For both transformers, the number of MFs for each input is 2 and that is enough to reflect the structure of the top and bottom oil temperature data. However when the number of MFs for each input is increased, it produces redundancy for the structure of top and bottom temperature data. It is important that the structure of the system matches the data. Therefore, to build the model with ANFIS, the choice of data must express all the properties of the system. It is by no means true that the more complex the structure, the better the prediction.

- The ANFIS model was tested with 2 and 3 input variables. When using 3 inputs, the prediction capability of ANFIS is extremely good. This result may simply reflect the fact that the third input (bottom oil temperature to predict the top-oil temperature or the top-oil temperature to predict the bottom oil temperature) was more directly correlated with the parameter being predicted. Table 5.10 provided the summary of ANFIS used in predicting the top and bottom oil temperatures.

TRANSFORMER	Type of temperatures	Number of inputs	RMSE	Min. peak error	Max. peak error
8 MVA transformer	top-oil	2	3.5299	-23.2437	21.4646
		3	0.8915	-4.3810	4.5026
	bottom oil	2	2.9099	-15.6571	14.8042
		3	0.7617	-3.9086	2.5578
27 MVA transformer	top-oil	2	2.9608	-6.6134	2.7499
		3	0.4492	-1.1554	0.6953
	bottom oil	2	2.9222	-6.6692	2.5354
		3	0.6449	-1.9242	0.9824

Table 5.10: ANFIS performances in predicting the top and bottom oil temperatures with different number of inputs.

5.3 Transformer Temperature Prediction Using Neural Network Models

This section presents detailed simulation and results for the prediction of top-oil temperature using two specific neural network models: Multilayer Feedforward Neural Network (MFNN) and Elman Recurrent Neural Network (ERNN).

The determination of neural network architecture is a designer-defined process. In most cases it follows a heuristic approach where several networks have various parameters [type of activation functions, number of hidden layers, and number of nodes (neurons)] in the hidden layer that are trained: the best performing network among them is then selected.

This section also provides some guidelines as to how to construct neural network models for the prediction of top-oil temperature using Matlab.

Network Structures

The commonly used MFNN and ERNN are shown in Figure 5.34 and 5.34 respectively. It has been established that a network of these types can approximate any continuous function as closely as desired (Haykin, 1994). With appropriate arrangement of the input variables one can obtain different system models from these neural networks between the input(s) and the output(s). Though neural networks can have any number of hidden layers, typically one hidden layer is used. Otherwise, the number of possible interactions between nodes may make the model unnecessarily complex and difficult to train.

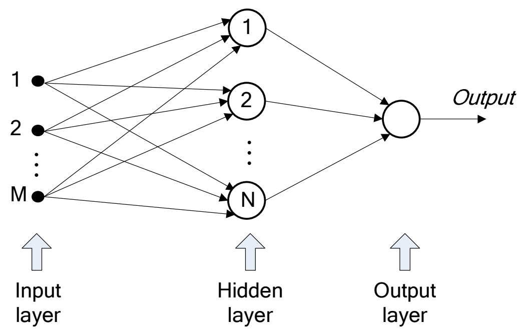

Figure 5.34: A MFNN for case of M inputs, 1 hidden layer (N hidden nodes) and 1 output layer.

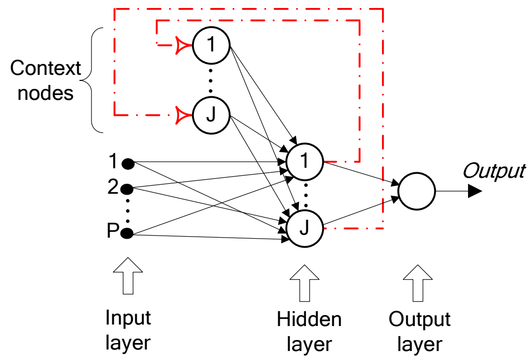

Figure 5.35: An ERNN for case of P inputs, 1 hidden layer (J hidden nodes) and 1 output layer.

Training of Neural Networks

In this thesis, the Levenberg-Marquardt algorithm is used for the training of Multilayer feedforward and Elman neural networks in the prediction of top-oil temperature for the simple reason that it is faster than the other techniques in Matlab (The MathWorks, 1999).

Determining the number of hidden nodes

A combined consideration of training error and testing error has demonstrated to be an appropriate criterion for determining the number of hidden nodes in neural networks (Cortes, 1995). Let P be the total number of training samples and n_h be the number of hidden nodes. In the top-oil temperature prediction case, P is fixed and n_h is determined by the following principle. Consider the case of n_h being much less than P, both training and testing errors are large because the number of parameters available to model the irregularities of the training data is small. Increasing n_{h} allows the model to more closely represent the training data (as measured by smaller training error). The training error will continue to decline as n_{h} is increased and eventually, may even reach zero. However, the testing error will initially decrease as n_{h} increases but at some stage, $n_{\mathrm{h}}^{''}$, the testing error will start increasing. The rise of testing error is due to overfitting. Therefore, $n_{\mathrm{h}}^{''}$ should be the appropriate number of hidden nodes, which normally will result in both smaller training and testing errors. In this thesis for the neural networks application of top-oil temperature prediction, the process described above for determining the number hidden nodes was used.

In the case of determining the optimal number of hidden nodes, for each n_{h}, ten different training processes were executed in order to eliminate the random effect of arbitrary initialization of neural network weights. Therefore, the optimal number of hidden nodes was chosen from averaged minimum training and testing errors. The number of hidden nodes was varied from 2 to 20 in order to choose the right number of hidden nodes.

5.3.1 The MFNN Top-oil Temperature Prediction with Two Input Variables (8 MVA Transformer)

This network has 2 inputs (ambient temperature and load current), 1 hidden layer, and 1 output layer (top-oil temperature). The activation function used in each node of a layer can be any standard function (The MathWorks, 1999) that provides a meaningful relationship between the input and output. Several feedforwad neural networks with different activation functions (i.e. linear, symmetric saturation linear, logarithmic sigmoid, hyperbolic tangent sigmoid, symmetric hard limit, etc) and number of nodes within the hidden layer were tried in order to find the architecture that modeled the data most effectively. It was found that the logarithmic sigmoid function (*logsig*) is suitable for each hidden node, whereas the linear transfer function (*purelin*) is suitable for the output node. The network was then trained by the implementation of backpropagation algorithm with Levenberg-Marquardt technique in Matlab. The study commenced by using the smallest number of hidden nodes. Table 5.11 Shows the RMSE and peak error values of the measured and predicted top-oil temperature of the 8 MVA transformer by using a two layer feedforward neural network with different number of hidden nodes in the in hidden layer.

Number of hidden nodes	RMSE	Min	Max
2	3.6834	-23.1107	21.9852
4	3.6984	-22.1694	21.5040
6	3.6508	-23.9462	21.9656
8	3.5667	-23.0794	21.9265
10	3.6459	-23.3562	21.4455
12	3.6611	-27.8921	21.3158
14	3.6183	-24.3391	21.4898
16	3.6946	-27.4761	21.5601
18	3.6588	-22.7552	21.3557
20	3.6339	-23.3131	21.5840

Table 5.11: The RMSE and peaks error values for a 2-layer feedforward neural network with different number of hidden nodes of top-oil temperature prediction (8 MVA transformer).

The multilayer feedforward neural network prediction of top-oil temperature of transformer 8 MVA has produced the following results of RMSE = 3.5667 with minimum peak error = -23.0794 and maximum peak error = 21.9265 under the following condition: number of hidden nodes = 8 (*logsig*); learning rate = 0.01; epoch number = 100 epochs; output node = 1 (*purelin*).

In the next section an extra input variable (bottom oil temperature) (see Figure A.2 in appendix A) has been added to the input vector (input data set) to investigate whether an improved RMSE can be achieved.

5.3.2 The MFNN Top-oil Temperature Prediction with Three Input Variables (8 MVA Transformer)

The MFNN model now has three inputs, θ_{amb} , θ_{bot} and I_{load}. The input and output data sets have been divided into two halves similar to the MFNN with two input variables case. As a result of the findings for the two input case the following parameters were set: number of hidden nodes = 8 (*logsig*); learning rate = 0.01; epoch number = 100 epochs; output node = 1 (*purelin*). Figure 5.36 shows the measured and predicted top-oil temperature (see Appendix C for the Matlab source codes). The error waveform is shown in Figure 5.37. Table 5.12 shows the comparison between the MFNN with 2 and 3 input variables.

