INVESTIGATION OF METHODOLOGIES FOR FAULT DETECTION AND DIAGNOSIS IN ELECTRIC POWER SYSTEM PROTECTION

by

ADEYEMI CHARLES ADEWOLE

Thesis Submitted in fulfilment of the requirements for the degree

Master of Technology: Electrical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Prof R. Tzoneva
Co-supervisor: Adj. Prof. P. Petev

Bellville
October, 2012

DECLARATION

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

Signed **Date**

The widespread deregulation and restructuring of electric power utilities throughout the world and the surge in competition amongst utility companies has brought about the desire for improved economic efficiency of electric utilities and the provision of better service to energy consumers. These end users are usually connected to the distribution network. Thus, there is a growing research interest in distribution network fault detection and diagnosis algorithms for reducing the down-time due to faults. This is done so as to improve the reliability indices of utility companies and enhance the availability of power supply to customers.

The application of signal processing and computational intelligence techniques in power systems protection, automation, and control cannot be overemphasized. This research work focuses on power system distribution network and is aimed at the development of versatile algorithms capable of accurate fault detection and diagnosis of all fault types for operation in balanced/unbalanced distribution networks, under varying fault resistances, fault inception angles, load angles, and system operating conditions.

Therefore, different simulation scenarios encompassing various fault types at several locations with different load angles, fault resistances, fault inception angles, capacitor switching, and load switching were applied to the IEEE 34 Node Test Feeder in order to generate the data needed. In particular, the effects of system changes were investigated by integrating various Distributed Generators (DGs) into the distribution feeder. The length of the feeder was also extended and investigations carried out. This was implemented by modelling the IEEE 34-node benchmark test feeder in DIgSILENT PowerFactory (DPF).

In the course of this research, a hybrid combination of Discrete Wavelet Transform (DWT), decision-taking rule-based algorithms, and Artificial Neural Networks (ANNs) algorithms for electric power distribution network fault detection and diagnosis was developed. The integrated algorithms were capable of fault detection, fault type classification, identification of the faulty line segment, and fault location respectively.

Several scenarios were simulated in the test feeder. The resulting waveforms were exported as ASCII or COMTRADE files to MATLAB for DWT signal processing. Experiments with various DWT mother wavelets were carried out on the waveforms obtained from the simulations. In particular, Daubechies db-2, db-3, db-4, db-5, and db-8 were considered. Others are Coiflet-3 and Symlet-4 mother wavelets respectively. The energy and entropy of the detail coefficients for each decomposition level based on a sampling frequency of 7.68 kHz were analysed. The best decomposition level for the diagnostic tasks was then selected based on the analysis of the wavelet energies and entropy in each level of decomposition. Consequently, level-1 db-4 detail coefficients were selected for the fault detection task, while level-5 db4 detail coefficients were used to compute the wavelet entropy per unit indices which were then used for fault classification, fault section identification, and fault location tasks respectively.

Decision-taking rule-based algorithms were used for the fault detection and fault classification tasks respectively. The fault detection task verifies if a fault did indeed occur or not, while the fault classification task determines the fault class and the faulted phase(s). Similarly, Artificial Neural Networks (ANNs) were used for the fault section identification and fault location tasks respectively. For the fault section identification task, the ANNs were trained for pattern classification to identify the lateral or segment affected by the fault. Conversely, the fault location ANNs were trained for function approximation to predict the location of the fault from the substation in kilometres.

Also, the IEEE 13 Node Benchmark Test Feeder was modelled in RSCAD software and batch mode simulations were carried out using the Real-Time Digital Simulator (RTDS) as a 'proof of concept' for the proposed method, in order to demonstrate the scalability, and to further validate the developed algorithms. The COMTRADE files of disturbance records retrieved from an external IED connected in closed-loop with the RTDS and the runtime simulation waveforms were used as test inputs to the developed Hybrid Fault Detection and Diagnosis (HFDD) method.

Comparison of the method based on entropy with statistical methods based on standard deviation and Mean Absolute Deviation (MAD) has shown that the method based on entropy is very reliable, accurate, and robust. Results of preliminary studies carried out showed that the proposed HFDD method can be applied to any power system network irrespective of changes in the operating characteristics. However, certain decision indices would change and the decision-taking rules and ANN algorithms would need to be updated.
The HFDD method is promising and would serve as a useful decision support tool for system operators and engineers to aid them in fault diagnosis thereby helping to reduce system down-time and improve the reliability and availability of electric power supply.

Key words: Artificial neural network, discrete wavelet transform, distribution network, fault simulation, fault detection and diagnosis, power system protection, RTDS.

ACKNOWLEDGEMENTS

I am indeed grateful to my supervisor, Prof. R. Tzoneva, whose encouragement, guidance, and support helped me throughout my course of study.

I am also grateful to Messrs. C. Kriger and S. Behardien for their comments, insights, and advice at presentations held and at other times during my research.

I am thankful to Inga Witbooi and my friends for their support and understanding throughout my seclusion period.

Lastly, but by no means the least, I am grateful to my entire family members for their motivation and support.

Acknowledgements can go on and on, but some people will still be left out. To those I have unknowingly left out but were helpful to me, please accept my gratitude.

Gloria Tibi Domine.

Adeyemi Charles Adewole
Bellville, October 2012

TABLE OF CONTENTS

CHAPTER SIX: RESULTS AND DISCUSSION

LIST OF FIGURES

LIST OF TABLES

APPENDICES

Abbreviations	Definition/Explanation
ANN	Artificial Neural Network
BP	Back Propagation
BR	Bayesian Regularization
COMTRADE file	Common Format for Transient Data Exchange file
CSAEM	Centre for Substation Automation and Energy Management Systems
CT	Current Transformer
CWT	Continuous Wavelet Transform
DFRs	Digital Fault Recorders
DFT	Discrete Fourier Transform
DG	Distributed Generator
DIgSILENT	Digital SImuLator for Electrical NeTwork
DMS	Distribution Management System
DPF	DIgSILENT PowerFactory
DWT	Discrete Wavelet Transform
EMS	Energy Management System
FCI	Faulted Circuit Indicators
FT	Fourier Transform
FFT	Fast Fourier Transform
FCIs	Fault Current Indicators
FSI	Fault Section Identification
FL	Fault Location
GPS	Global Positioning System
GUI	Graphical User Interface
HFDD	Hybrid Fault Detection and Diagnosis
IEDs	Intelligent Electronic Devices

IP	Internet Protocol
L-M	Levenberg-Marquardt Algorithm
LVQ	Learning vector Quantization
MAD	Mean Absolute Deviation
MTTR	Mean Time To Fault
MV/LV	Medium Voltage/Low Voltage
NN	Neural Network
OSS	One-Step Secant Training Algorithm
RBF	Radial Basis Function
RP	Resilient Propagation Training Algorithm
RTDS	Real-Time Digital Simulator
SCADA	Supervisory Control and Data Acquisition
SCG	Scaled Conjugate Gradient Training Algorithm
SEL	Schweitzer Engineering Laboratories
STFT	Short-Time Fourier Transform
TWR	Travelling Wave Recorder
WEE	Wavelet Energy Entropy
50G1P	Residual Ground Instantaneous Overcurrent Level 1 Pickup
50P1P	Phase Instantaneous Overcurrent Level 1 Pickup
67GID	Residual Ground Instantaneous Overcurrent Level 1 Time Delay
67PID	Phase Instantaneous Overcurrent Level 1 Time Delay

I_{a1}	positive sequence fault currents of phase A
I_{a2}	negative sequence fault currents of phase A
I_{a0}	zero sequence fault currents of phase A
θ_f	fault inception angle
R_f	fault resistance
g	high pass filter
h	low pass filter
E_{jk}	wavelet energy of signal at scale j instant k
E_j	sum of all the signal's energy at scale j , $k=1,2,...,N$
N	N is the total number of instants
P_{jk}	relative wavelet energy
WEE_j	wavelet energy entropy
r_{a1}, r_{b1}, r_{c1}	WEE of phases A, B, and C for level-1 detail coefficients
ζ_d	fault detection threshold
λ_{p5}	phase entropy calculated per unit for a given fault
ζ_{ca}, ζ_{cb}, ζ_{cc} and ζ_{cI0}	classification thresholds (derived from the three phase and zero sequence entropy)
I-H-O	Input-Hidden-Output layers of a NN
$\psi(t)$	DWT mother wavelet
cD1	detail coefficient of DWT
cA1	approximation coefficient of DWT
w_{kj}	weight coefficient in a NN
x_j	input of a neuron in a NN
$\varphi(.)$	activation function in a NN

b_k	bias in a NN
Z_s	self impedance
Z_m	mutual impedance

CHAPTER ONE
INTRODUCTION

1.1 Introduction

The recent restructuring and deregulation in the electric power sector over the last decade has brought about the need for efficient generation and transfer (transmission and distribution) of electric power to load centres. The mode of power transfer is usually via overhead lines. However, overhead lines (including underground cables) are subject to the forces of nature and other uncontrollable factors, thus liable to faults.

A very important component of power system design is the provision of adequate protection to detect and isolate faulty elements in the power system (NPAG, 2011). After the isolation of the faulty segment, it is important to investigate the root cause of the fault. To do this, diagnostic methods different from the protection device would need to be used.

Many diagnostic methods have been developed and proposed, but a perfect, dependable, and secure method is still the objective of continuous research. The basic objective of fault analyses and diagnosis involves the verification of a fault event, fault type classification, the affected segment, and the location of the fault.

Various methods in existence can be used for this. However, all these methods have one disadvantage or the other. Thus, it is best to make use of hybrid or composite methods in which two or more methods complement one another.

A diagnostic tool of this nature provides support to engineers and control center operators by aiding them in analyzing disturbance records retrieved from fault recorders. In recent times with the proliferation of power system monitoring equipment, the processing of data being recorded by these devices is herculean, and would take hours if not days to analyze. The results of analysis that takes hours or days are often not helpful to control centre operators who need to take informed decisions as soon as possible in fault situations.

This Chapter identifies the existing methods in distribution fault detection and diagnosis, the short coming of these methods, and what is being proposed as a solution to the problem.

The proceeding sub-sections discuss the awareness of the problem, research questions, research aim and objectives, research methodology, contribution of the research, etc.

1.2 Awareness of the Problem

Distribution networks have hitherto relied on a manual process for Fault Detection and Diagnosis (FDD). Most of these methods rely on measurements from voltage and current transformers, and short circuit current analysis using nomographs.

Nomographs are charts used to map the short circuit current magnitude to the likely fault location based on previous short circuit studies conducted. Other methods based on impedance calculation were also explored. However, certain issues like the existence of laterals/tap-offs, non homogeneous conductor size, inaccuracies in the configuration of the feeder, unbalanced phases, varying fault resistance, load uncertainly, etc., makes fault diagnosis in distribution networks peculiar and very challenging (Saha *et al.,* 2010; Mooney, 2012). After the detection of the fault and the subsequent tripping of the affected circuit, further diagnosis of the faulty section was usually carried out through reports from customers. In order to isolate the faulty segment, the line is energized section by section until the protection relay trips the circuit breaker. With this, the faulty section is identified. This procedure may be repeated several times which is time consuming and also exposes the equipment to additional stress from faults. Some utilities send linesmen to perform visual inspection of the lines. However, the damage caused by transients fault are usually minor and cannot be seen easily. Fault statistics on the South African Eskom Transmission network show that most faults are transients in nature and that single line-to-ground faults account for over 90% of such faults (Bekker & Keller, 2008). The use of oscillographs is equally time consuming and could be inaccurate. Another method in use is the Distribution Management System (DMS) (Saha *et al.,* 2010). Although, it provides an excellent platform for additional processing with the option of including terrain and weather conditions, it lacks accurate network modelling and requires protection relay settings, channel assignments, binary information from protection devices, etc. (Järventausta *et al.,* (1994); Kezunovic *et al.,* (2011).

Amorim *et al.,* (2004) reports that existing fault location methods adopt unrealistic simplifications and numeric relays can have up to 10% intrinsic error. The use of the fault locator function in digital protection relays is prone to error because such algorithms are meant for three phase homogeneous lines with constant Reactance/Resistance (X/R) ratio and not for distribution lines which are usually unbalanced and non-homogeneous (Gong & Guzman, 2011).

Therefore, an intelligent diagnostic scheme capable of extracting information from transient signals is thereby needed. Some of the advantages of such intelligent FDD scheme include the hastening of restoration times, improvement of system stability, system observability, reduced operating costs due to callouts and linesmen patrol, and expedited post-fault analysis.

The design and development of a FDD scheme require a lot of data for the implementation and validation of the algorithms. As it is often difficult to obtain data from utilities, the only recourse left is to use simulation tools to model near-real-life scenarios and generate such data.

1.3 Research Questions

The existing methods of FDD for distribution networks are not fully capable of accurate detection and diagnosis of the various types of distribution network faults, location of the faulted section, simultaneous operation in balanced/unbalanced networks, application in noisy environments, in overhead lines/underground cable networks, etc.

Furthermore, the trouble-shooting methods used are manual in nature and time consuming. Thus, there is the need to automate the process.

In view of the above, how can a method be developed to accurately and efficiently automate the process of fault detection and diagnosis of various low impedance faults wherever they may exist, classify the fault type, determine the faulted section of the distribution network and give the location of the fault in kilometres (km)?
In addition, how can the developed method be validated? Finally, can this method be commissioned to perform near real-time detection and diagnosis?

1.4 Research Aim and Objectives

1.4.1 Aim

To investigate the various methodologies for fault detection and diagnosis in electric power distribution system and to simulate and develop a method for the accurate detection and diagnosis of faults in distribution networks under various network operating conditions. This would entail the development of a hybrid algorithm based on Discrete Wavelet Transform (DWT), decision-taking rule-based algorithms, Artificial Neural Networks (ANNs), and the implementation and testing of the developed algorithm by using the Real-Time Digital Simulator (RTDS) in a closed-loop configuration with an external Intelligent Electronic Device (IED).

1.4.2 Objectives

i. To investigate existing fault detection and diagnosis methodologies.

ii. Modelling and simulation of a standard IEEE distribution network in DIgSILENT PowerFactory (DPF) for the purpose of generating data for steady state and dynamic conditions during faults.

iii. Investigate and select the best mother wavelet which best depicts the fault signature or characteristics for the DWT decomposition.

iv. Investigate the level of fault diagnostic signal decomposition, which would yield good characteristic patterns for the proposed Hybrid Fault Detection and Diagnosis (HFDD) method.

v. Develop decision-taking rule-based algorithms for the fault detection and classification tasks of the HFDD method.

vi. Optimization of the Neural Network architecture vis-à-vis the number of hidden layers, activation functions, and the number of neurons for the fault section identification and fault location tasks respectively.

vii. Design and validate a HFDD method in MATLAB using wavelet energy spectrum entropy in combination with decision-taking rule-based algorithms and neural networks.

viii. Testing and verification of the developed HFDD method on real disturbance signals extracted from RSCAD simulation runtime and also from events records retrieved from an Intelligent Electronic Device (IED) connected in a closed-loop with the RTDS during the real-time simulation studies.

1.5 Hypothesis

The limitations in existing power distribution system fault detection and diagnostic methods can be improved upon by developing a method based on the application of computational intelligence methods capable of operating in various network conditions, providing accurate detection of various fault types, classification of these faults into fault type and faulted phase(s), the line section affected, and location of the exact point where such faults exist. By implementing a hybrid method, the accuracy of the HFDD scheme would be improved since the advantages of the individual methods involved in the hybrid scheme would be enhanced in the process.

1.6 Delimitation of Research

1.6.1 Within Scope

i. Modelling of the IEEE 34-node test feeder is used for data generation.

ii. Simulation is done in DPF software environment.

iii. Simulation is carried out for steady state operating conditions and various fault scenarios.

iv. Fault types that are studied are Low Impedance Faults (LIF) in unbalanced systems, noisy conditions, and in overhead/underground lines.

v. A hybrid method for fault detection and diagnosis based on DWT, decision-taking rules, and ANN methodologies is developed in MATLAB.

vi. The validation and scalability of the developed method is demonstrated through the simulation of another IEEE benchmark model in the RTDS environment in a closed-loop with an external IED.

1.6.2 Outside Scope

i. Developing an interface between DPF and MATLAB and between the RTDS and MATLAB is software intensive and is outside the scope of this research work.

ii. Implementation of the developed algorithms in some hardware environment.

iii. Field deployment of the developed algorithms.

iv. This research covers faults on distribution lines only. Thus, fault detection and diagnosis on network equipment (generators, motors, transformers, etc) was not covered.

1.7 Motivation for the Research Project

The major factors to be considered in the selection of a power system fault detection and diagnosis scheme is system reliability and versatility. Literature review has shown that most of the existing schemes have one limitation or another and thus, unable to detect faults under all possible conditions (Thomas *et al.,* 2003; Lee *et al.,* 2004; Sengal *et al.,* 2005; Thukaram *et al.,* 2005; Perera *et al.,* 2006; Borghetti *et al.,* 2006; Filomena *et al.,* 2009; Campoccia *et al.,* 2010; Kumar *et al.,* 2011). This research project was motivated by the need to have in place, an effective fault detection and diagnostic system capable of all-round applications in balanced/unbalanced systems, operating in noisy conditions, and in overhead line/underground cable network. Furthermore, a fault detection and diagnosis scheme would aid the analysis of protection relay operations by providing a platform to distinguish between fault and ‘no fault’ transients, identifying the fault types, fault section, and fault location.

1.8. Assumption

The following assumptions are taken in the development of the fault detection and diagnosis algorithm:

i. Non-existence of ‘bad data’ in the results obtained from simulations.

ii. Evolving and cross-country faults are extreme cases and were not considered in the thesis.

iii. Appropriate thresholds for the fault detection and fault classification algorithms are obtainable empirically from the analysis of the entropy and entropy per unit values of the DWT decomposition.

iv. Neural network size is used as the first selection criterion based on the premise that the training time or epochs of a small sized NN can be increased in order to achieve comparable performance with larger networks.

v. A balance between the NN size, training time, and ability to generalize to untrained dataset would be achievable.

1.9 Research Methodology

The implementation of this research is multi-disciplinary and would involve a thorough review of information on methodologies in power system fault detection and diagnosis, power system modeling, simulation, signal processing, and software development. A hybrid combination of DWT, decision-taking rules, and ANN methodologies are used for this research. This is because a hybrid method would increase the algorithm's performance, robustness, and reliability.

The research methodology consists of literature review, analysis of the existing methods, design of a hybrid method, modelling and simulation for data generation, software development, experimentation, real-time verification/analysis of the results, and documentation.