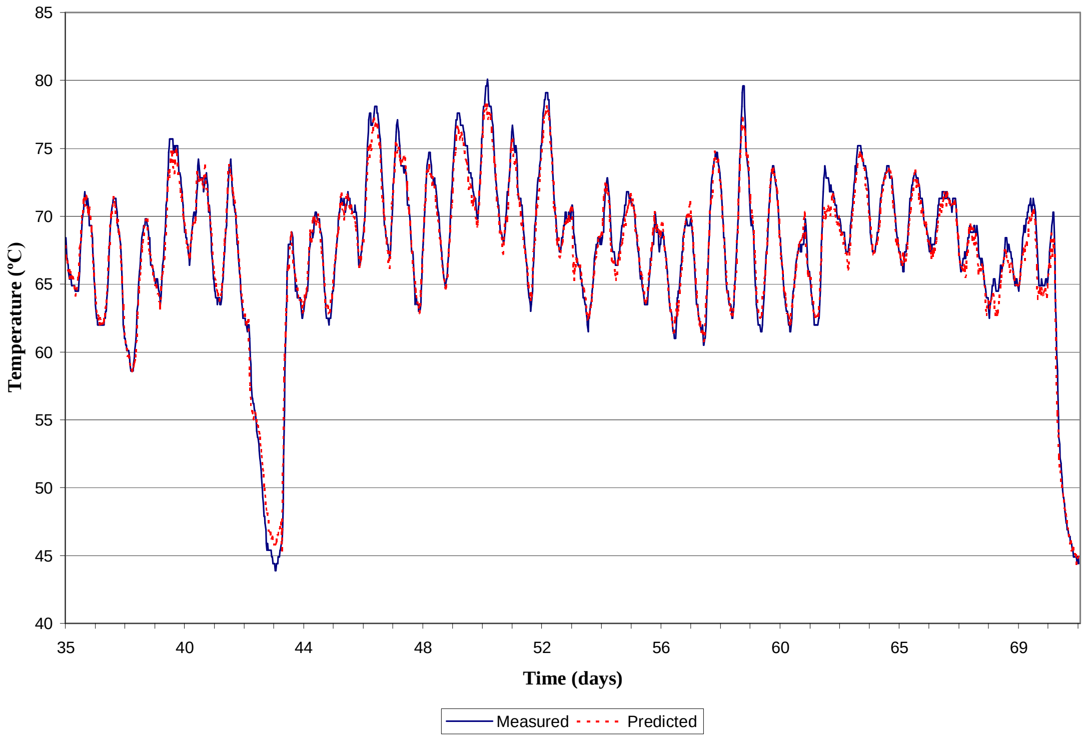

Figure 5.36: The measured and predicted top-oil temperature (8 MVA transformer) by using MFNN with 3 inputs.

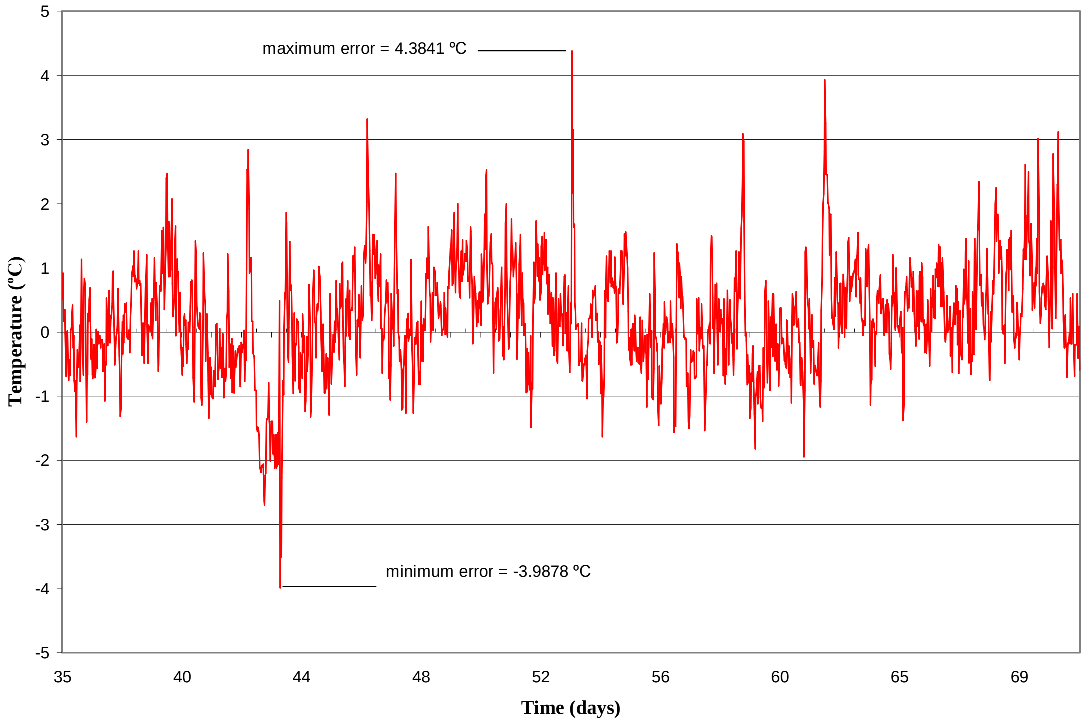

Figure 5.37: The error waveform of the predicted top-oil temperature using MFNN with 3 inputs.

Number of inputs	RMSE	Min	Max
2	3.5667	-23.0794	21.9265
3	0.8783	-3.9878	4.3841

Table 5.12: A comparison of the predicted top-oil temperature (8 MVA transformer) by using MFNN with 2 and 3 input variables.

5.3.3 The MFNN Bottom-oil Temperature Prediction (8 MVA Transformer)

An additional test of the capability of the MFNN model used in predicting the top-oil temperature was achieved by application to the bottom oil temperature data. The training and test data sets remain the same as in the top-oil temperature prediction for the MFNN model case with 2 variables. An extra variable (top-oil temperature) was added for the MFNN model with 3 input variables case.

The following parameters were set for the MFNN model with 2 and 3 input variables: number of hidden nodes = 8 (*logsig*); learning rate = 0.01; epoch number = 100 epochs;

output node = 1 (*purelin*). Table 5.13 shows the comparison between the MFNN with 2 and 3 inputs.

Number of inputs	RMSE	Min	Max
2	2.9361	-15.4849	15.4849
3	0.7909	-3.8191	4.1215

Table 5.13: A comparison of the predicted bottom-oil temperature (8 MVA transformer) by using the MFNN with 2 and 3 input variables.

5.3.4 The MFNN Top-oil Temperature Prediction (27 MVA Transformer)

The data of the 27 MVA transformer were used to further examine the ability to forecast of the MFNN model in this Section.

Following the same methodology as in Section 5.3.1, the optimum values of RMSE and peaks of error were achieved under the following conditions: number of hidden nodes = 4 (*logsig*), learning rate = 0.01, epoch number = 100 epochs and output node = 1 (*purelin*). The results of top and bottom-oil temperatures prediction for the 27 MVA transformer with 2 and 3 input variables are shown in Table 5.14 and 5.15 respectively. The source code for the three input variables case can be referred to Appendix C.

Number of inputs	RMSE	Min	Max
2	3.3310	-7.6340	1.8398
3	0.2631	-1.1171	0.3642

Table 5.14: A comparison of the predicted top-oil temperature (27 MVA transformer) by using the MFNN with 2 and 3 input variables.

Number of inputs	RMSE	Min	Max
2	3.0615	-6.9128	2.1698
3	0.3154	-0.5323	0.9681

Table 5.15: A comparison of the predicted bottom-oil temperature (27 MVA transformer) by using the MFNN with 2 and 3 input variables.

5.3.5 The ERNN Temperatures Prediction (8 MVA Transformer)

Following the same procedure as in Section 5.3.1, the ERNN model was used to predict the top and bottom-oil temperatures for the 8 MVA transformer. The optimum values of RMSE and peaks of error were obtained under the following conditions: number of hidden nodes = 8 (*logsig*); learning rate = 0.01; epoch number = 100 epochs; output node = 1 (*purelin*). Table 5.16 shows the comparison between the ERNN with 2 and 3 input variables. Refer to Appendix D for the Matlab source codes.

Type of temperatures	Number of inputs	RMSE	MIN	MAX
Top-oil	2	3.6439	-23.983	21.9549
	3	0.9467	-3.2013	4.5050
Bottom-oil	2	2.9579	-15.532	14.7587
	3	0.8199	-3.5119	3.5131

Table 5.16: A comparison of the predicted temperatures for the 8 MVA transformer by using the ERNN model with 2 and 3 input variables.

5.3.6 The ERNN Temperatures Prediction (27 MVA Transformer)

Following the same procedures as in Section 5.3.1, the optimum values of RMSE and peaks of error were achieved under the following conditions: number of hidden nodes = 4 (*logsig*); learning rate = 0.01; epoch number = 100 epochs; output node = 1 (*purelin*). Table 5.17 shows the comparison between the ERNN with 2 and 3 input variables.

Type of temperatures	Number of inputs	RMSE	MIN	MAX
Top-oil	2	3.0149	-6.7041	1.1003
	3	0.3482	-1.0118	0.7040
Bottom-oil	2	2.9828	-6.3518	1.6420
	3	0.3043	-0.4985	1.0560

Table 5.17: A comparison of the predicted temperatures for the 27 MVA transformer by using the ERNN model with 2 and 3 input variables.

5.3.7 Summary

In this Section, two different neural network models (MFNN and ERNN) were applied to the prediction of transformer temperatures. The two models performed comparably in terms of RMSEs and peaks of error. The MFNN provided the best performance in predicting the top-oil temperature of the 27 MVA transformer with: RMSE = 0.2631, minimum peak error = -1.1171 (degree C) and maximum peak error = 0.3642 (degree C). In choosing the right network structure, different number of hidden nodes and type of activation functions for each node were trained and tested. The Levenberg-Marquardt algorithm was used in training of all the networks. This algorithm was indicated in (The MathWorks, 1999) to provide fast and stable training results when compared to the other available methods.

5.4 The Transformer Temperatures Prediction Using the Conventional Models

This section describes the calculation and analysis of the conventional approaches in predicting the top and bottom oil temperature of power transformers.

5.4.1 Top-oil Temperature Prediction Using Models 2 and 3 (27 MVA Transformer)

The calculation of the transformer's top-oil temperature based on Model number 2 is given by Equation (2.10). For FA cooling mode, n = 0.9 (IEEE, 1995), Equation (2.10) becomes,

$$\theta_{top}[k] = \frac{T_o}{T_o + \Delta t}\theta_{top}[k-1] + \frac{\Delta t \theta_{fl} R}{(T_o + \Delta t)(R+1)}\left(\frac{I[k]}{I_{rated}}\right)^{1.8} + \frac{\Delta t \theta_{fl}}{(T_o + \Delta t)(R+1)} \qquad (5.1)$$

where,

θ_{top} : top-oil temperature, ^oC

θ_{fl} : top-oil temperature at rated load, ^oC

θ_{amb} : ambient air temperature, ^oC

T_o : time constant at rated KVA

C : thermal capacity (Wh/ ^oC)

n : oil exponent

K : ratio of load to rated load

R : ratio of load loss to no-load loss at rated load.

k: step index.