1.9.1 Modelling

The IEEE 34-node benchmark test feeder was modelled in DIgSILENT PowerFactory. This model had all the components that could be found in distribution networks. The accuracy of the model in DPF is validated to be accurate through comparison of the load flow results with the benchmark load flow results published by IEEE Distribution System Analysis Subcommittee.

1.9.2 Simulation

Simulations of several operating scenarios including worse case scenarios are carried out. Extensive fault simulations are carried out at different line locations, with different fault resistance, fault inception angle, load angle, integration of DGs, etc. Short circuit analysis is also done for the IEEE 34 Node Test Feeder in DIgSILENT PowerFactory.

1.9.3 Design and Development of the Hybrid Method

The best DWT level to use for the signal decomposition is determined by experimenting on the various levels of the Daubechies DWT family. Wavelet features are extracted at different levels and their relative effectiveness is compared. Levels-1 and -5 were found to be the best levels that showed the required characteristics for FDD.

Decision-taking rules are also developed to take decision for the fault detection and classification tasks respectively. The output of the fault classification algorithm serves as the trigger for the fault section identification and fault location tasks respectively.

The algorithms for the fault section and fault location tasks are designed using the neural network toolbox in MATLAB. The ANN learning algorithms, number of hidden layer(s), number of neurons in the hidden layers, etc., are determined empirically through extensive simulations.

1.9.4 Software Implementation

Three phase and zero sequence waveform results from the DIgSILENT simulations are exported to MATLAB for DWT decomposition using codes developed in MATLAB. The coefficients resulting from the DWT decomposition serves as input to the decision-taking rules, and also to the ANN.

1.9.5 Hardware-in-Loop Simulation

The developed algorithm is tested and implemented in a 'hardware-in-the-loop' configuration using an external SEL-451 IED with the RTDS. The test feeder used was the IEEE 13-node benchmark model. The modelling and simulation is done using RSCAD software. COMTRADE files of fault events within the RSCAD runtime environment and from the external IED are used to test the developed HFDD method.

1.10 Major Contribution of the Thesis

The major contributions of this thesis include:

- The development of a wavelet energy spectrum entropy per unit formulation as the criteria for fault classification, fault section identification, and fault location.
- The development of a Hybrid Fault Detection and Diagnosis (HFDD) method using wavelet energy spectrum entropy.
- The development of decision-taking rules and neural networks for various FDD tasks.

- The development of an integrated method for fault detection, fault classification, fault section identification, and fault location in electric power distribution networks.
- Investigation of the effects of system changes and disturbance like the integration of Distributed Generation (DGs) and line extension to the network.
- Modelling and simulation of distribution network operating conditions using the Real-Time Digital Simulator (RTDS) in a closed-loop with SEL-451 Intelligent Electronic Device (IED).
- Testing the algorithms with disturbance records retrieved from the IED and the RTDS runtime waveform plots.

1.11 Outline of the Thesis

This thesis is made up of eight chapters. The chapters provide answers to the questions on what the problem is, why it is a problem, the proposed solution to the problem, and how the solution is being implemented.

Chapter Two gives an extensive review of the methodologies used in fault detection and diagnosis. The various algorithms used in FDD and a summary of the literature reviewed are highlighted therein.

Chapter Three introduces the theoretical aspects behind power system signal processing and ANNs. Discrete Fourier Transform (DFT), Short-Time Fourier Transform (STFT), and Wavelet Transform (WT) are discussed. ANN architectures that are usually employed are also presented. The various learning strategies used in the training process of neural networks along with the critical factors that affect the size and output of a trained network are also discussed in this chapter.

Chapter Four presents the modelling and simulation of the IEEE 34 node test feeder in DIgSILENT PowerFactory. Details of the various components used in modelling the test feeder are presented and the load flow result comparison is done with the benchmark IEEE distribution sub-committee result in order to validate the simulation result obtained. Several fault scenarios are carried out at different fault locations with different fault types, fault inception angles, fault resistances, and loading angles. The aim of the extensive simulation is to cover all possible scenarios that are likely to occur in a typical power system network.

Chapter Five deals with the actual implementation and development of the DWT decomposition, decision-taking rule-based algorithms for fault detection and classification, and the design of the neural networks for fault section identification and fault location. An overview of the training and testing of the neural networks used in this work are outlined in this chapter.

Chapter Six presents series of simulation results that have been obtained using MATLAB’s Wavelet Toolbox and Neural Network Toolbox in detail. This is to emphasize the efficiency and accuracy factors of the proposed HFDD method. Several neural networks with varying configurations have been trained and tested. Their performances are analyzed in this chapter.

Chapter Seven presents a real-time simulation using the RTDS in a hardware-in-the-loop configuration with an external IED. This is meant to act as a proof of concept, and is implemented on a different benchmark test feeder.

Conclusions drawn from the research, deliverables, and the direction for future work are given in Chapter 8. References used and the appendices are presented immediately after Chapter 8.

1.12 Conclusion

This chapter discussed the awareness of the fault detection and diagnosis problem in distribution networks, the research questions resulting from the awareness of this problem, research aim and objectives of the thesis, research methodology, contribution of the research, etc.

Chapter Two presents an extensive literature review conducted on the various methodologies used in Fault Detection and Diagnosis (FDD). The methodologies covered are the impedance/fundamental frequency methods, high frequency/travelling wave methods, and the knowledge-based methods.

CHAPTER TWO

METHODOLOGIES FOR FAULT DETECTION AND DIAGNOSIS IN DISTRIBUTION POWER NETWORKS

2.1 Introduction

Fault detection and diagnosis is a central component of Abnormal Event Management (AEM) in electric power systems. The constituents of which include fault event detection, fault type classification, fault section identification, and fault location.

In the past, most of the research and development in power system fault detection and diagnosis has focused on power transmission systems, and it is not until recently with deregulation, competition, and enforcement of strict reliability indices by regulatory bodies, that research on power system faults has begun to investigate the unique aspects of distribution systems (Das *et al.,* 2003; Herraiz *et al.,* 2007).

Transmission and distribution lines are subject to faults caused by lightning, storm, snow, insulation breakdown, birds, etc. The early detection and diagnosis of these faults can expedite the restoration and return to service of these lines. The application of algorithms designed for transmission networks when used for distribution lines are subject to errors because distribution lines are not of the same type or size, the existence of tap-offs/sub-laterals, single source operation, load taps, etc. (Das *et al.,* 2003; Mora *et al.,* 2006; Mora-Fl'orez *et al.,* 2008).

When fault occurs in an electric power system, the effects are often not restricted to the faulty section, but noticeable throughout the whole system. Thus, the restoration after fault occurrence must be expedited by putting in place a scheme capable of fault detection and diagnosis; thus maintaining system stability, minimizing customer and network damages, as well as economic losses.

At present, fault detection and diagnosis in power systems are based on manual processes. For instance, fault location is based on customer trouble calls, visual inspections of lines, fault current indicators, etc. The use of the fault locator function in digital protection relays is prone to error because such algorithms are meant for three phase homogeneous lines with constant X/R ratio and not for distribution lines which are usually unbalanced and non-homogeneous. The X/R ratio varies for distribution lines; such that if a constant X/R ratio is used, the measured fault location would be different from the actual fault location. The use of nomographs to compensate for this error is a cumbersome and time consuming task for utilities (Gong & Guzman, 2011).

Another method used by utilities is the analysis of the short circuit results. This method uses the measured fault current to identify the possible fault locations in a feeder. However, it is necessary to make use of the system voltage during the fault and also assume that the effects of the fault resistance are negligible.

Existing methods for transmission and distribution lines fault detection and location are generally classified into impedance/other fundamental frequency methodologies, high frequency components/travelling wave methodologies, and knowledge based methodologies (Tang, *et al.,* 2000; Saha *et al.,* 2010; Adewole & Tzoneva, 2011).

A fault detection and diagnosis scheme is a process aimed at verifying if a fault event did occur, analysis of certain parameters to determine the type of the fault, phase(s) affected by the fault, the faulted line section, and also estimate the distance to the fault with the highest accuracy possible. Generally, electric power fault diagnosis comprises of the following phases:

- Detection of fault event
- Fault type/phase(s) classification
- Fault section determination
- Fault location in kilometre

The literature review gives an overview of the existing methods developed for fault detection and diagnosis in distribution networks. This chapter provides a review of electric power system protection, faults in power systems, protection philosophy, protection equipment, and reliability indices. Furthermore, the various methodologies used in fault detection and diagnosis are discussed herein.

2.2 Electric Power System Protection

An electric power system is an assemblage of equipment that facilitates the generation, transmission, and distribution of electric energy. The purpose of electric power systems is to provide energy for human use in a secure, reliable and economic manner. The importance of electric energy and investments in the form of facilities and equipment, make the normal and continuous operation of power systems significant and strategic for every society.

Electric power is produced by generators located at power plants. These generators convert a primary source of energy into electrical power. Typically, the voltage at generation ranges from 10kV to 25kV. This voltage is then stepped up to higher voltage levels ranging from 230kV to 765kV by means of step-up transformers.

Transmission is carried out at these high voltages. At the High Voltage/Medium Voltage (HV/MV) substations, voltages are stepped down to lower levels of 66kV, 33kV, 11kV and 0.415kV (Laithwaite and Freris, 1980; Theraja and Theraja, 2008).

Electric energy is distributed to the load centres through primary and secondary distribution feeders depending upon the customer's energy requirement. Industrial customers that demand large amount of electric power are fed from the primary distribution feeders while other domestic users are connected to secondary feeders.

Figure 2.1 shows the interrelation of the various networks involved in the electric power system, from generation to the load centres. EHV is the extra high voltage resulting from the step-up transformers (primary transmission level), while HV signifies the high voltage at the secondary transmission level. MV and LV are the medium voltage and low voltage supply corresponding to the primary and secondary distribution levels respectively.

Protection in power system is concerned with the protection of power systems elements from faults through the timely separation of the faulty elements from the rest of the healthy network and restoration back to normal condition.

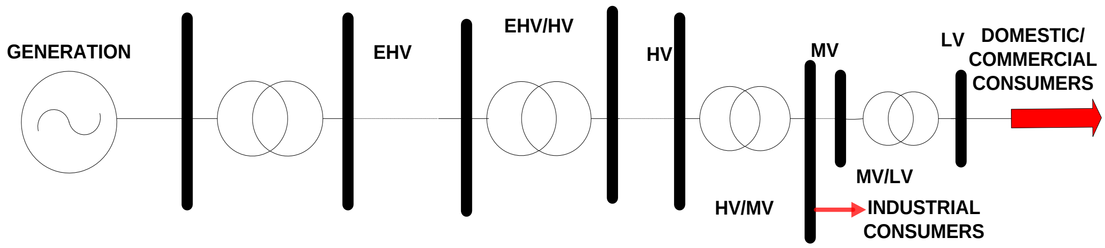


Figure 2.1: Block diagram of a typical transmission and distribution system

Protection systems are auxiliary equipment put in place to provide safety, reliability, and quality of service and are installed on all circuits and electrical equipment. The objective of protection in power system is mainly to achieve stability of the system. This it does by separating only the faulty elements. The major protection system components are: current and voltage transformers, protection relays, circuit breakers, power sources (batteries), and of course, a communication channel for the transmission of signals for the remote operation of the protection system (Horowitz & Phadke, 1992; Hewitson *et al.,* 2005).

The rapid growth of electric power systems over the past few decades has resulted in a large increase in the number of transmission and distribution lines in operation and thus, their overall length. These lines are usually overhead lines and are subject to the forces of nature. Thus, faults are inevitable and must be cleared before putting the line back into operation (Gers & Holmes, 1998; Das *et al.,* 2003; NPAG, 2011).

2.2.1 Types of Fault

An electrical power system fault is the unintentional and undesirable creation of a conducting path or a blockage of electric current (Whitaker, 2007).

Distribution feeder faults can be sub-divided into two major categories:

- High impedance faults
- Low impedance faults

High Impedance Faults (HIFs) can be defined as electrical contacts between a bare current-carrying conductor and an insulated foreign object. This is usually as a result of a current carrying conductor touching a high impedance surface after breaking. Such surfaces can be the road, sand, grass, etc, and it is a threat to human life and the environment. Another variant is when the current carrying conductor does not break, but comes in contact with insulated grounded objects (Aucoin, 1985).

Low Impedance faults (LIFs) include conventional shunt faults like:

1. Single line-to-ground fault
2. Line-to-line fault
3. Double line-to-ground fault
4. Balance 3 phase-to-ground fault

With a single line-to-ground fault only one phase has non-zero fault current. This is as illustrated in the figure below.

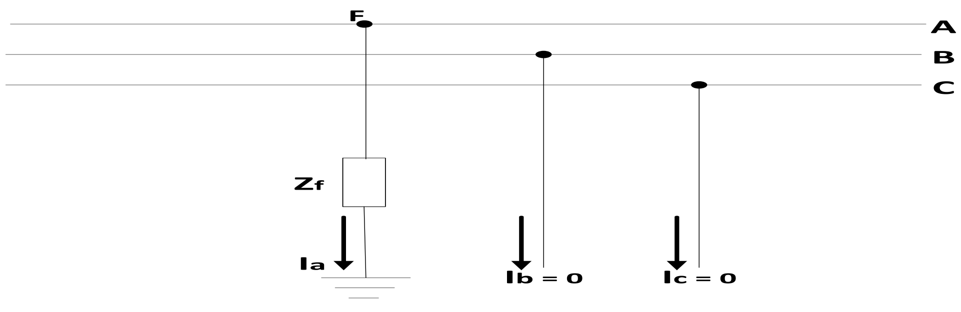


Figure 2.2: **Illustration of a single line-to-ground fault**

Double line-to-ground fault occurs when two (2) phases are short-circuited as shown in Figure 2.3. That is, two (2) line conductors come in contact both with each other and ground.

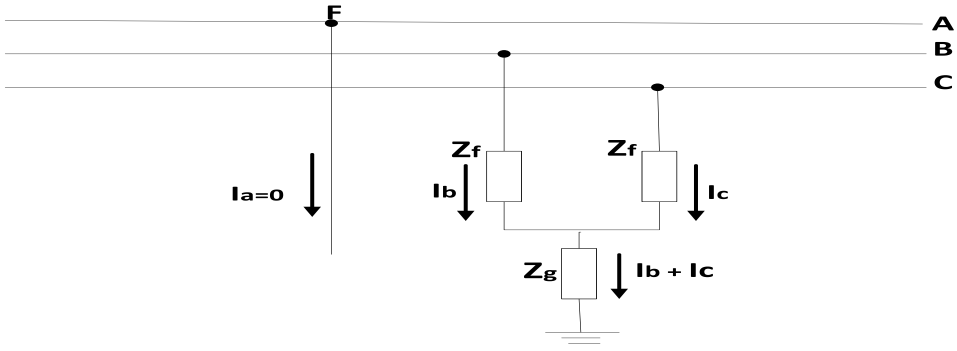


Figure 2.3: **Illustration of a double line-to-ground fault**

Line-line fault occurs when two (2) phases are short-circuited as shown in Figure 2.4. Line-to-line faults occur when two (2) of the conductors come in contact with each other. For example, phases B and C as shown in Figure 2.4 below.

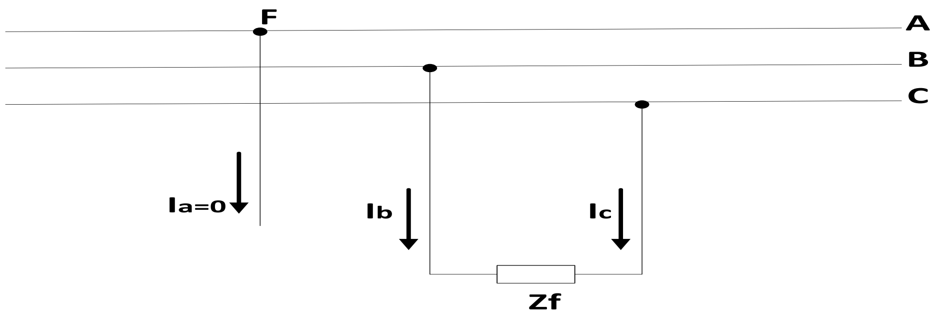


Figure 2.4: **Illustration of a line-to-line fault**

Three phase-to-ground faults are the least probable faults but yet the most severe (Makram *et al.,* 1987). This is as shown in Figure 2.5.

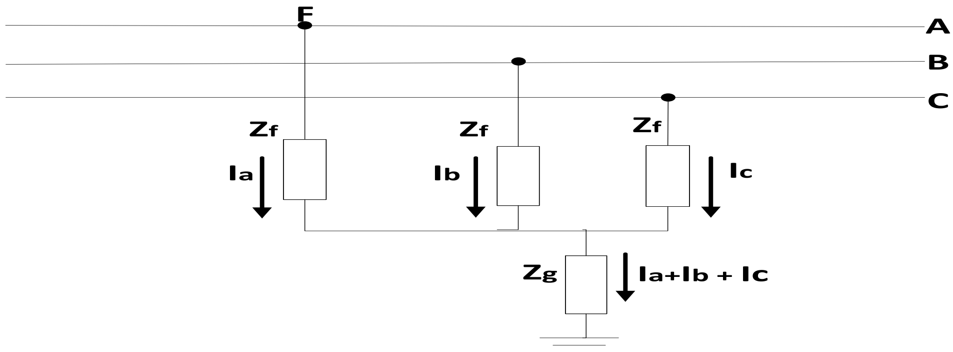


Figure 2.5: **Illustration of a three phase-to-ground fault**

Figures 2.2-2.5 show some common types of faults at location '*F*' in a given power system. According to the three component method (Anderson, 1985) the positive, (I_{a1}), negative (I_{a2}), and zero sequence (I_{a0}) fault currents of phase A at fault location '*F*' can be calculated. If a fault occurs at location '*F*', fault current will flow through any feeder of the whole system and if the positive, negative and zero sequence fault currents of phase A at a particular feeder *i* are I_{ia1}, I_{ia2}, and I_{ia0}. Equation (2.1) can be used to calculate these currents based on positive, negative, and zero sequence networks (Chan & Lu, 2001).

$$\left.\begin{array}{l} I_{ia1} = K_{i1} I_{a1} \\ I_{ia2} = K_{i2} I_{a2} \\ I_{ia0} = K_{i0} I_{a0} \end{array}\right\} \tag{2.1}$$

In the above equation, K_{i1}, K_{i2}, K_{i0} are positive, negative and zero sequence feeder coefficients of feeder *i*, respectively with regard to fault location '*F*'. If the fault current at a specified location in a network is so great with respect to the load currents, the load currents can be ignored. Thus, for a particular feeder related to a particular fault location in the system, these three feeder coefficients are constants and can be calculated on the basis of three sequence networks and circuit theory. Feeder coefficient differs from feeders and locations and for a particular feeder *i*, the feeder coefficient may vary for different fault location. For a particular fault location, different feeders may have different feeder coefficients.