Rated current is given as: $I_{rated} = 915A$

Sampling time is 15 min = 0.25 h

The top-oil temperature at rated load is:

$\theta_{fl} = 26.1°C$

Ratio of load loss at rated load to no-load loss is:

$$R = \frac{114519}{12576} = 9.1$$

For FA cooling mode, the thermal capacity is given by:

$C = 0.06 \times$ (weight of core and coil assembly in pounds)

+ 0.04 $\times$ (weight of tank and fittings in pounds)

+ 1.33 $\times$ (gallons of oil)

$$C = (0.06 \times 95700) + (0.04 \times 81600) + (1.33 \times 11484) = 24279.72 \text{ Wh/}^oC$$

Thermal time constant at rated KVA is given by Equation (2.3):

$$T_o = \frac{C\theta_{fl}}{P_{fl}} = \frac{24279.72 \times 26.1}{114519} = 5.53h$$

From the data given in table 1 below,

$\theta_{top}[k-1] = 27.2\ ^oC$ (sample number 1)

Then (5.1) becomes,

$$\theta_{top}[k] = \frac{5.53}{5.53+0.25} \times 27.2 + \frac{0.25 \times 26.1 \times 9.1}{(5.53+0.25)(9.1+1)}\left(\frac{779}{915}\right)^{1.6} + \frac{0.25 \times 26.1}{(5.53+0.25)(9.1+1)} = 26.92°C$$

This value can be checked with the measured value (sample number 2) given in Table 5.18.

Similarly, the transformer's top-oil temperature can be calculated using IEEE Model number 3, Equation (2.13) becomes:

$$\theta_{top}[k] = \frac{T_o}{T_o+\Delta t}\theta_{top}[k-1] + \frac{\Delta t}{T_o+\Delta t}\theta_{amb}[k] + \frac{\Delta t\theta_{fl} R}{(T_o+\Delta t)(R+1)}\left(\frac{I[k]}{I_{rated}}\right)^{1.6} + \frac{\Delta t\theta_{fl}}{(T_o+\Delta t)(R+1)} \qquad (5.2)$$

Please note that, IEEE Model number 3 has taken into account the affects of ambient temperature. By substituting the parameter values into Equation (5.2). The top-oil temperature was obtained as:

$$\theta_{top}[k] = \frac{5.53}{(5.53+0.25)} \times 27.2 + \frac{0.25}{(5.53+0.25)} \times 9 + \frac{0.25 \times 26.1 \times 9.1}{(5.53+0.25)(9.1+1)}\left(\frac{779}{915}\right)^{1.6}$$
$$+ \frac{0.25 \times 26.1}{(5.53+0.25)(9.1+1)} = 27.31°C$$

Once again, this value can then be checked with the actual value in Table 5.18.

Following in the same manner as above the results of the subsequent samples were tabulated for Models 2 and 3 in Table 5.18.

Sample Number	Measured	Predicted	
		using Model 2	using Model 3
1	27.2	27.2	27.2
2	27.3	26.9	27.3
3	27.4	26.6	27.5
4	27.7	26.4	27.6
5	27.8	26.1	27.7
6	28.0	25.9	27.9
7	28.1	25.7	28.1
8	28.2	25.5	28.3
9	28.2	25.3	28.5
10	28.3	25.1	28.6

Table 5.18: The first 10 samples of the measured and predicted top-oil temperature (27 MVA transformer) using Models 2 and 3.

As indicated in Table 5.18, Model 3 has produced a better result than Model 2. Therefore, Model 3 was chosen as for the conventional approach.

In a similar manner as above, the top-oil temperature for day 4.5 to 9 (sample numbers 300 to 598) can be predicted by using Model 3 as shown in Figure 5.38.

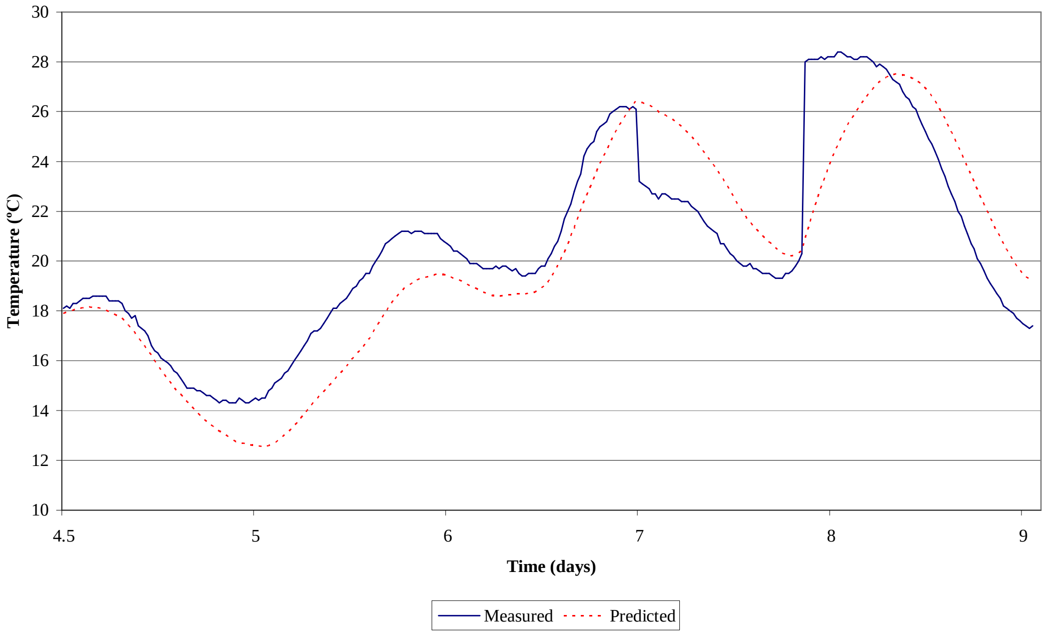

Figure 5.38: The measured and predicted top-oil temperature (27 MVA transformer) using Model 3.

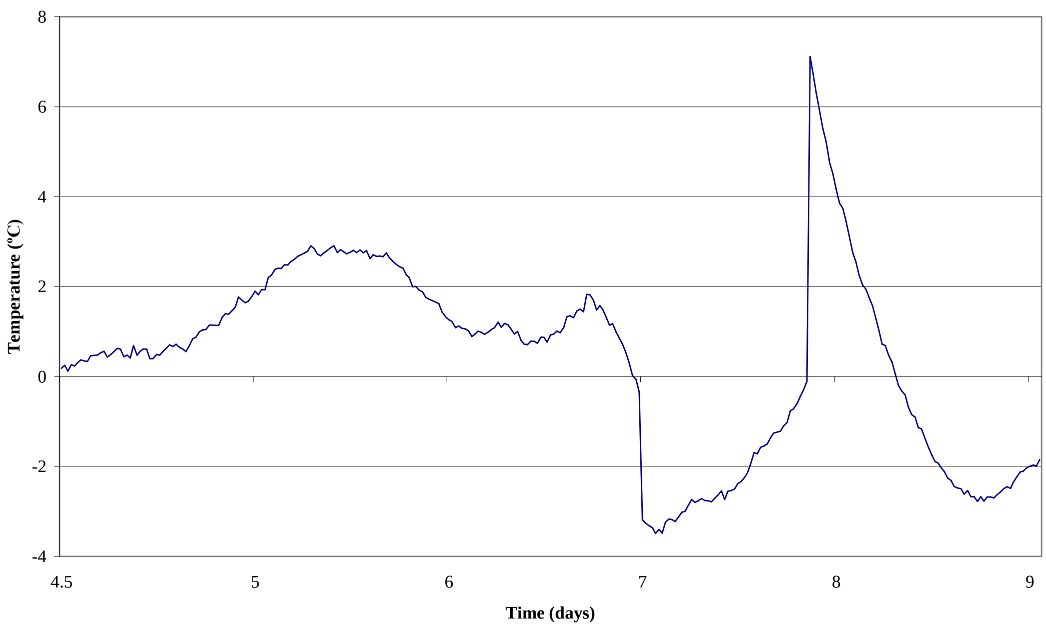

Figure 5.39: The error waveform of the measured and predicted top-oil temperature (27 MVA) by using Model 3.

A comparison of Model 2 and 3 results can be seen in Table 5.19.

Type of temperatures	Model	RMSE	MIN	MAX
Top-oil	2	4.7347	-0.0814	10.8086
	3	4.5194	-3.4876	7.1133

Table 5.19: A comparison of the predicted top-oil temperature (27 MVA transformer) by using Models 2 and 3.

5.4.2 Bottom-oil Temperature Prediction Using Models 2 and 3 (27 MVA Transformer)

Following the same methodology as in Section 5.4.1, the predicted bottom-oil temperature for the 27 MVA transformer were tabulated in Table 5.20.

Type of temperatures	Model	RMSE	MIN	MAX
Bottom-oil	2	14.6544	9.4169	21.0539
	3	10.9647	8.5946	16.7406

Table 5.20: A comparison of the predicted top-oil temperature (27 MVA transformer) by using Models 2 and 3.

5.4.3 The Temperatures Prediction by Using Models 2 and 3 (8 MVA Transformer)

The temperatures for the 8 MVA transformer can be obtained by following the same methodology as in Section 5.4.1.

A plot of the measured and predicted top-oil temperature was obtained by using Model 3 is shown in Figure 5.40. The error waveform can be seen in Figure 5.41.