From equation (2.1), if $K_{i1} = K_{i2} = K_{i0} = K_i$, it can be written as:

$$\left.\begin{array}{l} I_{ia1} = K_i I_{a1} \\ I_{ia2} = K_i I_{a2} \\ I_{ia0} = K_i I_{a0} \end{array}\right\} \tag{2.2}$$

where k_i is called general feeder coefficient of feeder *i* with regard to fault location *'F'*. Therefore, the three phase fault currents at feeder *i* can be derived from Equation (2.5) (Chan & Lu, 2001).

$$\begin{bmatrix} I_{ia} \\ I_{ib} \\ I_{ic} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \begin{bmatrix} I_{ia0} \\ I_{ia1} \\ I_{ia2} \end{bmatrix} \tag{2.3}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \begin{bmatrix} K_i I_{ia0} \\ K_i I_{ia1} \\ K_i I_{ia2} \end{bmatrix} \tag{2.4}$$

$$= K_i \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \begin{bmatrix} I_{ia0} \\ I_{ia1} \\ I_{ia2} \end{bmatrix} = K_i \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{2.5}$$

where *a* is an operator that gives a phase shift of 120^o clockwise and a^2 gives a phase shift of 240^o.

Therefore,

$$\begin{bmatrix} |I_{ia}| \\ |I_{ib}| \\ |I_{ic}| \end{bmatrix} = |K_i| \begin{bmatrix} |I_a| \\ |I_b| \\ |I_c| \end{bmatrix} \tag{2.6}$$

Equation (2.6) confirms that the fault current magnitude at feeder *i* is $|K_i|$ times greater at a particular fault location. In this equation, for a particular fault location '*F*' and a particular feeder *i*, $|K_i|$ is a constant. When fault occurs, the feeder coefficients of all the feeders in the system determine the fault current distribution in in the faulted system.

2.2.2 Protection Philosophy

When fault occurs in a power system, the protective IEDs detect the fault and trip appropriate circuit breakers which isolate the affected equipment from the rest of the power system. Protection philosophy ensures that in the event of a fault, the faulted element must be disconnected from the system in order to prevent further damage to the components of the system through which the fault currents are flowing. To achieve this objective, the system is divided into separate zones. The aim of these zones is to ensure that the system elements are covered by at least one of the zones.

Also, these zones of protection must overlap in order to ensure that no system element is left unprotected. The consequence of this practice is that a fault in an overlapping zone trips the circuit breakers to isolate more than one element from the system.

The overlap as shown in Figure 2.6 is carried out by connecting the protection relays to the appropriate current transformers (Horowitz & Phadke, 1992; Gers & Holmes, 1998).

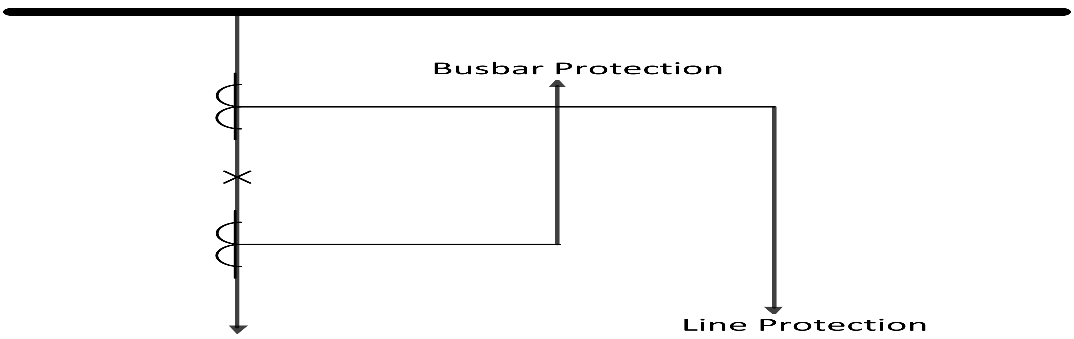


Figure 2.6: **Protection zones overlap implementation**

Figure 2.7 further illustrates the protection zones of a simple power system. In order to ensure that every element within a given power system is adequately protected; back-up protection is used to compliment the primary protection.

The primary protection is the main protection for a particular zone. Back-up protection is the practice of configuring certain relays to detect both faults within a particular zone and also in adjacent zones. Usually, this is used to back up the primary protection as a second line of defence in the event that the main protection fails to operate. The back-up protection relay includes a time-delay facility to slow down the operation of the relay thereby allowing the primary protection to operate first.

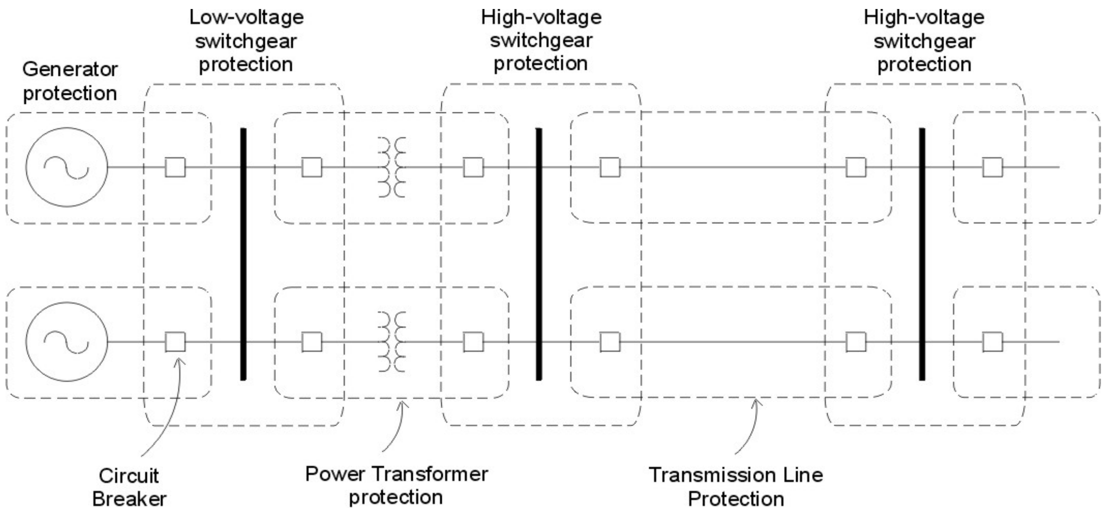


Figure 2.7: **Relay protection zones in a power system**

2.2.3 Types of Protection Equipment

2.2.3.1 Fuses

A fuse is an overcurrent device used in the protection of a power system network. Under normal operating conditions, the heat built up in the fuse element is dissipated to the surrounding air and thus, the fuse remains at a temperature below its melting point. During fault conditions such as a short circuit, the heats become very great and cannot be dissipated fast enough. This causes the fuse element to heat up and melt, thereby breaking the circuit.

Fuses based on the expulsion principle find more application in distribution system (Gers & Holmes, 1998). When faults occur, the interior fibre material of the expulsion-type fuse is heated up as the fuse element melts and produces gases. The accumulated gases are compressed and expelled out of the tube in order to eliminate the established arc during the destruction of the fuse element.

2.2.3.2 Instrument Transformers

Instrument transformers are transducers used to transform high electric current and voltages to lower values proportional to the primary magnitudes thereby providing isolation between the electric power circuit and the measuring instruments. These transducers {current transformers (CTs) and voltage transformers (VTs)} measure the current and voltage in a network and provide low level signals to relays in order to detect abnormal conditions (Gupta, 2004).

CTs are used as inputs to the current coils of metering instruments and relays and help to isolate these instruments from high voltage. CTs have their primary and secondary windings wound on iron cores. The primary winding of the CT is connected in series with the load and carries the actual magnitude of the power system current. The secondary winding is connected to the measuring instruments or relay.

Similarly, VTs are used to feed the potential coils of metering instruments and relays and they make the low voltage instruments compatible for high voltage measurements. The primary winding is connected to the power circuit (usually between phases or between phase and ground), while the secondary winding is connected to various metering instruments or relays.

In recent times, Non-Conventional Instrument Transformers (NCITs) are beginning to find applications in power system networks. NCITs make use of optical technology without multiple CT cores; they have small dimensions, and are very accurate, safe, and have low weight (NPAG, 2011). NCITs are key components in the implementation of IEC 61850-9.2 standard-based digital substation.

2.2.3.3 Relays

A protective relay is a device capable of detecting changes in the received signal and if the magnitude of the received signal is outside a preset range, it operates to initiate appropriate control action in order to protect the power system. Protective relays have transformed over the years as power systems increased in size and complexity (Gers & Holmes, 1998).

Protection relays can be classified based on their characteristics, as follows:

- General function: Auxiliary, protection, monitoring, or control.
- Construction: Electromagnetic, solid state, microprocessor, etc.
- Input signal: Current, voltage, power, frequency, temperature, pressure, etc.
- Type of protection: Overcurrent, directional overcurrent, distance, overvoltage, differential, etc.

Early protective relays were constructed using solenoids and electromagnetic actuators. In electromechanical relays, the actuating force is created by electromagnetic interaction due to input electrical parameters. These relays contain mechanical parts such as spring, dashpot, induction disc, etc. If the torque generated by the input overcomes the torque generated by the pre-stored energy in the spring, the moving part of the relay will act and generate the output signal (Horowitz & Phadke, 1992). Electromechanical relays are in general reliable and have low requirements on operational environment. However, these relays have large sizes, are slow to act, have high power consumption for auxiliary mechanisms, and are unable to implement complex characteristics (Horowitz & Phadke, 1992; Gupta, 2004).

The introduction of integrated circuits and static circuits brought about the development of solid-state relays. Solid-state relays replaced electromechanical actuators with analogue electronic elements. They have the advantages of lower costs, reduced space and weight, ease of use, speed, accuracy, and can realize more complex functions. They also consume less power and require little maintenance. In practical applications, solid-state relays are mainly applied in the protection of transmission systems (Gers & Holmes, 1998; Hewitson *et al.,* 2005).

However, the overall reliability of solid-state relay can be affected by individual components since it consists of many electronic components which are sensitive to changes in the ambient temperature and voltage transients, and may cause mal-operation (Horowitz & Phadke, 1992).

The developments in digital technology led to the incorporation of microprocessors in the construction of relays. Digital and numerical relays are sophisticated, multi-functional equipment with the capacity to record signals during faults, perform self-monitoring, and communicate with their peers. Numerical relays employ microprocessors specially designed to process digital signals, which make them faster and more powerful. The main advantages of digital relays over conventional relays are their reliability, functional flexibility, self-monitoring, and self-adaptability (Aggarwal & Johns, 1997). Digital relays are able to implement more complex functions, increased accuracy, stability, and are immune to effects of the physical environment.

To safeguard the investment in transmission and distribution lines, several types of protection techniques are used. Distance, over-current, differential, directional, etc are some of these techniques. A single technique or combinations of two or more techniques are employed to detect faults on transmission and distribution lines.

Distance relays use components of voltages and currents acquired at the relay location to calculate the apparent impedance of the protected line. The calculated apparent impedance is compared with predetermined impedance which is called the reach of the relay. During normal operation, the apparent impedance must be larger than the impedance-reach of the relay. If the apparent impedance is less than the impedance-reach, then it is inferred that a fault has occurred in the protected line between the relay location and the impedance reach of the relay. Under these conditions, the distance relay sends a trip signal to the appropriate circuit breakers thereby isolating the faulted line from the rest of the system.

Directional relay is used when it is necessary to protect the system against fault current which could spread in both directions through a system element. The operation is based on the comparison of the phase angle between two AC inputs. This comparison can be between the current phasor and voltage phasor, and also between two current phasors. With these, it is possible to determine whether a fault is in the forward or in the reverse direction from the relay depending on the value of the estimated phase angle compared to the expected phase angle (Horowitz & Phadke, 1992).

Differential relays on the other hand are based on the algebraic sum of two or more inputs. In a general form, those inputs may be the currents entering (or leaving) a specific protection zone. Thus, differential relays operate when the vector difference of two or more similar electrical magnitude exceeds a predetermined value and it is usually used for unit protection. The principle of differential relaying is usually a unit protection. It is a common protection principle for transformers, motors, and generators.

2.2.3.4 Circuit Breakers

Circuit-breakers are used at all voltage levels in power system networks and their function is to protect a circuit from the harmful effects of a fault. Circuit breakers are designed to operate as quickly as possible, in most instances, in less than 10 cycles of the power frequency in order to limit the impact of a fault on the distribution system (Whitaker, 2007). The most common in medium voltage systems are either SF6 circuit-breakers or vacuum circuit-breakers. The basic principle of current interruption of an alternating current is based on the natural zero-crossings of the current twice per period of the power frequency.

2.2.3.5 Reclosers

In electric power distribution, an autorecloser is an overcurrent device capable of automatically closing a circuit breaker after it must have opened because of fault. This was developed because most faults in the utility systems are temporary in nature and can be cleared by de-energizing the circuit for a short period of time (Whitaker, 2007). Autoreclosers are used in coordinated protection schemes for overhead line power distribution circuits with a high probability of being affected by transient faults. An autorecloser is designed to make several pre-programmed attempts to reenergize the line. If the transient fault has been cleared, the autorecloser's circuit breaker will remain closed and normal operation of the power line will resume. If otherwise (e.g for a permanent fault), the autorecloser will exhaust its pre-programmed attempts (shots) to re-energize the line and remain tripped off until manual command for a retry is issued to it. About 80-90% of faults on overhead distribution lines are transient and can be corrected by autoreclosing (Gers & Holmes, 1998). Autoreclosers can be single-phase or three-phase versions, and can use oil, vacuum, or SF_6 interrupters. They are rated from 2.4kV-38kV, with load currents of 10A-1200A and fault currents from 1kA-16kA.

2.2.3.6 Sectionalisers

A sectionaliser is an overcurrent device installed in conjunction with back-up circuit breakers or reclosers and it is designed to automatically isolate faulted sections of a network once an upstream breaker or recloser has interrupted the fault current. Sectionalisers count the number of times the recloser tries to operate during fault conditions, and after a predetermined number of recloser openings, the sectionaliser opens and isolates the faulty section of the line. This enables the recloser to close and re-establish supply to the healthy sections of the network.

In the selection of a sectionaliser, the system voltage, maximum load current, maximum short circuit level, and the coordination with the installed upstream and downstream protective devices should be considered (Gers & Holmes, 1998).

2.2.4 Reliability Indices

Electric power interruptions are characterized by standard reliability indices. These indices measure the system performance and reflect the status of the overall performance/Quality of Service (QoS) of a particular network at the load points. The IEEE P1366 standard defines a momentary interruption as an event or outage less than five minutes. While a sustained interruption is an event lasting more than five minutes.

The most commonly used indices in utility companies are:

1. System Average Interruption Frequency Index (SAIFI): This is an important reliability index used by distribution utilities.

SAIFI is the mean of the number of interruptions a customer experiences and is measured in units of interruptions/customer over a year. It should be noted that SAIFI excludes momentary interruptions and it is calculated as:

$$SAIFI = \frac{Total\ Number\ Customer\ Interruptions}{Total\ Number\ of\ Customers\ Served} \tag{2.7}$$

2. System Average Interruption Duration Index (SAIDI): This gives the system average interruption duration per customer per year. This index is commonly referred to as customer minutes of interruption.

$$SAIDI = \frac{Sum\ of\ Customer\ Interruption\ Durations}{Total\ Number\ of\ Customers\ Served} \tag{2.8}$$

3. Customer Average Interruption Duration Index (CAIDI): CAIDI is the mean outage duration a particular customer would encounter. CAIDI is measured in units of time, often minutes or hours and like SAIFI, it is also measured over the course of a year. CAIDI is expressed by:

$$CAIDI = \frac{Sum\ of\ all\ Customer\ Interruption\ Durations}{Total\ Number\ of\ Customer\ Interrruptions} = \frac{SAIDI}{SAIFI} \tag{2.9}$$

4. Customer Average Interruption Frequency Index (CAIFI): CAIFI is defined as the frequency of sustained interruptions for customers experiencing sustained interruptions during the reporting period.

$$CAIFI = \frac{Total\ Number\ of\ Customer\ Interrruptions}{Total\ Number\ of\ Customers\ Interrrupted} \tag{2.10}$$

5. Customer Total Average Interruption Duration Index (CTAIDI): This defines the total average outage time in the reporting period that customers were without electric power. This is denoted by:

$$CTAIDI = \frac{Sum\ of\ Customer\ Interruption\ Durations}{Total\ Number\ of\ Customers\ Interrupted} \tag{2.11}$$

6. Average Service Availability Index (ASAI): This defines the ratio of the total number of customer hours that service was available during the year to the total customer hours demanded.

$$ASAI = \frac{Customer\ Hours\ Service\ Availability}{Customer\ Hours\ Service\ Demand} \tag{2.12}$$

7. Momentary Average Interruption Frequency Index (MAIFI): MAIFI is used to reflect how often served customers would experience a momentary interruption in a given year.

$$MAIFI = \frac{Total\ Number\ of\ Customers'\ Momentary\ Interruption\ per\ annum}{Total\ Number\ of\ Customers\ Served} \tag{2.13}$$

8. Momentary Average Interruption Frequency Index of events (MAIFIe): This is used to indicate how often served customers would experience a momentary event in a given year.

$$MAIFIe = \frac{Total\ Number\ \ of\ Customers'\ \ Momentary\ \ Interrupt\,ion\ \ events\ \ per\ annum}{Total\ Number\ \ of\ Customers\ \ Served} \qquad (2.14)$$

The occurrence of faults in a distribution network results in low reliability indexes. The availability of a fault detection and diagnostic scheme would greatly help in the reduction of the Mean Time To Repair (MTTR) and systems down-time by the swift identification of faults and location of the faulty node. The faulty segment(s) can then be isolated and power is restored to the healthy segments.

Table 2.1 shows Eskom's SAIDI and SAIFI figures for South Africa (Eskom integrated report, 2012). Similarly, Figure 2.8 illustrates the trend in Eskom's reliability indices for five years (Eskom integrated report, 2011). The challenge would be to further improve the QoS by achieving the target indices for coming years.

Eskom's intentions to reduce SAIDI to 39 hours by 2016/17 and reduce SAIFI to 17 interruptions per annum also by 2016/17 would only be achievable through effective outage management coordination (fault diagnostic schemes) and other corporate strategies.

Table 2.1: **Distribution technical performance for Eskom**

Measure	Description of measure /(and unit)	Target 2012	Actual 2012	Actual 2011	Actual 2010
System Average Interruption Duration Index (SAIDI)	Availability of supply index /(hours per annum)	≤49.0	45.75	52.61	54.41
System Average Interruption Frequency Index (SAIFI)	Reliability of supply index /(number per annum)	≤22.0	23.73	25.31	24.65

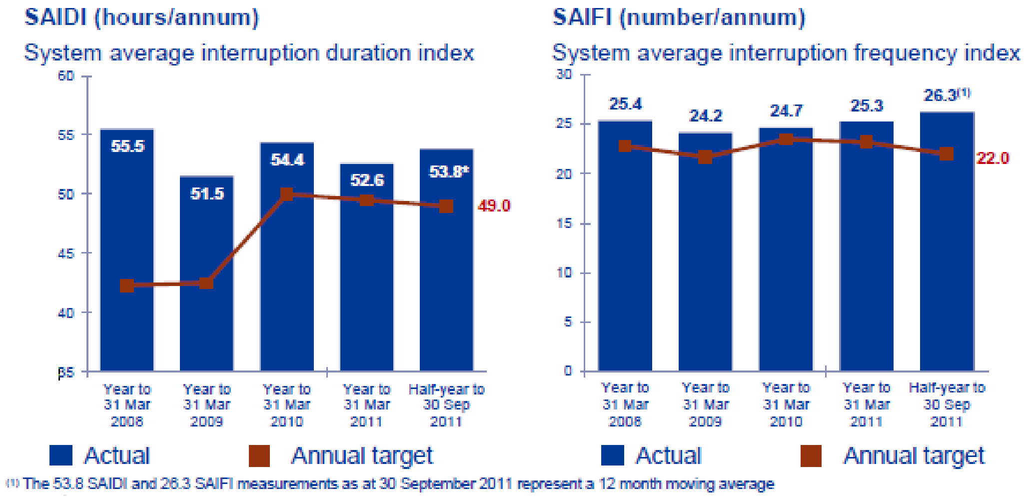


Figure 2.8: **A chart showing Eskom's SAIFI and SAIDI trends for five years (Eskom integrated report, 2011)**

2.3 Review of Methodologies for Fault Detection and Diagnosis in Distribution Networks

The first set of algorithms developed for Fault Detection and Diagnosis (FDD) in distribution networks were based on impedance calculations (Takagi *et al.,* 1982; Girgis *et al.,* 1993; Santoso *et al.,* 2000; Das *et al.,* 2000).

Voltage and current measurements were used to calculate the line characteristics by using various mathematical methods. However, this method is beset by the problem of multiple estimations in distribution networks because of situations where more than one lateral or section corresponds to the calculated distance.

With recent developments in technology and computing, methods based on travelling waves and computational intelligence began to find application in power systems. Research works by Thomas *et al.,* 2003; Ohrstrom *et al.,* 2005; Hizam and Crossley, 2007, etc. were based on high frequency components which are often difficult to capture except with state of the art technology that requires the use of travelling wave recorder, Global Positioning System (GPS), high speed communication channel, etc.

The use of computational intelligence was facilitated by developments in signal processing and computing. Thus, this particular method is the subject of recent research works. This is demonstrated in Martins *et al.,* 2003; Cormanne *et al.,* 2006; Mora-Florez *et al.,* 2009; Rayuda, 2010; Kumar *et al.,* 2011, etc.

Existing methods for transmission and distribution lines fault detection and location are generally classified into impedance/other fundamental frequency methodologies, high frequency components/travelling wave methodologies, and knowledge based methodologies (Tang *et al.,* 2000; Mora Florez *et al.,* 2008; Saha *et al.,* 2010, Adewole & Tzoneva, 2011).

2.3.1 Review of Impedance and Other Fundamental Frequency Methods: Impedance Based Method

The basic approach to fault detection and diagnosis using impedance and other fundamental frequency methodologies involves initiating the measurements of voltage and current at the start of the fault, or after the fault, at one end or both ends of the feeder. The measured quantities are afterwards used to compute the characteristics of the line by using mathematical equations.

Basically, the line impedance per unit length is used to determine the fault location. The simple reactance method makes use of equations (2.15)-(2.17) to calculate the fault distance.

The voltage drop from the sending end of the line is given by:

$$V_S = (x.Z_l \cdot I_S) + R_F \cdot I_F \tag{2.15}$$

$$\mathrm{Im}\left(\frac{V_S}{I_S}\right) = \mathrm{Im}(x \cdot Z_l) = x \cdot X_l \tag{2.16}$$

$$x = \frac{\mathrm{Im}\left(\frac{V_S}{I_S}\right)}{X_l} \tag{2.17}$$

where x is the distance to the fault point, V_s and I_s are the voltage and current of the sending terminal, I_s'' is the current difference between the pre-fault and post-fault (fault component current), and I_F is the fault current. X_l is the power line reactance per unit length, *Im* is the imaginary component and $*$ is the conjugate component.

An improvement over the reactance method was a proposal by Takagi *et al.,* (1982) for fault location. The Takagi method is a single-end approach and it utilizes both pre-fault and fault data. It is also an enhancement of the simple reactance method by reducing the effect of load flow and fault resistance (Zimmerman, 2005).

$$x = \frac{\mathrm{Im}\left(V_S \cdot I_S''^{*}\right)}{\mathrm{Im}\left(Z_l I_S \cdot I_S''^{*}\right)} \tag{2.18}$$

where x is the distance to the fault point, V_s and I_s are the voltage and current of the sending terminal respectively. I_s'' is the current difference between the pre-fault and post-fault (fault component current). Z_l is the power line impedance per unit length, *Im* is the imaginary component and $*$ is the conjugate component.

Another method is the modified Takagi method. This method makes use of zero sequence current. Thus, there is no need for pre-fault data.

$$x = \frac{Im\left(V_S \cdot (3 \cdot I_{0S})^{*} \cdot e^{-jT}\right)}{Im\left(\left(Z_l \cdot I_S \cdot I_S^{"*}\right) \cdot e^{-jT}\right)} \tag{2.19}$$

A review paper on the methodologies for fault detection and diagnosis in distribution networks (Adewole & Tzoneva, 2011) gave a comprehensive review of methods and techniques used in FDD. Therein, it was mentioned that Girgis, (1993) presented equations to calculate various types of fault occurring at the main feeder and at a single phase lateral line. He used Discrete Fourier Transform (DFT) to filter off harmonics using simple computation. Also, the problem of multiple fault locations corresponding to calculated values was solved by updating the current and voltage vectors using static impedance load model. Santoso *et al.,* (2000) suggested two equation methods for fault distance estimation. In the first method, two linear equations were formed from the apparent impedance equation, while the second method used the distance equation as a function of the fault resistance. The latter was shown to give better results than the former. However, the linear equation method produced more accurate result when the fault was without any resistance.

Senger *et al.,* (2005) proposed a method utilizing measurements from Intelligent Electronics Devices (IEDs). These devices are installed at the substation and it is implemented with a database comprising of the network topology and parameters. The fault types considered were Phase-A-to-Ground (A-G), Phase B-to-Phase C (B-C), Phase B-to-Phase C-to-Ground (BC-G), Three Phase (3Ph.), and Three Phase-to-Ground (3Ph.-G) faults. Saha *et al.,* (2005) used the recorded three-phase voltage measured at the supply busbar of a 16 feeder-Medium Voltage (MV) network, current from the faulted feeder, and network impedance for the symmetrical components to determine the fault location for Single Line-to-Ground (SL-G) and Three Phase (3-Ph.) faults. This was pre-calculated after each change in the network parameters and stored in the database.

Another technique for SL-G faults was also put forward by Pereira *et al.,* (2006). This technique made use of voltage and current phasors measured at the substation level, and voltage magnitudes measured at some buses of the feeder. It had a database that contained electrical, operational, and topology parameters of the distribution network. The proposed technique was tested on an actual 13.8 kV, 238 nodes overhead feeder. The technique can easily be implemented using Power Quality (PQ) measurement devices already installed on the feeders. With this, the PQ devices perform power quality analysis and fault location. Thus, helping electric utility companies to maximize their investments.

Pereira *et al.,* (2009) presented a fault location algorithm, which used voltage sag measurements from PQ meters and voltage measurement devices on the feeders. Data generated by Alternative Transient Program (ATP) simulations are used as input data to the algorithm. Morales-España *et al.,* (2009) investigated an approach for eliminating the multiple estimation problems of impedance-based fault location methods applied to power distribution systems by using single-end measurements of current and voltage fundamentals at the power substation. Three test systems were used to identify the faulted lateral. The technique posited that by identifying only one faulted node, the multiple estimation problem is avoided. Salim *et al.,* (2009) proposed an extended impedance-based fault-location formulation for generalized distribution system in which unbalanced distribution feeders were considered. The input data to the proposed method were voltages and currents measurements, and it considered load variation effects, and different fault types. Results are obtained from simulations of a distribution system in Southern Brazil. Although, the technique presented is commendable, no real-life fault cases have been analyzed by the software.

Filomena *et al.,* (2009) suggested an algorithm for SL-G and 3Ph. faults for a distribution network. The inputs were also the apparent impedance computed from the local voltage and current. The capacitive current of the underground cables was compensated for by using an iterative algorithm. The performance of this technique is independent of the fault resistance and distance values. However, the existence of laterals may affect the accuracy level of the fault distance estimate.

De Almeida *et al.,* (2010) presented a method to improve impedance-based methods by allocating Faulted Circuit Indicators (FCI) along the feeder to reduce or eliminate the uncertainty about fault location. The technique proposed used the Chu–Beasley Genetic Algorithm (GA) to optimally allocate a given number of FCIs along distribution feeders. Results were presented for the IEEE 34-bus system and for a 475-bus actual system.

Similarly, a technique that involved the classification into zones using impedances, fuse links installation in the feeder, time comparison, and detection of fuse operation was developed by Dashti and Sadeh (2010). This involved the measurement of fault current, voltage, impedance, and fault duration. These quantities were compared with pre-fault values stored in a database. The measured impedance is used to detect the faulty zone, and by analyzing the fault duration, the operated cut-out fuse is determined, thereby identifying the faulted section.

2.3.2 Fundamental Frequency Based Method

Whenever fault occurs in a power system, the operating quantities contain harmonics and decaying direct current (dc) in addition to the fundamental frequency component. The fundamental frequency techniques for fault detection and diagnosis rely on measurements of power frequency quantities. These measurements are provided by protection instruments such as current transformers (CTs), voltage transformers (VTs), and protection relays located either at terminal stations, zone substations or in the field. The quantities obtained are used in sequence components analysis to determine fault types, faulty phases as well as the fault location estimates.

Das *et al.,* (2000) described a technique implemented with inputs based on the fundamental frequency pre-fault and fault voltages and currents. However, this technique was considered only for SL-G faults and also assumed that loads beyond the faulty node are lumped into a single load at the other end. Similarly, Saha *et al.,* (2001) proposed an algorithm for estimating fault locations on a radial MV network by comparing the measured feeder impedance with the calculated impedance. The algorithm was tested for Line-to-Line (L-L) and SL-G faults respectively. Lee *et al.,* (2004) developed a fault location and diagnosis system in C programming language, in which calculations using the load current and fault current obtained from Electromagnetic Transients Program (EMTP) simulations were used to identify the location of faults. The current waveform pattern at any point in time was compared with the expected operation of the protective devices. Consequently, the comparison of the interrupted load with the actual load was then used to diagnose the exact fault location.

Gohokar and Khedker, (2005), described a method in which both short circuit and open conductor faults are treated simultaneously in an automated 11kV distribution network. During normal operation, the load current is read as 0. After fault, the faulty sections have high fault current denoted by 1. Fault is detected in sections where the status is 1 and 0. The healthy sections have either 1 and 1 or 0 and 0. Yuan *et al.,* (2008) developed a matrix algorithm for fault section location in distribution network with DGs based on current inputs from feeder terminal units. The matrix algorithm is made up of three sub-routines. These include the network matrix, fault information matrix, and the fault detection matrix.

Campoccia, *et al.,* (2010), suggested a method that made use of RMS values of line currents and voltages at the MV/LV substations in the location of all fault types. The developed algorithm was implemented in microprocessors and installed at substations. Takani *et al.,* (2007) suggested a method based on recursive charging current and least-squares. Gong and Guzman, (2011) proposed an impedance based method that uses the impedances, length of the feeder sections, relay event reports, and FCIs, to calculate fault locations in a distribution network.

2.3.3 Comparative Analysis of the Existing Methods

Table 2.2 gives a comparative analysis of the impedance and other fundamental frequency methods. From the various approaches highlighted above, Pereira *et al.,* (2006) and Filomena *et al.,* (2009) presented works using real distribution networks and considered SL-G faults and 3-Ph. faults respectively. Even though they incorporated additional information into their algorithms, it was nevertheless based on measurements from one end only and did not consider underground cable networks.

Similarly, the technique by Girgis *et al.,* (1993) using recursive optimal estimation could fail when used in multiple single-phase, multiphase branches, and underground cable networks. He also assumed the loads to be of constant impedance and did not take into account load uncertainty. The method by Santoso *et al.,* (2000) showed that the distance equation provided better results than the apparent impedance equation.

However, the linear equation method produced more accurate result when the fault was without any resistance. Furthermore, Lee *et al.,* (2004) assumed that all the load impedance were accurately known in the implementation of his technique in which he compared the interrupted load with the actual load in the diagnosis of the fault location.

The inadequacy of the method proposed by Das *et al.,* (2000) is that it is only suitable for SL-G fault. But then, distribution systems are subjected to different types of faults. A technique that comprises of all fault types must be provided in order to enhance the reliability of the fault-location process. A similar method by Saha *et al.,* (2005) presented a dynamic revision of the network configuration by updating a database whenever there is a change in the network impedance.

Gohoker and Khedker (2005) presented a method suitable for low impedance faults and high impedance faults. The method in Sengal, *et al.,* (2005) is time consuming because of the iterative process.

Table 2.2: **Impedance and fundamental frequency methods for fault detection and location in distribution network**

S/N	Paper	Network Model	Database used?	Type of Measurements	Fault Type	Method/Simulation	Real-Life Implementation?	Type of Diagnosis
1	*Girgis et al.,et al.,* (1993)	Real rural overhead network.	Yes	Voltage and current magnitudes at one location.	SL-G, L-L, DL-G, and 3 Ph.	Simulation in EMTP.Fault distance estimation based on apparent impedance.	No	Fault location
2	Santoso *et al.,* (2000)	24kV, 20 bus real overhead network.	No	Single point voltage and current measurement at the substation and positive and zero sequence impedance of the primary feeder.	SLG	Simulation in EMTP. Fault distance estimation based on apparent impedance. First method: Two linear equations were formed from the apparent impedance equation. Second method used the distance equation as a function of the fault resistance.	Yes	Fault location
3	Senger *et al.,* (2005)	13.8 kV real overhead network.	Yes	Voltage and current measurements at substation level and load currents at each node.	A-g, B-C, BC-g, ABC, and ABC-g	Simulation in ATP-EMTP. Algorithm based on a set of equations depending on the fault type.	Yes	Fault classification and location
4	Saha *et al.,* (2005)	10kV, 16 feeder real overhead network	Yes	Phase voltage and 3 Ph. Current measured at supply busbar and faulted feeder respectively.	SL-G, L-L, DL-G, and 3 Ph.	Simulation in ATP-EMTP. Algorithm based on impedance measurement.	No	Fault location
5	a. Pereira et al, (2009) b. Pereira *et al.,* (2009)	13.8kV, 238 bus real overhead feeder.	Yes	[a]: Pre-fault and fault voltage and current phasors at substation level and fault voltage magnitude at some buses. [b]: Sparse measurements of voltage sag magnitudes.	SLG	Simulation in ATP-EMTP. 2006: 236 faults from nodes 3-238. 2009: Five voltage sag meters were placed along the feeder at nodes 11, 109, 166, 176 and 225	Yes	Fault location
6	Morales-Espana *et al.,* (2009)	24.9kV, IEEE 34-node overhead test feeder	No	Voltage and current measurements at substations	SLG, L-L, DLG, and 3 Ph.	Simulation in ATP-EMTP. 5 different fault locations, 11 fault resistances, effects of fault types and loading were considered.	No	Fault location
7	Salim *et al.,* (2009)	13.8kV, 11 bus real underground network.	No	Sending-end voltages, and currents, loads, series impedances, and shunt admittance matrices.	SL-G, L-L, DL-G, and 3 Ph-G.	Network simulation in ATP. Fault location implementation in MATLAB. 31 different fault locations, 5 different fault resistances from 0-100 Ω, and 10 fault types, fault inception angle of 0^o, 30^o, 45^o, 90^o were considered.	Yes	Fault location
8	Filomena *et al.,* (2009)	13.8kV, 9 load bus real underground network.	No	Sending-end voltages, and currents, loads, series impedances, and shunt admittance matrices.	SLG and 3Ph.	Network simulation in ATP-EMTP. Fault location implemented with apparent impedance calculation. 77 different fault locations , 5 different fault resistance from 0-100 Ω, and 770 faults were considered.	No	Fault location

S/N	Paper	Network Model	Database used?	Type of Measurements	Fault Type	Method/Simulation	Real-Life Implementation?	Type of Diagnosis
9	De Almeidaa *et al.,* (2001)	24.9kV, IEEE 34-node system and a real 475-bus network	No	Optimal allocation of faulted circuit indicators	Fault or No-Fault	Chu–Beasley genetic algorithm.	No	Fault location
10	Dashti *et al.,* (2010)	20kV, 11 bus overhead network.	Yes	Fault-induced disturbance current waveform measured at the substation and time-current characteristic of fuse links.	3Ph.	Simulation in ATP-EMTP. Method is based on impedance classifiers.	No	Fault location
11	Das *et al.,* (2000)	25kV real overhead network.	Yes	Fundamental frequency component of prefault and fault voltages and currents measured at the line terminal.	SL-G, L-L, DL-G, and 3 Ph-G.	Simulation in PSCAD-EMTDC. Fault location software was developed in FORTRAN. 5 Ω and 50 Ω fault resistances used in the simulation.	No	Fault location
12	Saha *et al.,* (2001)	10kV, 16 feeder, real overhead network.	No	Voltage and current simulations and measurements from Digital Fault Recorder.	SLG, L-L	Simulations in ATP-EMTP. MATLAB based fault locator model	Yes	Fault location
13	Lee *et al.,* (2004)	22.9kV, 21 node overhead network.	No	Load and fault current estimate at each line section.	SL-G	Simulation in EMTP using 10 fault locations, 4 fault resistance values, and 3 different load models.	No	Fault location and diagnosis
14	Gohokar & Khedkar, (2005)	11kV overhead network.	No	Voltage and current phasors.	LIF and HIF	Simulations in EMTP.	No	Fault detection and fault section identification
15	Campoccia *et al.,* (2010)	21.6kV real overhead network	No	RMS values of line current and voltages at MV/LV substation.	SL-G, L-L, DL-G, and 3 Ph-G.	Simulations in MATLAB/Simulink. Algorithm based on multi point measurements.	Yes	Fault location and diagnosis
16	Yuan *et al.,* (2008)	Distribution network with DGs	No	3ph. Line currents	Not specified	Not specified	No	Fault detection, fault type classification, fault section location
17	Gong & Guzman, (2011)	13kV 3Ph. & 1Ph. Distribution feeders	Yes	Reactance, feeder length, relay reports, FPI	SL-G, L-L, DL-G, and 3 Ph-G.	Field case studies	Yes	Fault location

The shortcoming of the method proposed by Campoccia, *et al.,* (2010) is the cost of execution and the maintenance needed for the microprocessors at each substation. The techniques by Saha *et al.,* (2005) and Salim *et al.,* (2009) for overhead and underground networks respectively are recommended. This is because they provided the best results when compared with the other techniques. Also, they are viable and easy to implement.