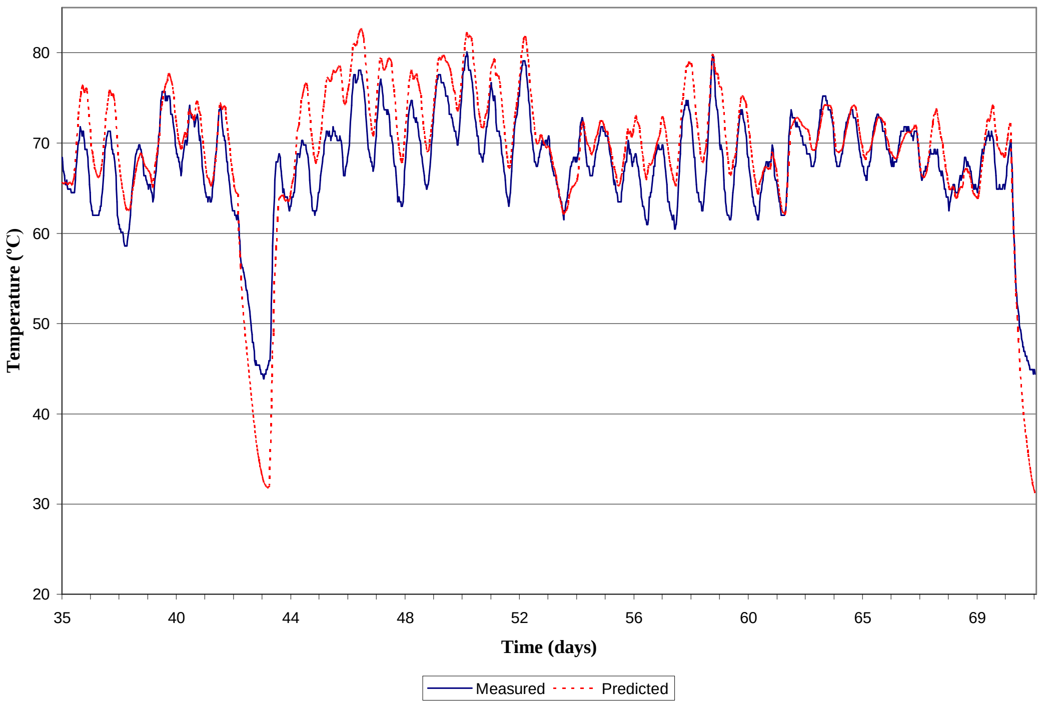

Figure 5.40: The measured and predicted top-oil temperature using Model 3.

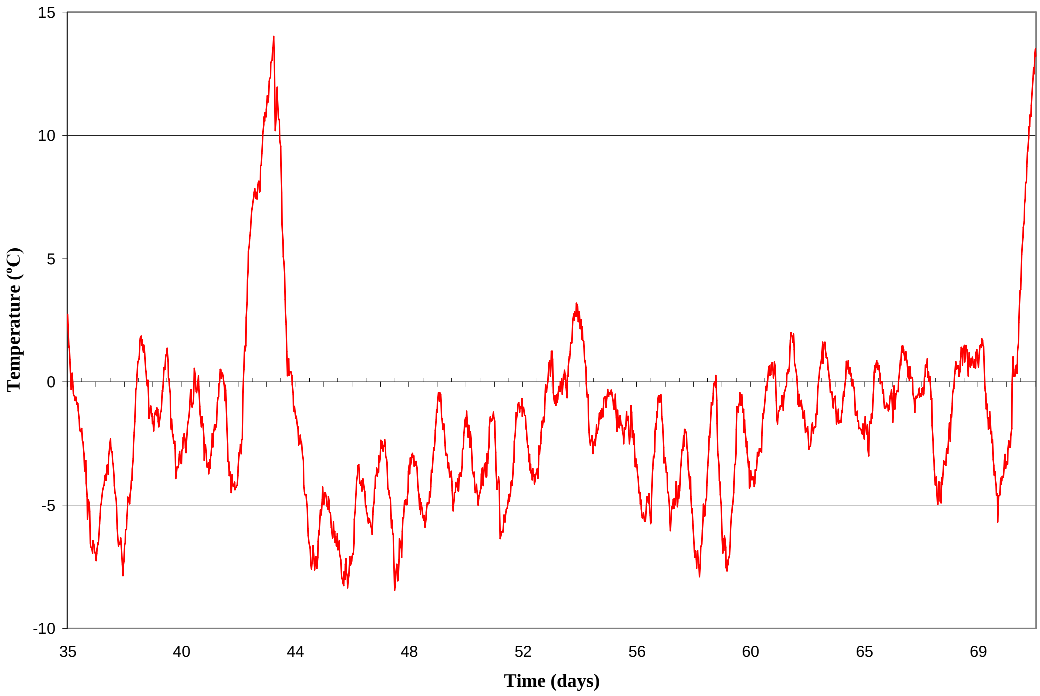

Figure 5.41: The error waveform of the predicted top-oil temperature (8 MVA transformer) by using Model 3.

The results of the top and bottom-oil temperatures prediction of the 8 MVA transformer by using Models 2 and 3 can be seen in Table 5.21 below.

Type of temperatures	Model	RMSE	MIN	MAX
Top-oil	2	24.1166	14.8103	38.7340
	3	4.0162	-8.4637	14.0173
Bottom-oil	2	17.2837	8.5972	23.9825
	3	8.7596	-12.9589	-2.2866

Table 5.21: A comparison of the predicted temperatures (8 MVA transformer) by using Models 2 and 3.

5.4.4 Summary

Two models have been used in this Section. They were Models 2 and 3 from the ANSI/IEEE standards. Model 3 was outperformed Model 2 in terms of RMSE and peaks of error as shown in Tables 5.19, 5.20 and 5.21. These results were consistent with (Lesieutre, 1997).

CHAPTER 6 : CONCLUSIONS AND RECOMMENDATIONS

In this thesis, a neural fuzzy technique has been applied for modeling and prediction of the transformer top and bottom-oil temperatures. A summary of the results of all models used in this thesis can be seen in Table 6.1 (a) and (b). Comparison with the commonly used ANSI/IEEE standard techniques showed superiority of the proposed technique in terms of predictability (RMSEs and peaks of error were improved comparatively), interpretability (besides load current and ambient temperature, top and bottom oil temperatures have a direct effect on each other), adaptability (tested on unseen data) and efficiency (tested on Pentium IV computer system with 512 MB of RAM, the computation time was less than 35 seconds) of reconstruction results. In addition, the developed technique shows a strong link between the top and bottom oil temperatures and also overcomes the problem of uncertainty and unavailability of transformer's parameters necessary when using the ANSI/IEEE numerical techniques. The described techniques are fast and easy to implement in any iterative reconstruction algorithm. Of the three soft computing models (MFNN, ERNN and ANFIS), the ANFIS model is improved over the MFNN and ERNN models in RMSE when used to predict the top-oil temperature of the 8 MVA transformer [with RMSE = 3.5299, minimum peak error = -23.2437 $^\circ$C and maximum peak error = 21.4646 $^\circ$C] and produces comparable outcomes when used to predict the bottom and top-oil temperatures for the 27 MVA transformer. When comparing the ANFIS model with the neural networks model, one has to average out the results over twenty different runs (training processes) for a given neural network due to the effect of arbitrary initialization of the neural network weights, whereas with the ANFIS model this process was not required.

Neuro-fuzzy models, like ANFIS, are well documented techniques which have been widely applied to other problems. In order to use it with a significant effect sufficient training data has to be collected, and data has to be representative of the problem for successful prediction. If the training data set size is increased, the prediction accuracy is higher. The practicality of the ANFIS technique can be concluded as follows: the load current is the most important predictor and the ambient temperature is less important and appears to be irrelevant for the bell and Gaussian MFs. The ANFIS model uses the grid partitioning method and there are only two choices for the output membership function: constant and linear. This limitation of output membership function choices is because ANFIS only operates on Sugeno-type systems.

(a)

Method	Transformer	Type of temperatures	RMSE	Min	Max
ANFIS	8 MVA	Top-oil	3.5299	-23.244	21.4646
		Bottom-oil	2.9099	-15.657	14.8042
	27 MVA	Top-oil	2.9608	-6.6134	2.7499
		Bottom-oil	2.9222	-6.6692	2.5354
MFNN	8 MVA	Top-oil	3.5667	-23.079	21.9265
		Bottom-oil	2.8971	-15.677	15.2908
	27 MVA	Top-oil	2.9102	-7.7301	1.9395
		Bottom-oil	2.9537	-7.4847	4.0262
ERNN	8 MVA	Top-oil	3.6439	-23.983	21.9549
		Bottom-oil	2.9579	-15.532	14.7587
	27 MVA	Top-oil	3.0149	-6.7041	1.1003
		Bottom-oil	2.9828	-6.3518	1.6420

(b)

Method	Transformer	Type of temperatures	MODEL	RMSE	Min	Max
IEEE MODELS	8 MVA	Top-oil	2	24.1166	14.8103	38.734
			3	4.0162	-8.4637	14.0173
		Bottom-oil	2	17.2837	8.5972	23.9825
			3	8.7596	-12.9589	-2.2866
	27 MVA	Top-oil	2	4.7347	-0.0814	10.8086
			3	4.5194	-3.4876	7.1133
		Bottom-oil	2	14.6544	9.1469	21.0539
			3	10.9647	8.5946	16.7406

Table 6.1: A summary of the results for all models.

REFERENCES

Abraham, A. (2001) Neuro Fuzzy Systems: State-of-the-Art Modeling Techniques. *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence,* 269-276.

Aquino, V. A., Sanchez, J. A. M., Romero, R. R. and Cruz, J. F. R., (2005) Dectecting and Classifying Attacks in Computer Networks Using Feed-Forward and Elman Neural Networks. *The First European Conference on Computer Network Defence.*

Boyd, S. and Vandenberghe, L., (2004) Convex Optimization. Cambridge University Press.