However, extensive literature review (Adewole & Tzoneva, 2011) shows declining interest in the impedance and other fundamental frequency based methods in fault detection and diagnosis.

2.4 Review of High Frequency Components and Travelling Wave Methods

2.4.1 High Frequency Components and Travelling Wave Methods

High frequency components and travelling wave techniques involve the analysis of signals that are with higher frequency than the fundamental frequency signals, i.e. harmonics, natural oscillating frequencies, and travelling waves. They can present a challenge when attempts are made to extract them from the fault composite signal due to their usually low magnitudes and the overpowering effect of background noise.

This technique is implemented in two ways:

- Analysis of the harmonics and natural oscillating frequencies
- Travelling wave technique

In the harmonics analysis technique, the higher harmonic currents are examined to differentiate the characteristics typical for load switching, capacitor switching and/or arc producing devices from those of faults surges along the lines. These frequency components propagate away from the fault position in both directions. As these waves travel along the feeder, they stumble upon a number of discontinuities and are reflected back towards the fault point and will arrive at the relay ends giving a highly correlated signal. The initial values of these waves mainly depend on the fault position on the feeder, the fault path resistance, and the instance of fault occurrence.
Equation (2.20) gives the fault distance (x) for a double ended line on the basis of the travelling wave technique (Zimath *et al.,* 2009).

$$x = \frac{l + k_c(t_a - t_b)}{2} \tag{2.20}$$

l is the cable length of the line

k_c is the travelling wave propagation speed

$t_a - t_b$ is the time difference between the times of arrival of the travelling waves of each side.

Wavelet Packets (WP) are linear combinations of wavelets with the same properties of their parent wavelets. Wavelet Transform (WT) decomposes signals into successive wavelet components corresponding to a time-domain signal within a frequency range. The original signal can be represented in terms of a wavelet expansion that utilizes the coefficients of the wavelet functions.

Sequel to the Fourier transform, a technique called Short-Time Fourier Transform (STFT) was proposed. STFT is used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. The idea behind the STFT was the segmentation of a signal by using a time-localized window, and performing the analysis for each segment.

However, the STFT uses the same time-window size for all frequencies. Wavelet analysis overcomes this deficiency by utilizing a windowing technique with variable-sized regions with long time intervals for precise low-frequency information, and the shorter regions where high-frequency information is required, thereby giving a time-frequency representation of the signal. Several wavelets can be constructed from a function $\psi(t)$ known as a “mother wavelet”, which is confined in a finite interval. That is, WT decomposes a given signal into frequency bands, and then analyses them in time. WT are broadly classified into Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT).

CWT is defined as the sum over all time of the signal to be analyzed multiplied by the scaled and shifted versions of the wavelet function ψ. The CWT of a signal $x(t)$ is defined as shown in (Santoso *et al.,* 2000; Borghetti *et al.,* 2007):

$$CWT\ (a,b) = \int_{-\infty}^{\infty} x(t)\psi_{a,b}(t)\,dt \qquad (2.21)$$

$\psi(t)$ is the mother wavelet, a is the scale factor, b is the translation factor, and $a,b \in R, a \neq 0$ (R is a real continuous number system) are the scaling and shifting parameters, respectively. $|a|^{-1/2}$ is the normalization value of $\psi_{a,b}(t)$ so that if $\psi(t)$ has a unit length, then its scaled version $\psi_{a,b}(t)$ also has a unit length.

CWT is used to divide a continuous-time function into wavelets that are highly localized in time.

Unlike Fourier transform, the continuous wavelet transform possesses the ability to construct a time-frequency representation of a signal that offers very good time and frequency localization and is an excellent tool for mapping the changing properties of non-stationary signals.

DWT analyses a given signal with different resolutions for different frequency range. This is done by decomposing the signal into coarse approximation and detail coefficients. It employs the scaling and wavelet functions for this. DWT is a variant of WT that has a change in the analysis scale by a factor of two.
This is given by (Daubechies, 1992; Makming *et al.,* 2002; Jung *et al.,* 2007):

$$DWT(m,n)=\frac{1}{\sqrt{2^m}}\sum_k f(k)\psi\left(\frac{n-k\,2^m}{2^m}\right) \tag{2.22}$$

where $f(k)$ is a discrete signal, $\psi(n)$ is the mother wavelet (window function), m and n are discrete time scale parameters, k is the number of coefficients, 2^m is the variable for scale, $k2^m$ is the variable for shift, and $1/2^m$ is the energy normalization component to ensure the same scale as the mother wavelet.

Magnago and Abur (1999) developed a method using the high frequency signals measured at the substation. The proposed method identified the fault path based on the fault transient signals information. Afterwards, it calculated the fault location along the identified path by using the power frequency signals. Tang *et al.,* (2000) suggested a method based on the voltage magnitude difference between the devices' terminal voltages with fault indicators inserted into the distribution feeders in order to identify the faulty section. Thomas *et al.,* (2003) proposed a technique which was implemented in a 23.8 kV distribution system in which measurements of the incident transient currents are taken at the substation, while the voltage magnitudes are estimated by using the transmission line modelling method. A cross-correlation function was then used to identify the transient travelling waves and compute the distance to the fault. The use of a longer correlation window was suggested as a possible way of eliminating discontinuities in single-end technique.

Yanqiu *et al.,* (2004) developed an algorithm for faulted feeder detection by using transient signals and steady state signals. These two signals were compared for a Peterson-coil-grounded system. It determines the faulted feeder by using wavelet packet analysis to extract the fundamental and high frequency components. Öhrström *et al.,* (2005) presented a paper based on the adaption of principles developed for high voltage transmission systems in medium voltage distribution networks. Different travelling wave algorithms were evaluated using correlation, directional wave, and wavelets methods test systems. A fault location technique based on high frequency signals was developed by Jalali and Moslemi (2005). The method identified the faulted lateral based on the wavelet decomposition of the transient signals. The identification of the faulted lateral was based on the voltage signal decomposition in a frequency spectrum of 12.5 kHz to 25 kHz. The technique took advantage of the special properties of the wavelet transform to differentiate between faults occurring along different laterals of the main feeder, with equal distance away from the main substation.

Borghetti *et al.,* (2006) proposed an algorithm for fault location for use in distribution networks. The algorithm was based on wavelet transform analysis of the voltage waveforms recorded during fault. The procedure was implemented using CWT. The IEEE 34-node test distribution system was used to evaluate the performance of the technique. Salim *et al.,* (2007) used the wavelet transform through the multi-resolution analysis of the current signals measured at the relay point for SL-G fault detection in primary distribution systems. Yang *et al.,* (2007) also proposed a method based on the Multi-Resolution Analysis (MRA) of measurement parameters. This approach detected High Impedance Faults (HIFs) using the MRA output in which the distribution feeder voltage and current signals were extracted even when these features were very weak. The feature extraction system based on DWT and the feature identification technique based on statistical confidence are then applied to discriminate effectively between the HIFs and the switch operations.

Hizam and Crossley (2007) suggested a method in which high frequency voltage and current components are taken at one point. These signals are analysed by comparing the distance between the peaks of the travelling waves with the known feeder distance. Dwivedi *et al.,* (2008) worked on fault classification and location using wavelet multi-resolution approach demonstrated on 7-node and 19-node three-phase test systems simulated with different fault types, different fault resistances, and different fault inception angles. Oliviera *et al.,* (2009) developed an algorithm implemented in MATLAB and tested with simulation obtained from Development Coordination Group’s (DCG) Electro-Magnetic Transients Program-Restructured Version (EMTP-RV). A modified IEEE 13-Node Test Feeder was used to estimate the faulted section of an unbalanced system in a noisy environment. The technique used travelling wave method. The methodology used was autocorrelation theory and local voltage data as the only input.

Akorede and Katende (2010) presented a wavelet transform method for the detection of HIF events in distribution networks. Electrical models for a HIF and capacitor switching event on a power network were developed in MATLAB. The classifier algorithm is based on a moving window approach whereby one-cycle window of the DWT output is moved continuously by one sample. A method for fault detection and location in underground distribution system using DWT was proposed by Zhao *et al.,* (2000). A 400 kV underground cable system was simulated by ATP-EMTP for three phase balanced loads under various system and fault conditions. Daubechies db-8 wavelet transform was employed to analyse fault transients.

Butler-Purry and Cardoso, (2008) presented a preliminary research work on an on-line technique capable of fault detection and prediction of the remaining life of the cable lateral. Two experimental setups were designed and deployed for on-line monitoring of underground cable lateral to record data. An analysis of the recorded data which was based on a time-frequency multi-resolution technique was presented. The results of using an artificial neural network for pattern identification of the recorded data are also presented.

Furthermore, Ngaopitakkul *et al.,* (2010) developed a DWT based travelling wave algorithm to detect the high frequency components and also identify fault locations in underground distribution system. The first peak time obtained from the faulty bus was employed for calculating the distance of fault from sending end. The validity of the proposed technique was tested with various fault inception angles, fault locations, and fault types. ATP/EMTP was used to simulate fault signals at a sampling rate of 200 kHz. Apisit and Ngaopitakkul (2010) investigated a technique for identifying the faulty phase in an underground cable. Various faults were simulated in ATP-EMTP. The wavelet transform was used to extract the high frequency components superimposed on the fault signals. The coefficients of positive sequence current signals are calculated and employed in a fault detection decision algorithm.

2.4.2 Comparative Analysis of the Existing Methods

A summarized comparative analysis of the high frequency and travelling wave methods is given in Table 2.3.

Magnago and Abur, (1999), Thomas *et al.,* (2003), Ohrstrom *et al.,* (2005), Jalali and Moslemi (2005) all presented travelling wave methods designed for the location of only one fault type in overhead lines. The method proposed by Thomas *et al.,* (2003), though designed for double and single ended measurements, could fail when

Table 2.3: **High frequency components and travelling wave methods for fault detection and location in distribution network**

S/N	Paper	Network Model	Database Used?	Type of Measurement	Fault Type	Method/Simulation	Real-Life Implementation?	Type of Diagnosis
1	Magnago & Abur, (1999)	13.8kV overhead network.	No	Single ended DFR measurement of voltage and current magnitudes and feeder configuration	SL-G	ATP simulations using fault resistance, fault inception angle, load variation. Method used was Wavelet transform.	No	Fault location
2	Thomas *et al.,* (2003)	23.8kV real overhead real network.	No	Single ended and double ended voltage and current measurements.	3 Ph.	MATLAB power system blockset. Sampling at 1.25 MHz sampling rate with 8 bit resolution. Travelling wave based.	Yes	Fault location
3	Ohrstrom *et al.,* (2005)	12kV overhead network.	No	Phase voltages and currents recorded at different locations.	SL-G, L-L, 3Ph.	PSCAD/EMTDC simulations with 10kHz sampling frequency, different fault resistances (5, 50, and 500 Ω) and different fault inception angles (5°, 45°, and 90°. MATLAB implementation of travelling wave method.	No	Fault detection and location
4	Jalali *et al.,* (2005)	24kV real overhead network.	Yes	Voltage transducer with a bandwidth of 50 kHz.	SL-G	ATP simulation. 100kHz sampling rate. Wavelet analysis based.	No	Fault section identification and location
5	Borghetti *et al.,* (2007)	24.9kV IEEE 34-node test feeder (overhead network).	No	Recorded fault originated voltage transients.	SL-G and 3 Ph.	EMTP-RV simulations. Fault resistance of 0, 10, and 100 used in simulation. CWT method used.	No	Fault location
6	Salim *et al.,* (2007)	4.8kV, IEEE 37 bus test feeder (underground feeder).	No	Sending-end voltages and currents, loads, series impedances, and shunt admittance matrices.	SL-G and HIF	The time-step simulation used was 192 samples/cycle, or 11,564 Hz and Daubechies-8 mother wavelet.Fault resistances of 0, 10, 20, 50, 100, 500, 1000, 1500, and 2000 Ω were simulated in EMTP.	Yes	Fault detection
7	Yang *et al.,* (2007)	11.4 kV, 16 feeder real network.	No	zero sequence current.	HIF or No fault	Sampling frequency was set to 6 kHz, and the resolution level of the MRA filter banks was set to 5. 2 fault detection algorithms were used: CWT and DWT.	Yes	Fault detection
8	Hizam & Crossley, (2007)	33/11kV real underground and overhead lines.	No	High frequency voltage and current measurements at one end.	SL-G	PSCAD-EMTDC modelling and simulation of fault conditions. Fault resistance of 0.01 and time step of 0.8µsec. Travelling wave based.	Yes	Fault section identification and location

S/N	Paper	Network Model	Database Used?	Type of Measurement	Fault Type	Simulation	Real-Life Implementation?	Type of Diagnosis
9	Dwivedi *et al.,* (2008)	7-node and 19-node 3 ph. Networks.	No	Current measurements at the substation.	SL-G, L-L, L-L-G, and 3 Ph.-G	Fault simulations in MATLAB. Multi-Resolution Analysis (MRA) based fault identification and location algorithms.	No	Fault classification and location
10	Oliveira *et al.,* (2009)	IEEE 13 Node Test Feeder, (underground and overhead lines).	Yes	Three phase local voltage magnitudes at the substation terminals.	SL-G	Algorithm implemented in MATLAB. EMTP-RV used for simulations. Travelling waves and autocorrelation theory based.	No	Faulted section identification.
11	Akorede & Katende, (2010)	11kV, 4 feeder real network.	No	Current and voltage magnitudes.	HIF and No fault	Simulations in MATLAB. DWT based with sampling frequency of 6 kHz	No	Fault detection
12	Zhao *et al.,* (2000)	400 kV, underground network.	No	3 Phase fault current signals.	SL-G, L-L, L-L-G, and 3 Ph.	ATP-EMTP simulations. DWT Daubechies-8 based scheme with 40kHz sampling rate.	No	Fault detection and classification.
13	Butler-Purry & Cardoso, (2008)	7.2kV underground network.	No	Phase and neutral current using current transducers	SL-G	Multi-Resolution Analysis (MRA) based.	No	Fault detection
14	Ngaopitakkul *et al.,* (2010)	200kV underground network.	No	Current signals measured at the sending end.	SL-G, L-L, L-L-G, and 3 Ph.	ATP/EMTP simulations using various fault locations, fault inception angles, and fault types. Algorithm in MATLAB using DWT (Db4) and travelling wave based with a sampling rate 200 kHz.	No	Fault detection and location
15	Yanqui et al, 2004	Distribution network	No	Bior5.5 scale-4 decomposition of the line current	SLG.	Simulations in MATLAB with 4000Hz sampling frequency'.	No	Fault detection

adapted for other fault types apart from the 3-Ph. fault which it was hitherto designed for. The proposal by Öhrström *et al.,* (2005) was hampered by the lack of adequate sampling speed.

Borghetti *et al.,* (2006) did consider SL-G and 3-Ph. faults using IEEE 34 node test feeder implemented in CWT and the developed method would have been perfect if test result were presented for 2Ph. and 2Ph.-G faults. Fault location in underground network was investigated by Salim *et al.,* (2007), Zhao *et al.,* (2000), Butler-Purry and Cardoso (2008), Ngaopitakkul *et al.,* (2010) using the wavelet method with different fault parameters. Salim *et al.,* (2007) studied SL-G faults and HIF in their proposed method. Similarly, Yang *et al.,* (2007), Akorede and Katende (2010), also presented a technique for HIF location on overhead lines. Yang *et al.,* (2007) used the multi-resolution analysis of measurement parameters to detect HIFs.

Amongst the literature analysed in this review for the high frequency and travelling wave methodologies, only the techniques by Hizam and Crossley (2007), and Oliviera *et al.,* (2009) considered fault location for both underground and overhead networks; albeit, for only SL-G faults. However, the method by Oliviera *et al.,* (2009) is highly recommended since it did take into cognisance fault location in noisy environments which are common in substations where digital relays are located.

2.5 Review of Knowledge Based Methods

Knowledge based methodologies are made up of:

- Computational intelligence and mathematical methods
- Distributed devices methods
- Hybrid methods

2.5.1 Computational Intelligence and Mathematical Methods

The mathematical technique relies on probabilistic and statistical methods to address the difficulties with fault location for very high resistance single-phase earth faults. Also, matrices can be formed to represent relations between nodes and sections as well as voltage sensors, present on a distribution network.

The Computational Intelligence techniques consist of (Konar, 2000; Choppin, 2005; Bollen *et al.*, 2009; Saha *et al.*, 2010):

- Expert System (ES)
- Fuzzy Logic (FL)
- Artificial Neural Network (ANN)
- Optimization methods

An expert system can be defined as an interactive system that exhibits an expert level of knowledge in a specific domain (Konar, 2000; Choppin, 2004; Bollen *et al.*, 2009; Saha *et al.,* 2010). An expert system solves problems in this domain by making use of a knowledge base and an inference engine. The knowledge base is made up of production rules and facts, while the inference engine uses a search space tree to determine the heuristic search structure.

Similarly, fuzzy logic systems try to give approximate solution to problems in ways similar to how humans express knowledge. This it does by using approximate terms to express knowledge, thus allowing the representation of imprecise human knowledge in a natural and logical manner which is neither true (1) nor false (0) (Konar, 2000; Choppin, 2004; Bollen *et al.*, 2009; Saha *et al.,* 2010).