Brouwer, R. K. (2004) Fuzzy Rule Extraction from a Feed forward Neural Network by Training a representative Fuzzy Neural Network Using Gradient Descent. *IEEE International Conference on Industrial Technology*, pp. 1168 - 1172.

Burns, R. S. (2001) Advanced Control Engineering. Butterworth-Heinemann.

Cai, C. H., Du, D. and Liu, Z. Y., (2003) Battery State-of-Charge Estimation Using Adaptive Neuro-Fuzzy Inference System (ANFIS). *IEEE International Conference on Fuzzy Systems*, pp. 1068-1073.

Cortes, C., Jackel, L. D. and Chiang, W. P., (1995) Limits on Learning Machine Accuracy Imposed by Data Quality. *Neural Information Processing Systems*, pp. 239 - 246.

Demuth, H. and Beale, M., (1998) Neural Network Toolbox for use with MATLAB. The MathWorks Inc.

El Choubassi, M. M., El Khoury, H. E., Alagha, C. E. J., Skaf, J. A. and Al-Alaoui, M. A., (2003) Arabic Speech Recognition Using Recurrent Neural Networks. *The 3rd IEEE Internal Symposium on Signal Processing and Information Technology*, pp. 543-547.

Elman, J. (1990) Finding Structure in Time. *Cognitive Science*, vol. 14. pp. 179 - 211.

Feuchter, B. and Feser, K., (1993) On-Line Diagnostic of The Thermal Behaviour of Power Transformers. *The 8th International Symposium on High Voltage Engineering.* pp. 125 - 128.

Hagan, M. T. and Menhaj, M. B., (1994) Training Feedforward Networks with The Marquardt Algorithm. *IEEE Transactions on Neural Networks,* vol. 6. pp. 989 - 993.

Hassoun, M. H. (1995) Fundamentals of Artificial Neural Networks. MIT Press.

Haykin, S. (1994) Neural Networks: A Comprehensive Foundation. Prentice Hall International.

Hopfield, J. J. (1982) Neural Networks and Physical Systems with emergent collective computational abilities. *Proceedings of the National Academy of Science.* pp. 2554 - 2558.

IEC (1993) IEC 72-1 Power Transformers Part1: General. Second Edition.

IEEE (1995) IEEE Standard C57.115-1995 IEEE Guide for loading Mineral-Oil-Immersed Power Transformers.

Jamshidi, M. (1997) Large-Scale Systems: Modelling, Control and Fuzzy Logic. Prentice Hall.

Jang, J. -S. R. (1993) ANFIS: Adaptive Network-Based-Fuzzy Inference System. *IEEe Transactions on Systems, Man, and Cybernetics,* Vol. 23. No. 3, pp. 665-685.

Jang, J. -S. R., Sun, C. -T. and Mizutani E. (1997) Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine intelligence. Upper Saddle River, NJ, Prentice Hall.

Jordan, M. (1986) Serial order: A Parallel Distributed Processing Approach. *Technical Report 8604.* Institute for Cognitive Science, University of California, San Diego.

Kandel, A. (1992) Fuzzy Expert Systems.

Kosko, B. (1991) Neural Networks and Fuzzy Systems: A Dynamical Systems Approach. Prentice Hall.

Lee, C. - C. (1990) Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I and II. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 20. No. 2, pp. 404 - 435.

Lesieutre, B. C., Hagman, W. H. and Kirtley, Jr. J. L., (1997) An Improved Transformer Top Oil Temperature Model for Use in An On-Line Monitoring and Diagnostic System. *IEEE Transactions on Power Delivery,* Vol. 12. pp. 249 - 254.

References

Levenberg, K. (1944) A Method for The Solution of Certain Problems in Least Squares. *Quaterly of Applied Mathematics,* pp. 164 - 168.

Ljung, L. (1987) System Identification: Theory for the User. Englewood Cliffs, NJ, Prentice-Hall.

Mamdani, E. H. and Assilan, S., (1975) An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies,* Vol. 7. No. 1, pp. 1 - 13.

Marashdeh, Q., Warsito, W., Fan, L-S. and Teixeira, F. L., (2006) Nonlinear Forward Problem Solution for Electrical Capacitance Tomography Using Feed-Forward Neural Network. *IEEE Sensors* Vol. 6. 2, pp. 441 - 449.

Marquardt, D. W. (1963) An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society of Industrial and Applied Mathematics,* pp. 431 - 441.

Mastacam, L., Olah I., Dosoftei C. C. and Ivana D., (2005) Neuro-Fuzzy Models of Thermoelectric Power Station Installations. *IEEE International Conference on Computational Intelligence for Modelling, Control and Automation.*

McCulloch, W. S. and Pitts, W., (1943) A Logical Calculus of The ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics,* Vol. 5. pp. 115 - 133.

Mirhassani, M., Ahmadi, M. and Miller, W. C., (2005) Design and Implementation of Novel Multi-Layer Mixed-Signals On-Chip Neural Networks. *the 48th Midwest Symposium on Circuits and Systems,* vol. 1. pp. 413 - 416.

Narendra, K. S. and Parthasara, K. (1990) Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks,* pp. 4 - 27.

Nguyen, H., Shi, J. and Reznik, L. (2002) A Neural Fuzzy Approach to Modelling The Top Oil Temperature of Power Transformers. *Proceedings of the 4th International Conference on Modelling and Simulation.* pp. 27 - 31.

Pierce, L. W. (1992) An investigation of The Thermal performance of an oil-filled transformer winding. *IEEE Transactions on Power Delivery,* Vol. 7. pp. 1347 - 1356.

Pierce, L. W. (1994) Predicting Liquid Filled transformer Loading Capability. *IEEE Transactions on Industry Applications,* Vol. 30. pp. 170 - 178.

Rosenblatt, F. (1958) The Perceptron: A Probablistic Model for Information Storage and Organization in The Brain *Psycho. Rev.,* Vol. 25. pp. 386 - 408.

Rosenblatt, F. (1962) Principles of Neurodynamics. Washington, DC, Spartan Books.

Rumelhart, D. E., Hinton G. E. and Williams R. J., (1986) Learning Internal Representations by Error Propagation. IN Rumlhart D. E. and McClelland J. L. (Ed.) *Parallel Distributed Processing.* Cambridge, MA., MIT Press.

Sugeno, M. and Kang, T. G., (1988) Structure identification of fuzzy model. *Fuzzy Sets and Systems.* Vol. 28. pp. 15 - 33.

Takagi, T. and Sugeno, M., (1983) Derivation of Fuzzy Logic Control Rules from Human Operators Control Actions. *IFAC Symposium on Fuzzy Information Representation and Decision Analysis.* pp. 55 - 60.

The MathWorks, Inc. (1999) Fuzzy Logic Toolbox. *The MathWorks Inc,*

Thukaram, D., Shenoy, U. J., and Ashageetha, H., (2006) Neural Network Approach for Fault Location in Unbalanced Distribution Networks with Limited Measurements. *IEEE Power India Conference.*

Tsakourmis, A. C., Vladov, S. S. and Mladenov, V. M. (2002) Electric Load Forecasting with Multilayer Peceptron and Elman Neural Network. *NEUREL '02. 2002 6th Seminar on Neural Network Applications in Electrical Engineering,* pp. 87 - 90.

Tsukamoto, Y. (1979) An Approach to Fuzzy Reasoning Method. IN Gupta M. M., Ragade K. K. and Yager R. R. (Ed.) *Advances in Fuzzy Set Theory and Applications.* North Holland, Amsterdam,

Tylavsky, D. J., He, Q., McCulla, G. A. and Hunt, J. R., (2000) Sources of Error in Substation Distribution Transformer Dynamic Thermal Modeling. *IEEE Transactions on Power Delivery,* Vol. 15. No. 1, pp. 178 - 185.

Vitabile, S., Pernice, M. and Gaglio, S., (2004) Daily Peak Temperature Forecasting with Elman Neural Networks. *IEEE International Joint Conference on Neural Networks,* Vol. 4. pp. 2765 - 2769.

Werbos, P.J. (1974) Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences. *PhD Thesis.* Harvard University.

Yager, R. R. and Filev, D. P., (1994) Essentials of Fuzzy Modeling and Control. New York, John Wiley and Sons.

References

Ying, H. (2000) Fuzzy Control and Modeling: Analytical Foundations and Applications. New York, IEEE Press.

Zadeh, L. A. (1965) Fuzzy Sets. *Information and Control,* Vol. 8. pp. 338-353.

Zadeh, L. A. (1973) Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 3. No. 1, pp. 28-44.

Zimmermann, H. J. (1985) Fuzzy Set Theory and Its Applications. Boston, Kluwer Academic.

APPENDIX A: Transformers Details

8 MVA TRANSFORMER

Application	Power Transformer
Transformer name	8 MVA OA-cooled
Serial No	28629
Manufacturer	ASEA
Year of manufacture	1990
Years in operation	5yrs
Rating, kVA	8000 kVA
Voltage, kV	11kv/2,96kv
Core/shell type	Core
Cooling type	OA
Oil type	Napthenic
Oil capacity	6695 L
Oil preservation	Conservator
Cooling elements	Finned radiators

27 MVA TRANSFORMER

Application	Power Transformer
Transformer name	27 MVA FA-cooled
Serial No	M162241B
Manufacturer	G.E.
Year of manufacture	1991
Years in operation	8
Rating, kVA	27000
Voltage, kV	138 - 13.8
Core/shell type	Core
Cooling type	FA
Oil type	TYPE I
Oil capacity	10500 gal
Oil preservation	N2 GAS SPACE
Cooling elements	Radiators

APPENDIX A CON'T: Transformers Data

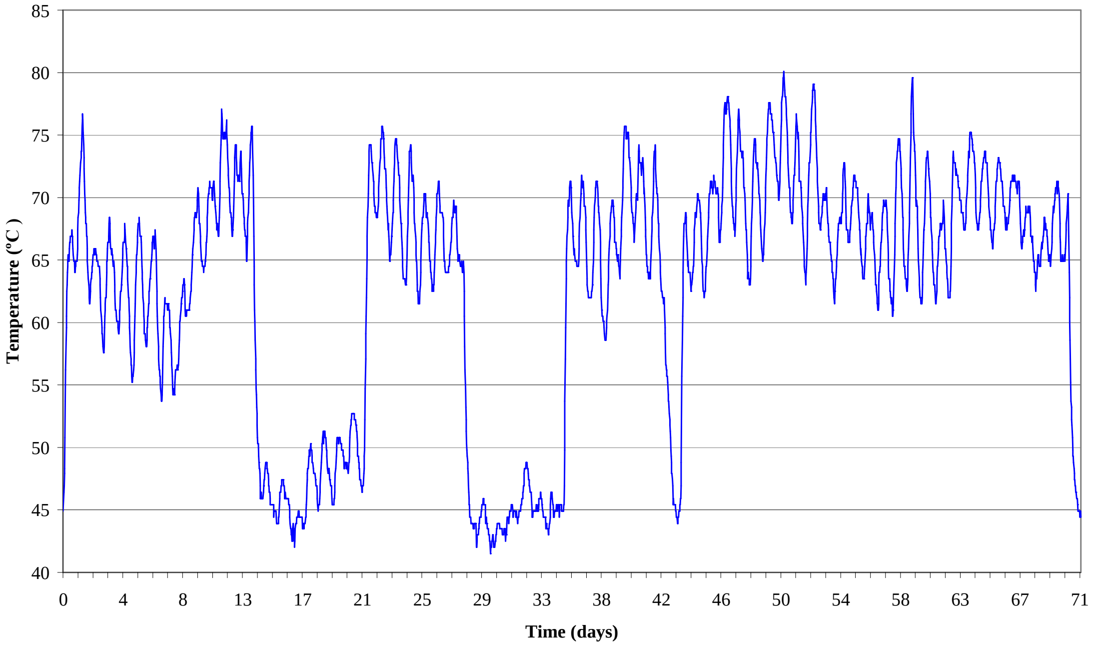

Figure A.1: The measured top-oil temperature $\theta_{\mathrm{T1\_top\_m}}$ waveform of the 8 MVA transformer.

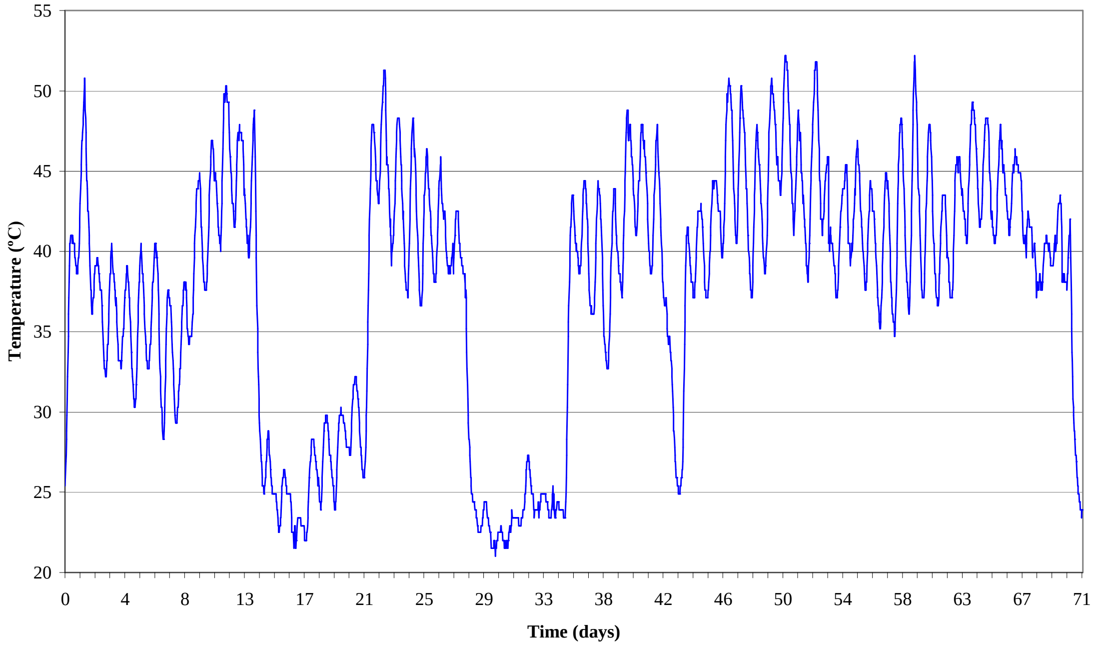

Figure A.2: The measured bottom oil temperature $\theta_{\mathrm{T1\_bot\_m}}$ of the 8 MVA transformer.

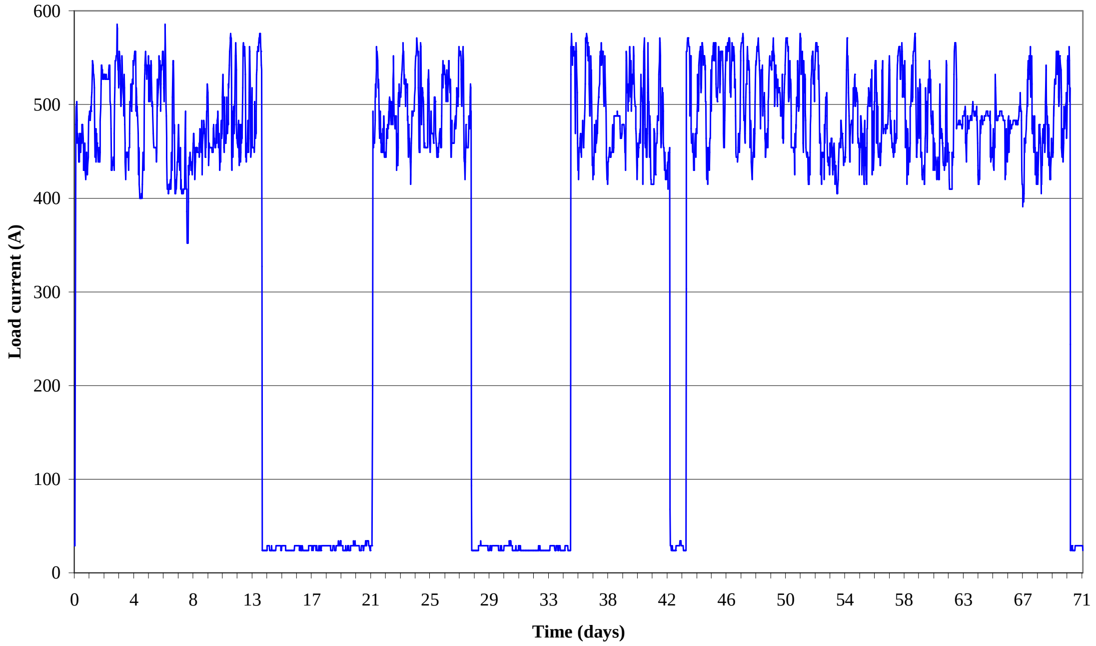

Figure A.3: The measured load current $I_{\mathrm{T1\_load\_m}}$ waveform of the 8 MVA OA-cooled transformer.

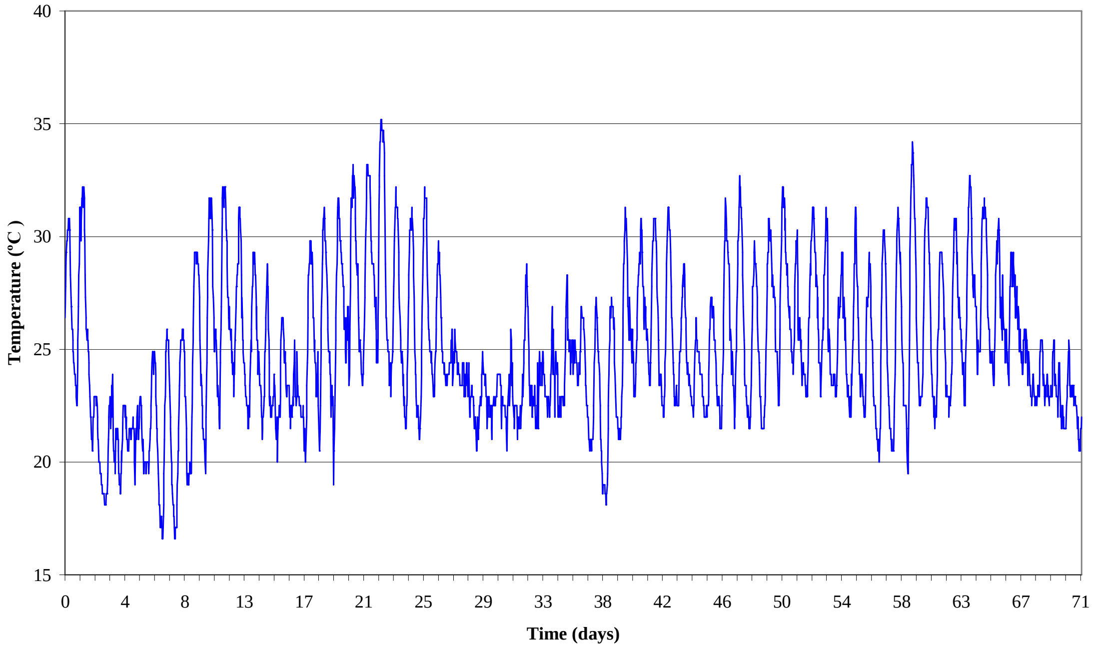

Figure A.4: The measured ambient temperature $\theta_{T1\_amb\_m}$ waveform of the 8 MVA transformer.