An Artificial Neural Network (ANN) is an information processing paradigm structured similar to the biological nervous systems. It consists of a large number of highly interconnected processing components which are known as neurons and just like humans, learn by example (Bishop, 1996; Nilsson, 1998; Haykin, 1999; Konar, 2000; Choppin, 2004). These layers consist of neurons which are connected to other neurons through synaptic weights and biases. It is by adjusting these connection weights and biases that training is implemented. Neural Networks (NNs) take decisions based on previously stored knowledge. Optimization methods like genetic algorithms are computational methods that emulate nature’s evolutionary process in the search for the global optimal solution (Saha *et al.,* 2010).

Butler and Momoh, (1993) introduced a neural network approach based on clustering algorithm to detect and classify line faults in a power distribution system. Teo, (1993) proposed a fault diagnosis system for power distribution networks using machine learning of fault patterns through a network state capturing mechanism. A similar method capable of HIF and LIF diagnosis was presented by Mohamed & Rao, (1995), and implemented with a cascaded multilayer ANN structure using the Back-Propagation (BP) algorithm.

Wen and Chang, (1998) presented a fault diagnosis method for distribution network. The method made use of parsimonious set covering theory and Genetic Algorithm (GA). A method aimed at locating the fault section in an electric power system was developed by Aygen *et al.,* 1999. The estimation of the faulted section was based on Artificial Neural Networks using binary information from protection relays and circuit breakers. Chan and Lu, (2001) proposed a fault-locating scheme using the concept of a characteristic vector utilizing digital Over-Current (OC) relays. The proposed scheme was capable of locating both bus fault and feeder fault by mapping corresponding fault current vector with the characteristic vectors in a pre-built database. By updating the characteristic vector database, the proposed scheme can be applied to the changed system as well, thereby achieving adaptive feature.

Al-Shaher *et al.,* (2003) developed an artificial neural network approach for determining the locations of three phase faults in a multi-ring distribution system using the feeder fault voltage, circuit breaker status, real power of feeders during the normal condition and during short circuit, etc., to train the neural network. This technique was applied to an existing 13.8kV distribution network. Lee *et al.,* (2004) described a technique based on load current and fault current estimation fort fault locations. According to Martins *et al.,* (2003), the Clark transformation of the line currents into the alpha, beta, and zero sequence (αβ0) components can be used to identify and locate different types of fault. This αβ0 space vector is a modified version of Clarke-Concordia transformation. The methodology comprised of data acquisition, mathematical treatment by Clarke-Concordia transformation, fault identification by comparison of fault and pre-fault characteristic curves, and lastly, fault location. The location of the fault was obtained from the relationship between distance and the eigenvalue of the line currents matrix. Cormanne *et al.,* (2006) presented a statistical methodology for fault location in distribution systems. The approach was based on the statistical modelling and extraction of information from data bases associated to fault recording. Potential limitations for the proposed methodology are the selection of the number of groups, the proportion of samples in each group and the initial values required by the algorithms. Mora *et al.,* (2006) described a method which involves the modelling of a 25kV power distribution system and protective relaying using ATP and MATLAB to obtain an extensive fault database. This database having 930 fault situations was used to perform different types of system analysis.

Fulczyk *et al.,* (2007) suggested a technique implemented with three phase current from all (*n*) terminals and three-phase voltage. The subroutines are formulated with the use of generalized fault loop and fault models. Multi-criteria selection procedure was applied for selecting the valid subroutine which indicates the faulted line section.
Herraiz *et al.,* (2007) developed a technique that automatically calculated the location of a fault in a distribution power system. The application was based on a *N*-ary tree structure. Salim *et al.,* (2008) developed a scheme made up of three (3) different processes. The fault detection and classification technique was wavelet based, while the fault-location technique used the impedance method and local voltage and

current fundamental phasors. ANN was used for fault section identification by using the local current and voltage signals to estimate the faulted section. A method based on differential current ratio was proposed by Guo-fang and Yu-ping, (2008) for application in distribution networks with Distributed Generators.

Mora-Florez *et al.,* (2009) proposed a technique of statistical nature based on finite mixture. A statistical model was obtained from the extraction of the magnitude of the voltage sag registered during a fault event, along with the network parameters and topology. The potential limitations for the proposed methodology are the selection of the number of groups, the proportion of samples in each group and the initial values required by algorithms. These difficulties are overcome by introducing theoretical and heuristic criteria. Rayuda, (2010) proposed a distributed diagnostic algorithm which has been implemented on a New Zealand sub-system. Distributed Architecture for Power Network Fault Analysis (DAPFA) is an intelligent, model-based diagnostic algorithm that incorporates a hierarchical power network representation and model based on substation automation implementation standards. The structural and functional model is a multi-level representation with each level depicting a more complex grouping of components than its predecessor in the hierarchy. The distributed functional representation contains the behavioural knowledge related to the components of that level in the structural model. The diagnostic algorithm of DAPFA was designed to perform fault analysis in pre-diagnostic and diagnostic levels. These two phases provide real-time analysis and final diagnostic analysis respectively. The diagnostic algorithm also incorporates knowledge-based and model-based reasoning mechanisms. One of the model levels was represented as a network of neural networks.

Kumar *et al.,* (2011) studied the capability of SVMs for prediction of fault in power systems. The inputs of Support Vector Machines (SVMs) model were power and voltage values. IEEE 118 test feeder was modelled. A comparative study was done between the developed SVMs model and Learning Vector Quantization (LVQ) ANN model. An equation was developed for the prediction of fault in a power system based on the developed SVMs model. A method based on Radial Basis Function (RBF) neural network was proposed by Zayandehroodi *et al.,* (2011). This method determines fault type and fault location in distribution network with DGs by using normalized fault current of the main source for the determination of the fault type, while the three phase short circuit current of all the sources in the network are used as input to two RBF neural networks for fault location.

Rezaie *et al.,* 2011 described a technique for automatic fault diagnosis in a distribution network with DG based on RBF neural network. This technique made use of current inputs in the process of training the four neural networks for four different fault classes.

Table 2.4 gives a summary of the computational intelligence techniques reviewed in this thesis.

2.5.2 Distributed Device Based Methods

The basic idea of the distributed device method is the development of a technique based on the integration of additional parameters like network information system and distribution automation into the existing framework of a distribution network protection system. With respect to fault detection and location, this means that only the existing device and data is utilized instead of developing new network models and adding new equipment. Furthermore, the distributed device method provides an excellent environment for further processing of data and it is also possible to integrate graphical user interfaces for fault management, information on the terrain, and weather conditions into the system.

Järventausta *et al.,* (1994) suggested a method of integrating network information systems and distribution automation into existing infrastructure. This included a database of the data needed for fault current analysis of the distribution network and also information on the real time topology. Furthermore, weather conditions and information from fault detector can be incorporated, with fuzzy logic applied for further processing.

Wang *et al.,* (2000) also suggested a method based on the mathematical formulation of a matrix. Matrices were formed between the voltage sensors and the sections and also between the sections and the nodes in the electric network. Thus, faulted section could be found through mathematical operation of the matrices. Kezunovic (2001) gave a technique in which measured data was matched with historical fault data. This historical fault data was compiled in a database comprising of previous records of voltage sag measurements taken during fault, fault type, and location. When a fault occurs, the measured voltage sags are compared to the database. However, this technique would not work if the fault has never happened in that location or is not saved in the database.

Furthermore, Kezunovic *et al.,* (2011) described a method for automated processing and analysis of disturbance data in substation. However, it was necessary to have the

Table 2.4: **Computational intelligence and mathematical methods for fault detection and location in distribution networks**

Nr.	Paper	Network Model	Database Used?	Type of Measurements	Fault Type	Method/Simulation	Real-Time Implementation	Type of diagnosis
1	Butler & Momoh, (1993)	Real distribution network.	Yes	Current waveform.	SL-G and Open circuit	Laboratory emulation. Supervised clustering algorithm used for NN.	Yes	Fault detection and classification
2	Teo, (1993)	22kV, 17 feeders overhead network.	Yes	Load flow calculation.	Overload, Fault, and No fault.	Distribution network simulator implemented by an IBM PC-AT and linked to the diagnostic system microcomputer. Machine learning based with knowledge representation and inference engine developed using Turbo Pascal.	No	Fault diagnosis
3	Mohamed & Rao, (1995)	24kV overhead network.	Yes	Local substation current and voltage measurements.	HIF and LIF	Multilayer ANN structure with Back-propagation (BP) learning algorithm used.	Yes	Fault detection, classification, and fault section identification.
4	Wen & Chang, (1996)	11kV overhead network.	Yes	Operational and tripping information from relays and circuit breakers respectively.	Fault and No fault.	Set Covering Theory and GA used.	No	Fault Diagnosis
6	Chan & Lu, (2001)	22kV, 4 buses overhead network.	Yes	Fault current magnitudes at relaying locations.	SLG, L-L, L-L-G, and 3 Ph.	Mapping of corresponding fault current vector with the characteristic vectors.	No	Fault location
7	Al-Shaher & Sabra, (2003)	13.8kV, 145 buses real multi-ring network.	Yes	Pre-and post-fault real and reactive power, short circuit feeder voltage, circuit breakers and motor statuses.	3Ph.	ANN based.FFNN: 17 Inputs, 2 hidden layers (10 & 6 neurons), and 3 Outputs. Learning: Back-propagation.	Yes	Fault location
8	Martins *et al.,* (2002)	20kV, 3 feeders underground network.	yes	Line current signals.	SLG, L-L, L-L-G, and 3 Ph.	Fault conditions simulations using MATLAB/Simulink. Eigenvalue and ANN based learnin algorithm. FFNN: 2 Inputs, 1 Hidden layer (5 neurons), and 1 Output. Learning : Error-back-propagation	No	Fault detection, classification, and location.
9	Cormane *et al.,* (2006)	21 buses overhead network.	Yes	Voltage and current signals measurement at the distribution substation.	SLG, L-L, L-L-G, 3 Ph, 3 Ph.-G.	ATP-EMP simulation of faults. Statistical based method using distributions. Principal Components Analysis, k-means algorithm, etc.	No	Fault location
10	Mora *et al.,* (2006)	25kV, 25 node real overhead network.	Yes	Current and voltage measurements.	SLG, L-L, L-L-G, and 3 Ph	ATP-EMTP and MATLAB simulation of faults. Multivariate Data Analysis (LAMDA) and SVM based method.	Yes	Fault location
11	Herraiz *et al.,* (2007)	25kV real network (overhead line and underground cable).	Yes	Current and voltage measurement at the line terminal.	SLG, L-L, L-L-G, and 3 Ph	Fault data from a real distribution line. A method based on N-ary tree data structure was developed in MATLAB.	No	Fault location

Nr.	Paper	Network Model	Database Used?	Type of Measurements	Fault Type	Method/Simulation	Real-Time Implementation?	Type of diagnosis
12	Salim *et al.,* (2008)	Underground distribution network.	Yes	Local voltage and current fundamental phasors.	SLG, L-L, L-L-G, and 3 Ph	Simulations in ATP-EMTP. Algorithm implemented in MATLAB. Fault detection and classification: wavelet based. Fault-location: Impedance based. The fault section determination method is ANN based.	Yes	Fault detection, classification, fault dection identification, and fault location.
13	Mora-Florez *et al.,* (2009)	25kV, 25 node distribution system.	Yes	Single-end measurements of voltage signals measured at the distribution substation.	SLG, L-L, L-L-G, 3 Ph, 3 Ph.-G.	The method used is based on statistical modelling using the k-means algorithm and mixtures distribution.	Yes	Fault location
14	Kumar *et al.,* (2011)	IEEE 118 bus system.	Yes	Voltage and power values.	Fault and No fault	SVM was developed in MATLAB. The number of support vectors = 8.	No	Fault section identification and location
15	Aygen *et al.,* 1999	9 section, 4 bus power system	Yes	Binary information from protection relays and circuit breakers	Fault and No fault	Unspecified.	No	Fault section estimation
16	Guo-Fang & Yu-Ping (2008)	69-segment, 8-lateral urban distribution system with DG.	No	Differential currents ratio.	3Ph. Short circuit	Power System Simulation for Engineering (PSS/E 30.1) was used. Input: Network parameter & fault current data from feeder terminal units. Algorithm based on computation of differential current ratios	No	Fault location
17	Zayandehroodi et al (2011).	14-bus, 20kV distribution system with DG.	No	Normalized fault current & 3Ph. Short circuit current.	SLG, L-L, L-L-G, 3 Ph, 3 Ph.-G.	DIgSILENT PowerFactory simulations and RBFNN designed in MATLAB.	No	Fault type and fault location
18	Rezaie *et al.,* 2011	22-bus, 20kV distribution system with DG.	No	Normalized fault current & 3Ph. Short circuit current.	SLG, L-L, L-L-G, 3 Ph, 3 Ph.-G.	DIgSILENT PowerFactory simulations and RBFNN designed in MATLAB.	No	Fault diagnosis and fault location

settings for the Intelligent Electronic Devices (IEDs), channel assignments, binary information from the IEDs and Circuit Breakers (CBs), and power system parameters.

2.5.3 Hybrid Methods

The hybrid approach for fault detection and location is adopted to formulate a new topology that harnesses the benefits offered by the combination of two or more methodologies. Toward this end, the main goal is to improve the accuracy and reliability of the resulting hybrid technique, while reducing or inhibiting the limitations of the individual methodologies.

Fukuyama and Ueki, (1993) developed a fault analysis system comprising of an Expert System (ES), Neural Networks (NNs), and a fault analysis package. The inputs used were fault voltage and current waveforms, binary information from protection relays, and CBs. The ES performed the fault section estimation task, while the NNs took care of the fault detection, fault resistance, fault point, and single/multiple faults decisions.

The fault analysis package is used to verify the decisions made by the NNs. A method based on recursive wavelet and Kohonen NN was proposed by Assef *et al.,* (1996) for distribution system. The inputs to the NN were the wavelet coefficients from the residual current and voltage in the faulty feeder.

Martins *et al.,* (2003) proposed a similar technique which used the Eigen value and an artificial neural network based learning algorithm. The method identified and classified the fault type before giving the fault location. Barros et al, 2003 described a method based on Short Time Fourier Transform (STFT) and Kalman filters for the detection of Power Quality (PQ) disturbance. Yuehai *et al.,* (2004) proposed an expert system for power system fault analysis. This system is based on intelligent agent theory, knowledge discovery theory, and rough set theory using threshold pickups and a logic table. The inputs are ratios of the zero sequence, negative sequence, and positive sequence phase angles depending on the fault type.

Thukaram *et al.,* (2005) proposed a method based on Artificial Neural Network (ANN) and Support Vector Classifiers (SVCs) for fault location distribution systems. The inputs used were substation measurements and binary statuses from circuit breakers and relays. Principal Component Analysis (PCA) was used to analyse the data. A practical 52-bus distribution system with loads was used for the study. A method based on intelligent systems and statistical techniques with phase voltages and line currents as input was proposed by Ziolkowski *et al.,* (2006). This method was aimed at identifying and classifying faults in distribution systems.

Another proposal was by Chunju *et al.,* (2007). This method employed wavelet fuzzy neural network to measure post-fault transient and steady-state measurements. Fuzzy theory and neural network are employed to fuzzify the extracted information.
Wavelet was then integrated with fuzzy neural network to form the Wavelet Fuzzy Neural Network (WFNN). This fault location method however, involved a lot of calculation and took a lot of time in the training. A technique using advanced signal-processing combined with neural network was presented by Samantaray *et al.,* (2008). In the proposed approach, time-frequency and time-time distributions of the High Impedance Fault (HIF) and No-Fault (NF) signals are extracted respectively. They used the features extracted to train and test a probabilistic neural network (PNN) for an accurate classification of HIF from no fault.

Baqui *et al.,* (2011) developed a technique based on the combination of WT and ANNs for addressing the problem of HIFs detection in electrical distribution feeders. The change in phase current waveforms caused by faults and normal switching events was made use of. The DWT decomposes the time domain current signals into different harmonics in time-frequency domain and extracted the required components for the training of the ANNs. This pre-processing reduced the number of inputs to the ANN and improves the training convergence. SimPowerSystem in MATLAB was used to acquire the data of several HIFs, Low Impedance Faults (LIFs), and normal switching events and the multilayer perceptron network and Levenberg–Marquardt back-propagation algorithm were used for the implementation.

Aslan and Türe, (2011) proposed a method using Programmable Logic Controller (PLC). This technique made use of the pre-fault and post-fault current and voltage measurements. Also, the travelling wave method suffers from discontinuities caused by the numerous sub-feeders that are characteristic of a radial distribution system. The discontinuities may be between the end of the line and the fault point, and these add up the reflections to the transient waves from the fault. Furthermore, current measurements were transferred from the lateral to the substation, thus addressing the issue of multiple fault point locations. An off-line processing of the collected data is carried out to locate the shunt faults on the main feeder or in the laterals. The implementation of the fault location algorithm is fully automated by scanning the overhead feeder or faulted lateral at 10 m intervals. A fault is initially assumed at a point of the sub-feeder. When fault is detected in the sub-feeder, the pre-fault and post fault data at the Digital Fault Recorder (DFR) are updated to the tap of the prospective feeder for fault location analysis. The fault path currents obtained for each assumed fault position are then written into an output file for further inspection. The fault location is taken as the point within the minimum fault currents path. A rule based algorithm using wavelet energy spectrum entropies derived from three phase and zero sequence currents was proposed by

Adewole and Tzoneva, (2012a) for distribution network fault detection and classification. In a further work by Adewole and Tzoneva, (2012b), the performance of wavelet entropy based algorithms was compared with others based on statistical methods such as standard deviation and mean absolute deviation.

Table 2.5 gives a comparative summary of the hybrid techniques reviewed in this thesis. From this, it can be seen that the combination of wavelet transform and knowledge based method presented interesting challenges and prospect worth investigating.

2.5.4 Comparative Analysis of the Existing Methods

Butler and Momoh (1993) and Mohammed and Rao (1995) proposals considered the effects of fault resistance and fault inception angle in the implementation of their techniques. The former also considered load variation. But their methods only covered SL-G faults. Although, Teo (1993) trained and verified his diagnostic system on two typical distribution systems, it took a lot of programming effort to develop the diagnostic system in Pascal. The technique developed by Wen and Chang, (1998) was shown to be fast and flexible, and of potential for on-line fault diagnosis in actual electrical distribution networks. It is also capable of dealing with complicated faults and could find multiple optimal solutions directly and efficiently in a single run. However, it was not tested for a large distribution system. Thukaram *et al.,* (2005), and Kumar *et al., (2011*) both used SVMs in their proposed techniques. Thukaram *et al.,* (2005) proposed the combination of SVCs and FFNNs for their technique.