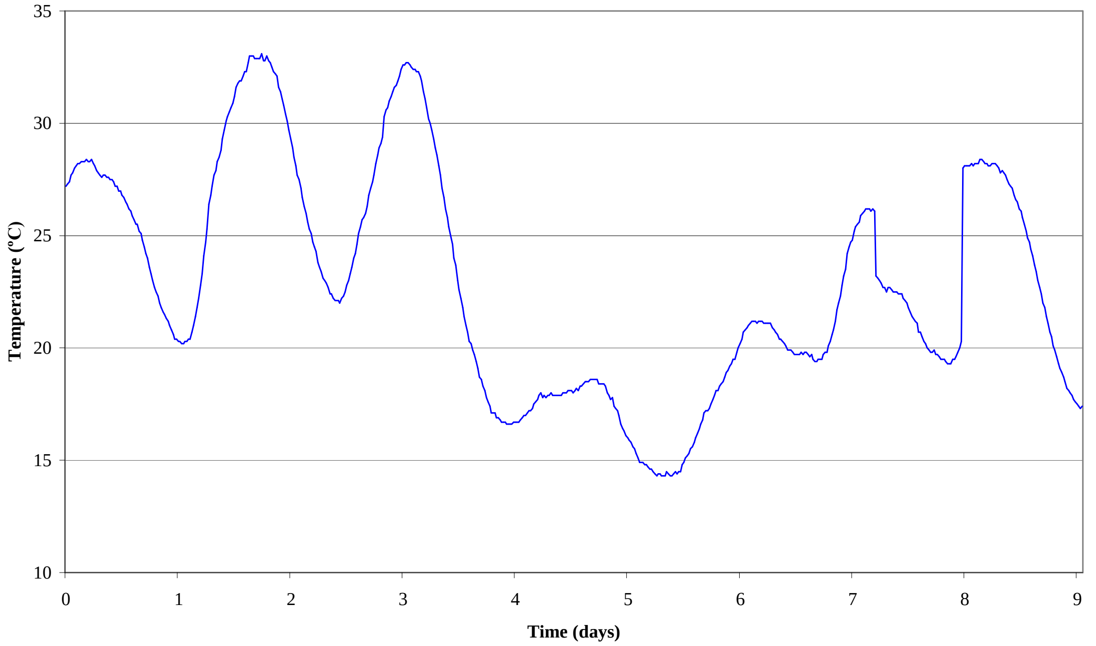

Figure A.5: The measured top-oil temperature $\theta_{T2\_top\_m}$ waveform of the 27 MVA transformer.

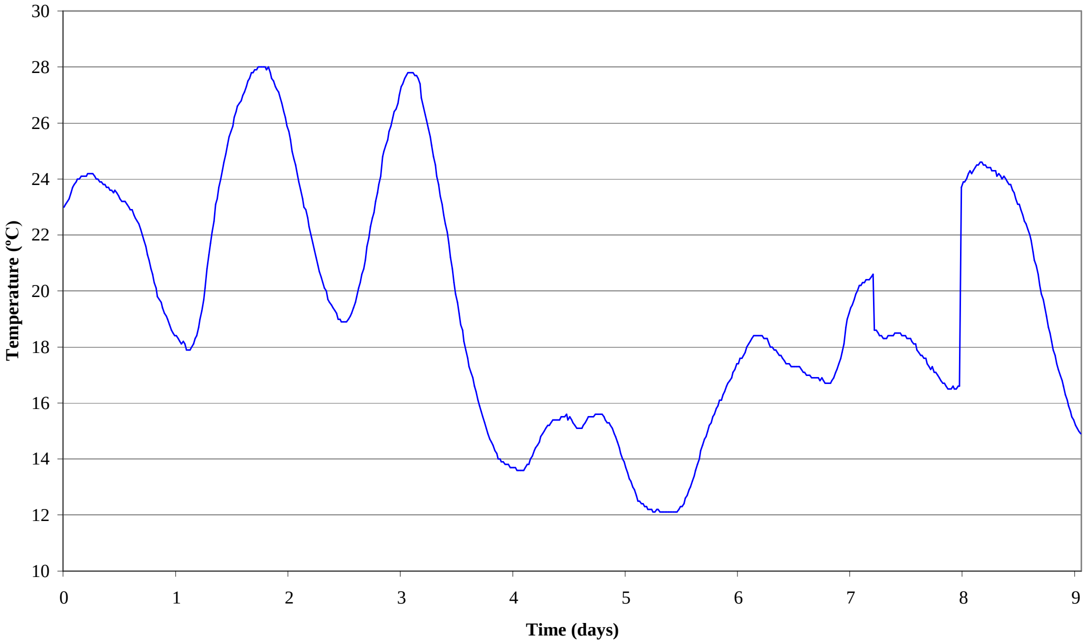

Figure A.6: The measured bottom oil temperature $\theta_{\mathrm{T2\_bot\_m}}$ waveform of the 27 MVA transformer.

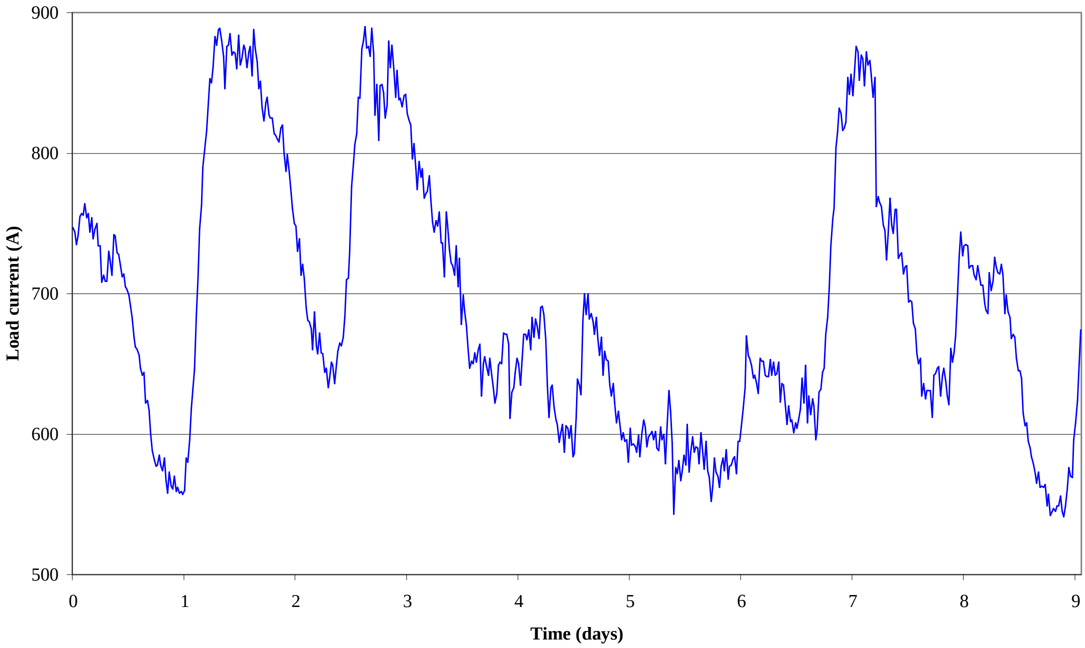

Figure A.7: The measured load current $I_{\mathrm{T2\_load\_m}}$ waveform of the 27 MVA transformer.

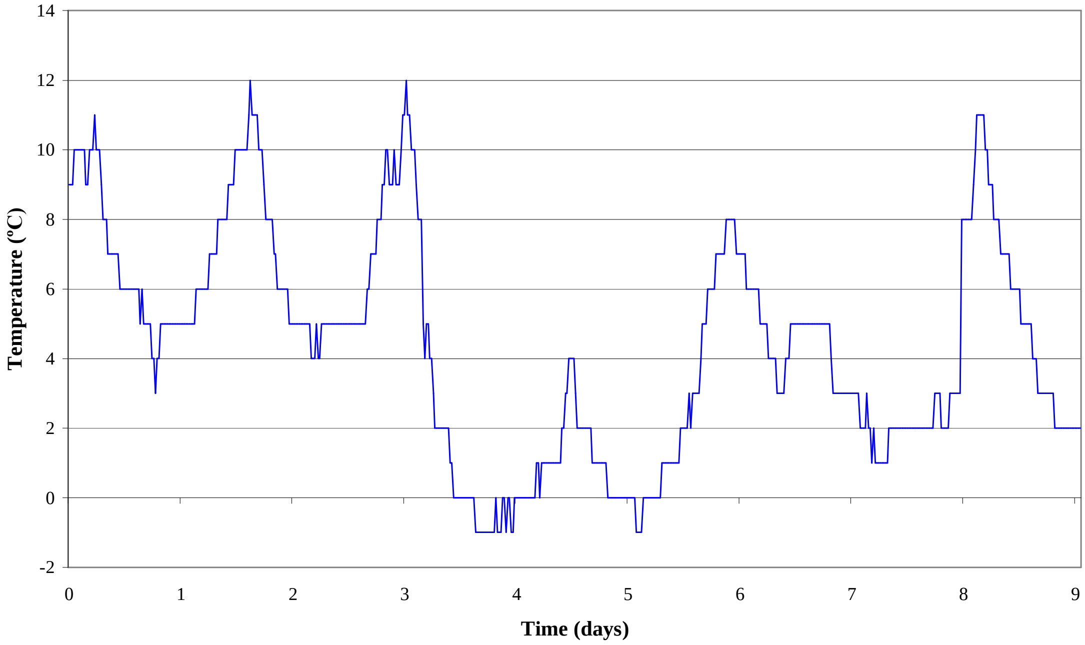

Figure A.8: The measured ambient temperature $\theta_{T2\_amb\_m}$ waveform of the 27 MVA transformer.