Despite the good performance of SVM, it suffers from many difficulties in finding the optimum solution when the input vector dimension or the sizes of the data set are large. Salim *et al.,* (2008) developed a method that used wavelets for fault detection and classification; impedance method for fault-location; and artificial neural network for fault section determination. Rayuda, (2011) also implemented his diagnostic algorithm using knowledge-based and model-based reasoning mechanisms with one of the model levels represented with neural networks. The method was further

Table 2.5: **Hybrid methods for fault detection and location in distribution network**

S/N	Paper	Network Model	Database Used?	Type of Measurements	Fault Type	Method/Simulation	Real-Life Implementation?	Type of Diagnosis
1	Martins *et al.,* (2003)	10kV, 3 feeders parallel double circuit network (underground cables)	Yes	Single-end line current.	SLG, L-L, L-L-G, 3 Ph.	Fault simulation in MATLAB/Simulink. Clarke-Concordia transformation (αβ0 space vector), eigenvalue and an ANN. FFNN with error back propagation. Two input vectors, one hidden layer, and one output layer.	No	Faults detection, classification, and location
2	Thukaram *et al.,* (2005)	11kV, 3 radial feeders-52 nodes overhead network.	Yes	Phase voltage and current measurements at the substation, circuit breaker and relay statuses.	SLG, L-L, DL-L, 3 Ph.	FFNN and SVM implementation. 3-layered FFNN. Hidden layer neurons: Nonlinear (tangent hyperbolic) transfer function Output layer neurons: Linear transfer functions Supervised training algorithm used is the Levenberg Marquardt.	No	Faults classification and location
3	Chunju *et al.,* (2007)	35 kV, industrial overhead network.	Yes	Post-fault transient and steady-state voltage and current measurements.	SLG	Simulation in EMTP. 6 layers Wavelet-Fuzzy-Neural Network. Wavelet: Daubechies mother wavelet-Db4: Fuzzy NN: Sugeno model, forward multi-layer fuzzy neural network. For the linear section: Least square method. or the nonlinear section: Back propagation algorithm.	No	Fault location
4	Samantaray *et al.,* (2008)	25kV, 3-Phase, overhead radial distribution network and 3-phase meshed network.	Yes	Current signals at no fault and during faults.	HIF and No Fault	Simulation in Power System Blockset (SIMULINK). Time-Frequency (S) and Time-Time (TT) Transforms used. FFNN and Probabilistic Neural Network (PNN).	No	Fault detection and classification
5	Baqui *et al.,* (2011)	13.8 kV, 5 feeders of overhead lines and underground cables.	Yes	Phase current signals from the feeders.	HIFs and LIfs	Simulation using SimPowerSystem Blockset of MATLAB. ANN structure: Multilayer perceptron network and Levenberg–Marquardt back-propagation algorithm. 24 input layer neurons, 14-6 hidden layers neurons, 3 output layer neurons Hidden layer neurons: Tan sigmoid transfer functions Output layer neurons: Log-sigmoid function.	No	Fault detection
6	Aslan *et al.,* (2011)	11kV real overhead distribution network	Yes	Pre- and post-fault current and voltage data from a DFR	SLG, L-L, DL-L	Simulations using EMTP. DFT is used to obtain the filtered pre- and post-fault voltage and current phasors. PLCs are used to transmit the superimposed components.	No	Fault detection and location

S/N	Paper	Network Model	Database Used?	Type of Measurements	Fault Type	Method/Simulation	Real-Time Implementation?	Type of diagnosis
7	Fukuyama & Ueki, (1993)	66kV Power system	No	inputs used were fault voltage and current waveforms, binary information from protection relays, and CBs	SLG, L-L, DL-L, 3 Ph.	Analysis system comprising of an expert system, Neural Networks (NNs), and a fault analysis package. The inputs are information from relays, circuit breakers, voltage and current waveforms.	No	Fault type and fault location
8	Assef *et al.,* 1996	20kV Power distribution system	Yes	Residual current and voltage	Fault & 'No-fault'	Simulation in EMTP. Recursive wavelet transform based ANN designed in MATLAB. The ANN structure used is 7 input neurons, 12 hiddden layer neuron, and 1 output layer neuron.	No	Fault classification
9	Ziolkowski *et al.,* (2006).	Real distribution feeder	No	3 Ph. Voltage and current waveforms	SLG, L-L, DL-L, 3 Ph.	Simulation in ATP. ANN and statistical based tools using MATLAB. Statistical tools incude constant of false alarm rates, skewness values, kurtosis measures, ratio of power, and symmetrical components.	Yes	Fault identification
10	Adewole & Tzoneva, (2012a)	IEEE 34 Node test feeder with DGs	Yes	Wavelet energy spectrum entropies of 3Ph. and zero sequence current waveform	SLG, L-L, DL-L, 3 Ph.	DIgSILENT PowerFactory simulations and rule based algorithm implemented in MATLAB. Input to the decision taking rules were entropy values of DWT decomposition of 3 ph. and zero sequence line currents.	No	Fault detection and classification

designed to detect multiple faults occurrence in power networks. Baqui *et al.,* (2011) implemented their algorithm using wavelets to pre-process the input signals in order to reduce the number of inputs to ANN and improve the training convergence. The ANN structure used was the multilayer perceptron network, while Levenberg–Marquardt back-propagation algorithm was used for the learning algorithm. From the foregoing, the method suggested by Salim *et al.,* (2008) though for underground systems, showed great potential, flexibility, and robustness. Thus, it is recommended for knowledge-based fault detection and location, and its accuracy when adapted for overhead lines need to be further investigated.

2.5.6 Discussions

This review began with a literature survey on the various techniques in distribution network fault detection and location. Namely: Impedance and fundamental frequency, travelling waves and high frequency, and Knowledge Based Methods (KBM) methodologies. The impedance and fundamental frequency based method although simple to implement, suffer from limitations due to the loading of the line, fault path resistance, harmonic components in the current source parameters, measurement error, load imbalance, etc.

As a result, the accuracy obtained is limited to about 2-3% of the total line length and it is unlikely that there would be any significant future improvement (Tang *et al.,* 2000). The main disadvantage of this particular methodology is that of multiple estimations due to the existence of multiple possible fault locations at the same distance in the power distribution system.

Furthermore, Faults can be located with high accuracy by using the travelling wave methodology. It is however faulted for its implementation costs, complexity, the need for high sampling rate, and sophisticated measuring equipment.
Techniques in the knowledge based method are regarded as complex and computational costs may be high in applications involving a high number of faults. Also, there is an inherent problem associated with the upgrading of trained neural networks.

When the KBM method is compared with the other methods, it is seen that KBM presents accurate results. It is fast, scalable, adaptable, cheaper, and remains the best option of all. Among the different KBM methods, Computational Intelligence methods find more application because of their success in fault detection, classification, and location in recent years.

2.6 Conclusion

This chapter began with an introduction to electric power system faults, protection philosophies, and reliability indices. An extensive review of various works done by researchers in the field of distribution network fault detection and diagnosis was also presented. An attempt is made to highlight the state of research in this area. Before now, most research and development in power system fault detection and diagnosis has focused on other segments of the power systems and not distribution networks.

Three (3) methodologies used in fault detection and diagnosis were discussed, i.e. impedance/other fundamental frequency methodologies, high frequency components/travelling wave methodologies, and knowledge based methodologies.

From the foregoing review, it can be inferred that using only one of these methodologies have inherent disadvantage(s). These drawbacks can be by-passed through the hybrid combination of multiple methodologies. For this reason, a hybrid method comprising of wavelet transform, decision-taking rules, and artificial neural network was employed in this thesis.

Chapter Three presents the theory behind the proposed techniques used in the hybrid method. Signal processing techniques including a brief look at Fourier transform and Short-Time Fourier Transform would be presented. Wavelet transform which is the signal processing technique adopted in this thesis will be examined in detail. Neural network architecture, transfer functions, training algorithms, etc. would also be presented.

CHAPTER THREE

SIGNAL PROCESSING AND ARTIFICIAL NEURAL NETWORK THEORY

3.1 Introduction

At the turn of the century, signal processing and computing has found increasing usage in power systems applications than most people could ever imagine. This is due to the ability of digital filters to detect and analyse typical disturbance waveforms accurately.

Furthermore, with the advancements in computer technology, measurements and analyses can now be done easily. This thesis employs a multidisciplinary approach in the design and development of the proposed Hybrid Fault Detection and Diagnosis (HFDD) method. This involves the combination of signal processing, decision-taking rule-based algorithms, and computational intelligence.

Therefore, it is necessary to discuss the theories on signal processing and artificial neural networks (ANN) respectively. This chapter covers signal processing techniques like Fourier transform, short-time Fourier transform, and wavelet transform. Two techniques used in discrete wavelet transform are also presented.

Furthermore, a perspective on the developments of the field of neural networks would be covered. Neural network topologies, activation functions, network training, and learning algorithms would be discussed.

3.2 Signal Processing

3.2.1 Introduction

Signal Processing deals with the operations and analysis of signals in either discrete or continuous time (Prandoli & Vetterli, 2008). In the power system context, signal processing is used for the analysis and selection of characteristic features of measured electrical quantities. Most naturally occurring signals exists in the time domain.

Frequency domain analysis converts such time-amplitude signals to the frequency-amplitude domain. Time-frequency domain techniques analyses a signal simultaneously both in the time and frequency domain. This is done by making use of transforms that are closely connected with the function to be analysed. This approach is very suitable for signals with time varying statistics (Mallat, 2009).

The following signal processing techniques are discussed in the subsequent sub-sections:

- Fourier Transform
- Short Time Fourier Transform
- Wavelet Transform

The reason for this is that Fourier Transforms (FT), Short Time Fourier Transform (STFT), Wavelet Transforms (WT), etc. have been applied extensively in electric power system (Ekici *et al.,* 2006; Borghetti *et al.,* 2006; Yilmaz *et al.,* 2007).

3.2.2 Fourier Transform

Historically, the earliest form of function representation using orthogonal basis functions is undoubtedly the Fourier series for continuous and periodic signals. This is given as follows (Alsberg *et al.,* 1997; Jenkins, 1999; Oppenheim *et al.,* 1999; Yilmaz *et al.,* 2007; Mallat, 2009):

$$s(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk(2\pi/T)t}, \quad c_k = \frac{1}{T}\int_{T/2}^{T/2} s(t)e^{-jk(2\pi/T)t}\,dt \tag{3.1}$$

where $s(t)$ is the continuous time signal to be analyzed, T is the period of the signal, and c_k are the complex Fourier coefficients representing the spectral components of the signal.

The complex exponential functions at different discrete frequencies of $2\pi jk/T$ are not compactly supported in time since they extend to infinity. Thus, the Fourier representation is not suitable for analyzing non-stationary signals.

Fourier Transform (FT) has always been the classical transform for signal analysis. It is made up of the product of the signal of interest $s(t)$ and complex exponential denoted by $e^{-j2\pi ft}$.

where :

$$e^{-j2\pi ft} = \cos(2\pi ft) - j\sin(2\pi ft) \text{ and } j = \sqrt{-1}. \tag{3.2}$$

where *f* is the frequency and *t* is the time at a particular instant.

FT is used to decompose a continuous time signal with finite energy function into a sum of sinusoidal waves of different frequencies by relating the temporal characteristics of a signal to its frequency spectrum (Daubechies, 1992; Yilmaz *et al.,* 2007; Mallat, 2009).

The main drawback of FT is that the time information of a particular event is lost during transformation to the frequency domain (Alsberg *et al.,* 1997; Sodagar, 2000; Mallat, 2009). This is depicted in the Figure 3.1.

From Figure 3.1, the only information that can be extracted is the various frequencies present in the signal. The figure gives poor localization in time but better frequency resolution. Thus, the time at which these frequencies appear is unknown.

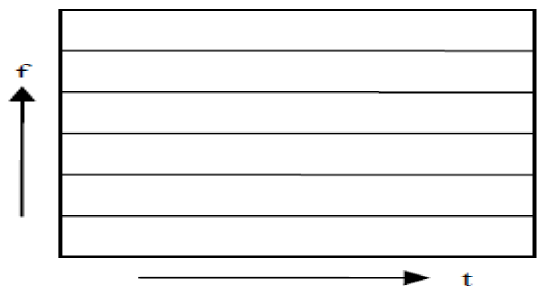


Figure 3.1: Visualization of the Time-Frequency Representation using Fourier Transform (Sodagar, 2000)

For non-stationary signal, the most interesting signals contain numerous non-stationary or transitory characteristics with frequency changing in time. Thus, FT is not suitable for the analysis of high frequency and non-periodic signals like power system transient phenomena but it is adequate for stationary signals, where every frequency components occur in all time.

3.2.3 Short Time Fourier Transform

Short-Time Fourier Transform (STFT) is a signal processing technique similar to FT. It provides both time and frequency information about a signal by mapping the signal into a two-dimensional function of time and frequency compared to the one-dimensional function of FT. This it does by dividing the signal into segments which are assumed to be stationary by using a time-localized window function which is shifted and multiplied with the signal to obtain stationary signals. FT is applied afterwards on these small segments to get the STFT.

The equation for STFT is given as (Alsberg *et al.,* 1997; Sodagar, 2000; Mallat, 2009; Saha *et al.,* 2010):

$$S_W(j\omega,\tau)=\int_{-\infty}^{+\infty} s(t)w(t-\tau)\, e^{-j\omega t}\, dt \tag{3.3}$$

where $S(t)$ is the signal of interest, $w(t)$ is the windowing function, ω is the angular frequency, and τ is the translation time.
It can be inferred from equation (3.3) that STFT is the convolution of the signal $S(t)$ with the windowing function $w(t)$.

In implementing STFT, the signal is divided into small segments and each segment of the signal is assumed to be stationary. Fourier transform is then performed on each of the segments while sliding the window along the signal thereby giving the time and frequency information of the signal components. As a result of this, the frequency spectrum of every window of the signal is known. This makes it possible to locate the time position of the shorter frequency components in the signal (Mallat, 2009).

To get better resolution in the time or frequency domains, the parameters of the windows can be modified by using a narrow window to obtain good time resolution or a wide window to get good frequency resolution. The resolution also depends on the type of window function. The window function can be a Hanning, Hamming, Blackman, or a Gaussian window. However, Gaussian window is considered to be more superior to rectangular windowing (Alsberg *et al.,* 1997). Figure 3.2 gives a plot of the above mentioned windows (Oppenheim *et al.,* 1999).

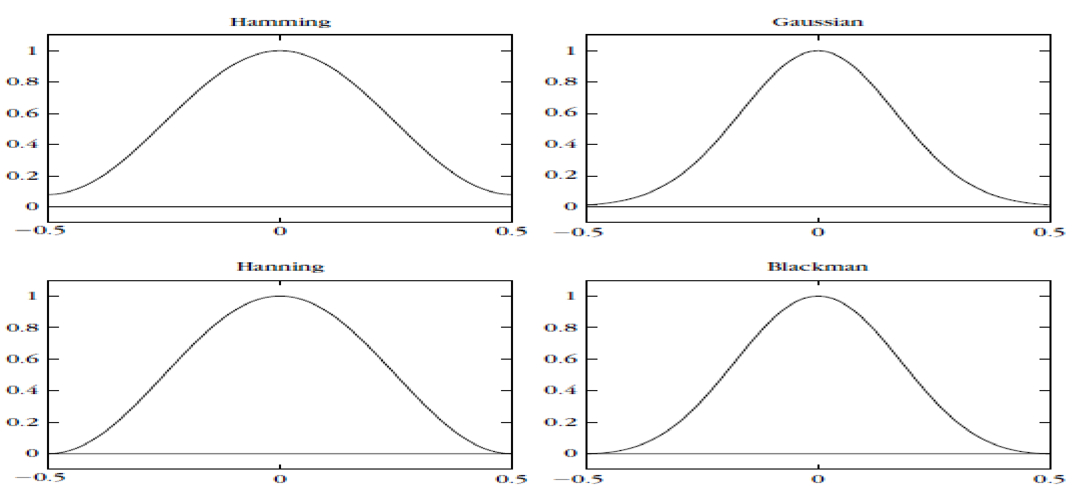


Figure 3.2: **Window functions for STFT (Oppenheim *et al.,* 1999)**

Although, STFT is an improvement over FT, it uses a window of fixed length for all frequencies. This is as depicted in Figures 3.3 and 3.4 (Oppenheim *et al.,* 1999; Sodager, 2000). The figures show that STFT uses fixed time resolution over all frequencies and fixed frequency resolution at all times. This implies that the time-frequency resolution in STFT is constant and independent at all times.

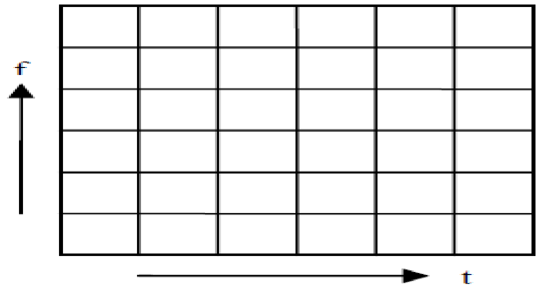


Figure 3.3: **Visualization of the Time-Frequency Representation using Short Time Fourier Transform (Sodagar, 2000)**

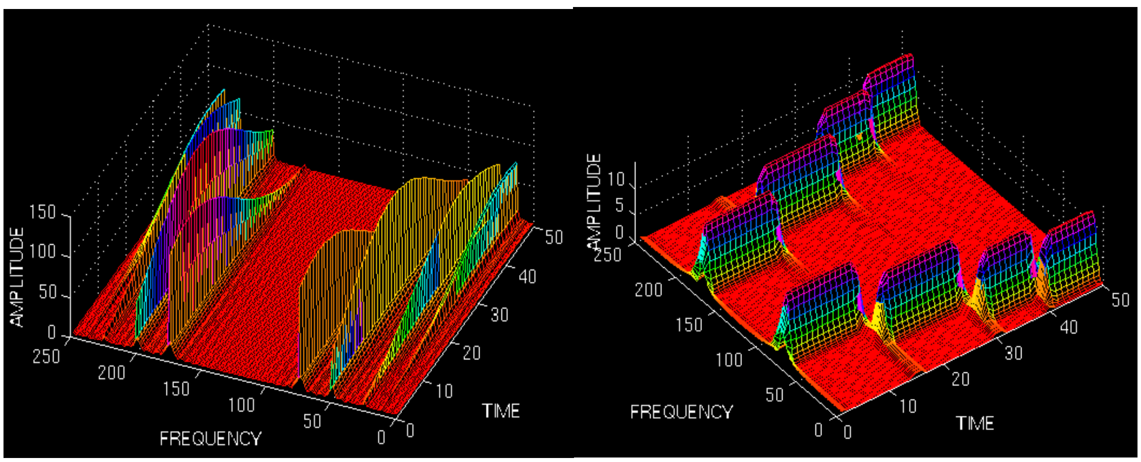


Figure 3.4: **Short Time Fourier Transform illustrating the use of narrow and wide windows respectively (Oppenheim *et al.,* 1999)**

3.2.4 Wavelet Transform

Active research on the use of the Wavelet Transform (WT) began in the early 1980s and has witnessed rapid development thereafter. Research works by Stromberg (1980), Morlet (1983), Grossman (1984), Meyer (1986), Mallat (1987), and Daubechies (1988), did have tremendous impact on the wavelet theory. WT finds a major application in the fields of speech and image processing. The variable time-frequency localization features of WT makes it very advantageous than STFT when analyzing non-stationary signals. WT analyses a signal by decomposing the signal into its frequency components before analyzing them in time. Multi-Resolution Analysis (MRA) is used in WT because most signals (for example, the human voice and power system transients), have their most important features in the high frequency range. WT uses narrow windows at high frequencies and wide windows at low frequencies by changing the width of the wavelet function along the frequency spectrum.