APPENDIX B: The 3-Input ANFIS Model Matlab Files

```
% Computer Program for training and testing the ANFIS model when used
% to predict the top-oil temperature of the 8 MVA transformer.
% See Figures 5.21, 5.22, 5.23 and 5.24 of the thesis.
% Huy Nguyen, 2007.

close all
clear all

% Load the input training data set i.e 1702 samples over 35.5 days
load input_train.dat;
% Load the input test data set i.e 1702 samples over 35.5 days
load input_test.dat;
% Load the output training data set i.e 1702 samples over 35.5 days
load output_train_top.dat;
% Load the output test data set i.e 1702 samples over 35.5 days
load output_test_top.dat;

% Let x1 be the input training data set
x1=input_train;
% Let x2 be the input test data set
x2=input_test;
% Let y1 be the output training data set
y1=output_train_top;
% Let y2 be the output test data set
y2=output_test_top;
% To create a training data set
traindata=[x1 y1];
% To create a test data set
testdata =[x2 y2];

%=======================================================================
% Genfis1: generates an initial Sugeno-type FIS for the ANFIS training
% using a grid partition.
% Number of membership functions: 2
% Type of membership functions: gaussmf
%=======================================================================
mfType = str2mat('gaussmf','gaussmf','gaussmf');

% To generate an initial FIS
in_fis = genfis1(traindata,2,mfType,'linear');

%=======================================================================
% ANFIS: Adaptive Neuro-Fuzzy training of Sugeno-type FIS.
% Epoch number: 100
%=======================================================================
numEpochs=100;
[fismat1,trnErr,ss,fismat2,tstErr]=anfis(traindata,in_fis,[numEpochs 0
0.04 0.9 1.1],[],testdata,1);
numepochs=1:numEpochs;
figure (1)
subplot(2,1,1);
```

```
plot(numepochs,trnErr,'o',numepochs,tstErr,'x');
ylabel('RMSE');
xlabel('Epochs');
legend('Training error','Testing error');
hold on
plot(numepochs,[trnErr tstErr]);
hold off
subplot(2,1,2);
plot(numepochs,ss,'-',numepochs,ss,'x');
ylabel('Step Size');
xlabel('Epochs');

%=======================================================================
% Evalfis: performs fuzzy inference calculations.
% Z: the predicted top-oil temperature
% Using the input test data set to determine the top-oil temperature
%=======================================================================
z = evalfis(x2,fismat2);

%=======================================================================
% To export the output to a file
%=======================================================================
fid = fopen('ANFIS_3INPUT.txt', 'w');
fprintf(fid,'%6.2f\n',z);
fclose(fid);

%=======================================================================
% A plot of the measured and predicted top-oil temperature versus the
% testing error
%=======================================================================
e = y2 - z;
numPts=1702;
for i=1:numPts
p(i)=e(i)^2;
end
s=1:1702;
t=s/48;
figure (2)
% Plot the error between the measured and the predicted top-oil
% temperature
plot(t,e,'g');
hold
% Plot the measured top-oil temperature
plot(t,y2,'r');
hold on
% Plot the predicted top-oil temperature
plot(t,z,'b');
hold off
grid;
ylabel('Temperature (degree C)');
xlabel('Time (days)');
legend('error','measured','predicted');
title('Prediction of the top-oil temperature by using the ANFIS
model');

figure (3)
```

```
subplot(3,3,1); plotmf(in_fis,'input',1);
title('(a) Initial MFs for Ambient temp.')
xlabel('Ambient temperature (degree C)')
subplot(3,3,2); plotmf(in_fis,'input',2);
title('(b) Initial MFs for Bottom oil temp.')
xlabel('Bottom oil temperature (degree C)')
subplot(3,3,3);plotmf(in_fis,'input',3);
title('(c) Initial MFs for Load current')
xlabel('Load current (A)')

subplot(3,3,4); plotmf(fismat1,'input',1);
title('(d) Final MFs with training data for Ambient temp.')
xlabel('Ambient temperature (degree C)')
subplot(3,3,5); plotmf(fismat1,'input',2);
title('(e) Final MFs with training data for Bottom oil temp.')
xlabel('Bottom oil temperature (degree C)')
subplot(3,3,6); plotmf(fismat1,'input',3);
title('(f) Final MFs with training data for Load current')
xlabel('Load current (A)')


subplot(3,3,7); plotmf(fismat2,'input',1);
title('(g) Final MFs with test data for Ambient temp.')
xlabel('Ambient temperature (degree C)')
subplot(3,3,8); plotmf(fismat2,'input',2);
title('(h) Final MFs with test data for Bottom oil temp.')
xlabel('Bottom oil temperature (degree C)')
subplot(3,3,9); plotmf(fismat2,'input',3);
title('(i) Final MFs with test data for Load current')
xlabel('Load current (A)')

% Calculate RMSE and peaks of error
rmse=sqrt(sum(p)/1702)
minimum=min(e)
maximum=max(e)
```

APPENDIX C: The 3-Input MFNN Model Matlab Files

```
% Computer Program for training and testing the Multilayer Feedforward
% Neural Network when used to predict the top-oil temperature of the 8
% MVA transformer.
% See Figures 5.36 and 5.37 of the thesis.
% Huy Nguyen, 2007.

clear all
close all

% Load the training and test data sets into the workspace
load input_train.dat;
load input_test.dat;
load output_train_top.dat;
load output_test_top.dat;

% Let x1 and y1 be the input and output for training data set
% respectively.
% Let x2 and y2 be the input and output for test data set respectively.
x1=input_train.'
x2=input_test.'
y1=output_train_top.'
y2=output_test_top.'

% Initialize the network
% Backpropagation network training function:
% Levenberg-Marquardt, i.e. trainlm.
% Backpropagation weight/bias learning function:
% Gradient descent, i.e. learngd.
% Performance function: Mean Square Error, i.e. mse
net=newff(minmax(x1),[8 1],{'logsig'
'purelin'},'trainlm','learngd','mse');

% Initialize some of the parameters for backpropagation training
net.trainParam.epochs = 100; % Maximum number of epochs to train
net.trainParam.lr = 0.01; % Learning rate
net.trainParam.show = 50; % Epochs between updating display
net.trainParam.goal = 0; % Sum-squared error goal

% Train the network with net.trainParam.epochs
net=train(net,x1,y1)

% Test the trained network with the unseen data
output=sim(net,x2);

% Write output to a file
fid = fopen('MFNN_3INPUT.txt', 'w');
fprintf(fid,'%6.2f\n',output);
fclose(fid);

% Calculate the residual between the measured and predicted top-oil
% temperature
```

```
e= y2 - output;

numPts=1702;
for i=1:numPts;
p(i)=e(i)^2;
end
s=1:1702;
t=s/48;

% Plot the measured and predicted top-oil temperature waveforms
figure (3)
plot(t,e,'g');
hold
plot(t,y2,'r');
hold on
plot(t,output,'b');
grid;
ylabel('Temperature (degree C)');
xlabel('Time (days)');
legend('error','measured','predicted');
title('Prediction of the top-oil temperature by using the MFNN model');

% Display weights and biases
W1=net.IW{1,1}
b1=net.b{1,1}
W2=net.LW{2,1}
b2=net.b{2,1}

% Calculate RMSE and peaks of error
rmse=sqrt(sum(p)/1702)
minimum = min(e)
maximum = max(e)
```

APPENDIX D: The 3-Input ERNN Model Matlab Files

```
% Computer Program for training and testing the Elman Recurrent Neural
% Network model when used to predict the top-oil temperature of the 8
% MVA transformer.
% See Table 5.16 of the Thesis.
% Huy Nguyen, 2007.

clear all
close all

% Load data sets
load input_train.dat;
load input_test.dat;
load output_train_top.dat;
load output_test_top.dat;

% Let x1 and y1 be the input and output for training data set
% respectively.
% Let x2 and y2 be the input and output for test data set respectively.
x1=input_train.'
x2=input_test.'
y1=output_train_top.'
y2=output_test_top.'

% Initialize the network
% Backpropagation network training function:
% Levenberg-Marquardt, i.e. trainlm.
net=newelm(minmax(x1),[8 1],{'logsig' 'purelin'},'trainlm');

%Initialize some of the parameters for backpropagation training
net.trainParam.epochs = 100; % Maximum number of epochs to train
net.trainParam.lr = 0.01; % Learning rate
net.trainParam.show = 50; % Epochs between updating display
net.trainParam.goal = 0; % Sum-squared error goal

% Train the network with net.trainParam.epochs
net=train(net,x1,y1)

% Test the trained network with unseen data
output=sim(net,x2);

% Write output to a file
fid = fopen('ERNN_3INPUT.txt', 'w');
fprintf(fid,'%6.2f\n',output);
fclose(fid);

% Calculate the difference between the measured and predicted
e= y2 - output;

numPts=1702;
for i=1:numPts;
p(i)=e(i)^2;
```

```
end
s=1:1702;
t=s/48;

% Plot the measured and predicted waveforms
figure (2)
plot(t,e,'g');
hold
plot(t,y2,'r');
hold on
plot(t,output,'b');
grid;
ylabel('Temperature (degree C)');
xlabel('Time (days)');
legend('error','measured','predicted');
title('Prediction of top-oil temperature using ELMAN neural network
model');

% Display weights and biases
W1=net.IW{1,1}
b1=net.b{1,1}
W2=net.LW{2,1}
b2=net.b{2,1}

% Calculate RMSE and peaks of error
rmse=sqrt(sum(p)/1702)
minimum = min(e)
maximum = max(e)
```