Although, WT was advocated by Jean Morlet, the mathematical theory leading up to WT can be accredited to the various works by Joseph Fourier on frequency analysis. Stephane Mallat in 1985 described the relationship between filter-based signal compression methods and orthogonal wavelets. Similarly, the research by Alex Grossmann (1984), Yves Meyer (1986), and Ingrid Daubechies (1988), on the discrete form of wavelets took WT analysis to a greater level. Figure 3.5 shows a representation of a signal in the time-frequency domain (Sodager, 2000).

Wavelets are functions used to decompose a time domain signal into different scales. Each scale component is a signal that covers a specific frequency band. Therefore, the time information of any specific frequency is not lost and signal representation will be localized in time as well as frequency. The scale components are then studied with a matching resolution.

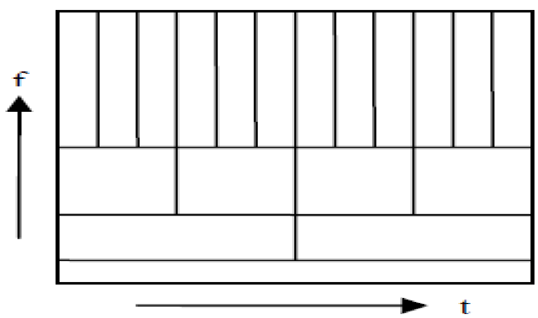


Figure 3.5: **Visualization of the Time-Frequency relationship using DWT (Sodagar, 2000)**

Wavelets are defined as short localized waves or mathematical functions used in dividing a continuous signal into different scale components often referred to as coefficients (Tang, 2009). Wavelet transform involves representing a given function with wavelets. These wavelets are also referred to as daughter wavelets and are obtained by scaling and shifting a finite length waveform known as the mother wavelet (Mallat, 2009). Wavelets can be defined using a scaling filter or by using the scaling and wavelet functions (Daubechies, 1992; Mallat, 2009). Daubechies and Symlet wavelets are defined using scaling filters comprising of low pass filters g_o and high pass filters h_o.

Similarly, Meyer wavelets are often defined using the scaling $\varphi(t)$ and wavelet $\psi(t)$ functions. These functions are the wavelet function (mother wavelet) and the scaling function (father wavelet). In filter terminologies, the scaling function corresponds to the low pass filters, while the wavelet function relates to high pass filters.

Research on wavelet began with the derivation of a constant function $\psi(t)$ for wavelet analysis by Haar in1910.

Given a constant function $\psi(t)$ (Mallat, 2009):

$$\psi(t)=\begin{cases} 1 & \text{if } 0 \le t < 1/2 \\ -1 & \text{if } 1/2 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The scaling and shifting $\psi(t)$ of the constant function in equation (3.4) results to an orthonormal basis:

$$\left\{\psi_{j,n}(t)=\frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\} \tag{3.5}$$

where *j* and *n* are the time scale parameters.

The energy of a function $s(t)$ associated with the space $L^2(\Re)$ is:

$$\|s\|^2 = \int_{-\infty}^{+\infty}|s(t)|^2 dt < +\infty \tag{3.6}$$

If the inner product in the $L^2(\Re)$ space is $\langle s,g\rangle=\int_{-\infty}^{+\infty}s(t)\,g^*(t)\,dt$, the signal can be represented by the inner product coefficient of its wavelet.

A signal s can then be substituted by the coefficients of its wavelets.

$$\langle s,\psi_{j,n}\rangle = \int_{-\infty}^{+\infty}s(t)\,\psi_{j,n}(t)\,dt \tag{3.7}$$

Equation (3.7) can be recovered by summing them in the orthonormal basis.

$$s = \sum_{j=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}\langle f,\psi_{j,n}\rangle\psi_{j,n} \tag{3.8}$$

The admissibility criteria are that the mother wavelet must have a zero mean and a square norm/finite energy (Mallat, 2009). This is as given in equations (3.9) and (3.10).

$$\int_{-\infty}^{\infty}\psi(t)\,dt=0 \tag{3.9}$$

$$\int_{-\infty}^{\infty}|\psi(t)|^2 dt=1 \tag{3.10}$$

Equations (3.9)-(3.10) give the conditions for zero mean and square norm one respectively.

The daughter wavelet $\psi_{a,b}(t)$ derived from the scaling by a factor of a and shifting by a factor of b of the mother wavelet is given by (Daubechies, 1992; Mallat, 2009):

$$\psi_{a,b}(t)=\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3.11}$$

The scaling of wavelet can be said to be the act of stretching or compressing that wavelet. Low frequencies correspond to general information of the signal; whereas high frequencies relate to latent information in the signal.

A good introductory overview on wavelet transform and history is given in Daubechies, (1992) and Mallat, (2009).

3.2.4.1 Continuous Wavelet Transform

The Continuous Wavelet Transform (CWT) of a signal $s(t)$ is the integral of the product between *s(t)* and the daughter-wavelets, which are time translated and scale dilated/compressed versions of the mother-wavelet. This process, equivalent to a scalar product, produces wavelet coefficients $C(a,b)$ which closely resemble the signal and the daughter-wavelet located at position ***b*** (shifting factor) as it is shifted through the signal, ***a*** is the scale parameter.

From equation (3.11) above, the CWT of a signal $s(t)$ is defined as (Borgetti, 2006; Mallat, 2009):

$$CWT(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} s(t)\psi\left(\frac{t-b}{a}\right)dt \tag{3.12}$$

where *C(a,b)* are the wavelet coefficients, $\psi(t)$ is the mother wavelet, a is the scale factor, b is the translation factor. $|a|^{-1/2}$ is the normalization value of $\psi_{a,b}(t)$ so that if $\psi(t)$ has a unit length, then its scaled version $\psi_{a,b}(t)$ would by induction also have a unit length.

When the scale parameter ***a*** is changed, the centre frequency and bandwidth of ***ψ(t)*** changes. Large scales (low frequencies) dilate the signal and provide detailed information of hidden short lasting components in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal (Mallat, 2009). CWT makes use of mother-wavelets which must satisfy the ‘admissibility condition’ given in (3.9) and (3.10) (Mallat, 2009):

CWT has the ability to operate at any scale, especially from that of the original signal up to some maximum scale. It is also continuous in terms of shifting. During computation, the analyzing wavelet is moved over the full length of that function (Mallat, 2009).

Unlike Fourier transform, the continuous wavelet transform possesses the ability to construct a time-frequency representation of a signal that offers very good time and frequency localization and is an excellent tool for mapping the changing properties of non-stationary signals.

3.2.4.2 Discrete Wavelet Transform

Discrete Wavelet Transform (DWT) is a versatile signal processing tool that finds many engineering and scientific applications. It is capable of capturing non-stationary signals and localizing them in both the time and frequency domain accurately.

DWT is similar to CWT. The time-frequency scale is sampled and the scale and shift parameters are discretized on a logarithmic grid of 2 (Daubechies, 1992; Mallat, 2009). The Discrete Wavelet Transform and Multi Resolution Analysis (MRA) use a narrow window for high frequency signal components and a wide window for low frequency signal components. Hence, providing a very good resolution in the time-frequency domain.

The discrete wavelet transform analyzes the signal by decomposing the transient signal with short scale of window for high frequency band while with long window scale for low frequency band using scale and shift technique.

$$DWT(m,n) = \frac{1}{\sqrt{a_0^{\ m}}} \sum_k f(k) \psi \left(\frac{n - k\, b_0 a_o^{\ m}}{a_0^{\ m}} \right) \qquad (3.13)$$

where $f(k)$ is a discrete signal, $\psi(n)$ is the mother wavelet (window function), m and n are time scale parameters, k is the number of coefficients, a_0^m is the variable for scale, $k\,b_0 a_0^m$ is the variable for shift, and $1/a_0^m$ is the energy normalization component to ensure the same scale as the mother wavelet.

The most widely used form of such discretization with a_0 = 2, and b_0 = 1 on a dyadic time-scale grid gives (Daubechies, 1992; Mallat, 2009):

$$DWT(m,n) = \frac{1}{\sqrt{2^m}} \sum_k f(k) \psi \left(\frac{n - k\, 2^m}{2^m} \right) \qquad (3.14)$$

After filtering the signal with low pass and high pass filters, the resulting signal is downsampled. Down-sampling is a signal processing technique for modifying the sampling rate of a digital signal and compressing the information. The decomposition of the signal into different frequency bands is as shown in Figure 3.6.

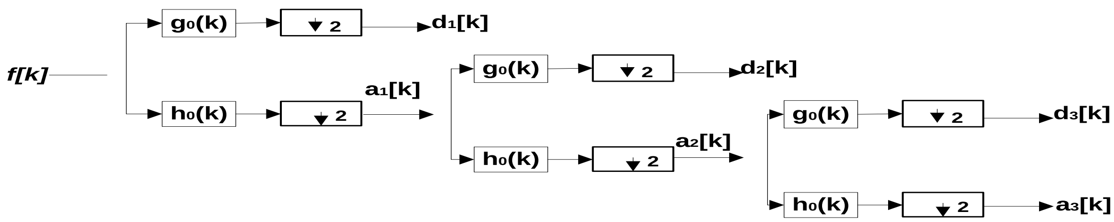


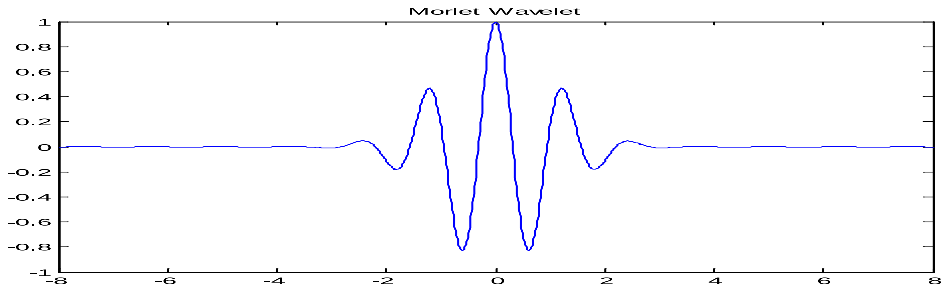
Figure 3.6: **Multi-level wavelet decomposition tree**

where g_o is a high pass filter, h_o is a low pass filter, $d_1[k]$, $d_2[k]$, $d_3[k]$ are the detail coefficients at levels-1, 2, and 3. $a_1[k]$, $a_2[k]$, $a_3[k]$, are the approximation coefficients at levels-1, 2, and 3 respectively.

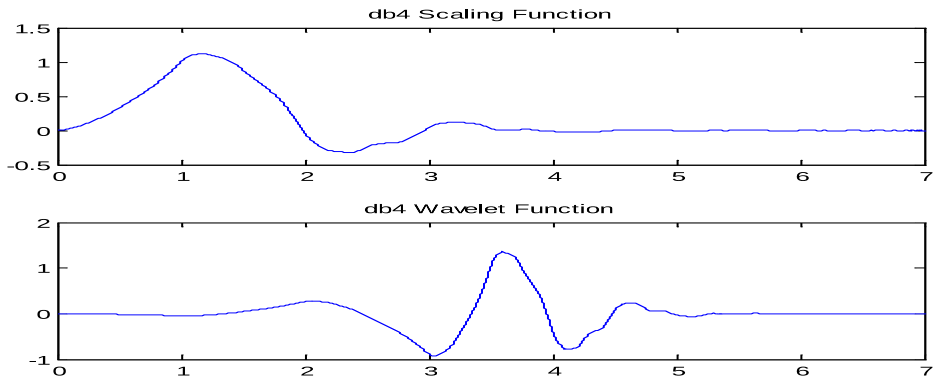
Figure 3.6 illustrates the implementation of DWT. The signal *f(k)* is decomposed into detail and approximation coefficients. The frequency band of $d_1[k]$ component is *fs/2 - fs/4* Hz, and $a_1[k]$ component is *fs/4 - 0* Hz. In the second decomposition level, the $a_1[k]$ component derived from the first decomposition level is further decomposed into $d_2[k]$ component of high frequency band and $a_2[k]$ component of low frequency band. The frequency band of $d_2[k]$ component is *fs/4–fs/8* Hz and the frequency band of $a_2[k]$ component is *fs/8 - 0* Hz band. *fs* is the sampling frequency.

The low pass filter is usually symmetric whereas the high pass filter can be symmetric or asymmetric. As the signal of the desired component can be extracted via repetitious decomposition, the number of decomposition steps should be decided by comparing the scale of sampling frequency with that of the frequency component of desired signal.

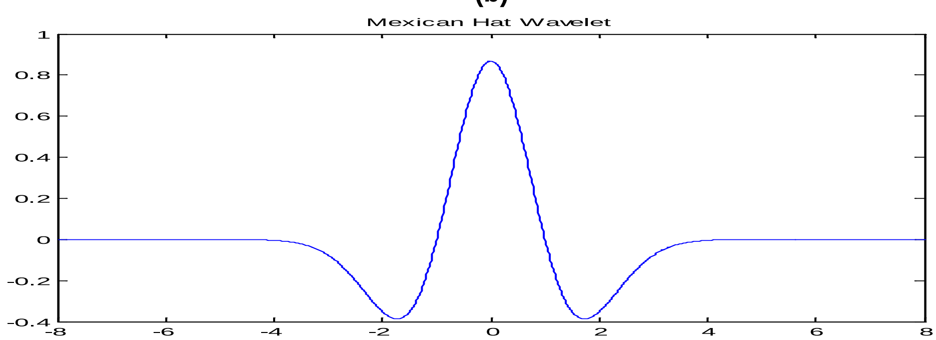
Examples of wavelet families include: Coiflet, Haar, Symlet, Daubechies, Meyer, Morlet, Mexican Hat, etc. Within each family of wavelets, there are wavelet subclasses. These subclasses are distinguished by the number of filter coefficients and the level of iteration. Figure 3.7 gives examples of some wavelet and scaling functions for Morlet, Daubechies, and Mexican hat wavelet families.



(a)



(b)



(c)

Figure 3.7: a. Wavelet function for Morlet mother wavelet; **b.** Scaling and wavelet function for Daubechies db-4 level-10; and c. Mexican hat function

3.3 Artificial Neural Network Theory

3.3.1 Introduction

Artificial Neural Networks (ANNs) are information processing paradigms structured after biological nervous systems, such as the human brain. They are made up of highly interconnected neurons operating in parallel (Nilsson, 1998; Jones, 2008). These neurons are capable of mapping a set of input patterns onto a corresponding set of output patterns. The ability to do this is by adjusting the values of the various interconnections (weights) between the elements. ANNs are commonly used in modelling, signal processing, pattern recognition, time series analysis, control systems, risk analysis, etc.

3.3.2 Historical Background of Neural Network

The history of neural networks can be traced back to the work by McCulloch and Pitt in 1943 where they described a formal calculus of networks involving simple computing elements. These basic ideas that were developed by them form the basis of artificial neural networks. In 1949, Donald Hebb developed the ‘Hebbian learning rule’ for self organized learning. He found that if two connected neurons were simultaneously active, then the connection between them is proportionately strengthened. In other words, the more frequently particular neurons are activated, the greater the weight between them (i.e., learning by weight adjustment). In 1958, Rosenblatt devised the perceptron model which was able to solve pattern classification problems through supervised learning. Another NN type was the ADAptive LInear Element (ADALINE) developed by Widrow and Hoff. In contrary, Minsky and Papert in 1969 provided mathematical proofs of the limitations of the single layer perceptron to multilayered systems and pointed out its weaknesses in computation.

Major neural network paradigms in recent years include the Kohonen's network in 1972, the first back-propagation algorithm developed by Werbos in 1974, and Hopfield network in 1982. Hopfield used the idea of energy function to formulate a new way of understanding the computation performed by recurrent networks with symmetric synaptic connections. He developed a new class of neural networks with feedback, which are well known as Hopfield Networks. Another important development in 1982 was made by Kohonen. He developed a self-organizing map using one or two lattice structure.

In 1986, Rumelhart, Hinton and Williams developed a learning algorithm called back-propagation algorithm. Back-propagation algorithm was afterwards modified by many researchers to increase the speed of training. Broomhead and Lowe, in 1988, described a procedure for designing feed-forward networks using radial basis functions, which provides an alternative to multilayer perceptrons.

3.3.3 Advantages of Neural Network

The following is a list of some of the advantages of ANNs:

a. Adaptivity: ANNs have the ability to adapt their synaptic weights to reflect changes. Also, a network can be easily retrained to adjust to some minor changes in the operating environment.
b. Robustness: Neural networks have demonstrated the ability to perform even when the inputs are degraded or noisy.
c. Flexibility: NN can be applied to several types of problems.
d. Generalization: ANNs when properly trained can provide the correct response to untrained input patterns that share similarity with the training patterns.

3.3.4 Biological Neuron

A neuron is a biological cell that processes information. The essential compositions include soma, axons, and dendrites. There are approximately 10^{11} neurons in the human brain with approximately 10^{14}-10^{15} interconnections (Hagan *et al.,* 1996; Jain *et al.,* 1996; Choppin, 2005).

Biological neuron receives signals from others through the dendrites. Dendrites are tree-like extensions of a neuron. They control the electrochemical excitation received from other neurons to the soma from which the dendrites project. Electrical activity is sent out via the axon which closely resembles a long, thin strand. At the end of each strand, is the synapse. A synapse is the point of contact between the axon of one cell and the dendrite of another cell. The synapse converts the activity from the axon into electrical effects that inhibit or fires (excites) the connected neurons. This triggers the release of neurotransmitter substances at the synapse. Thus, the brain performs extremely complex tasks by computing the weighted sum of the inputs and then firing a binary signal if the total input exceeds a certain threshold.

A typical biological nerve cell is as shown in Figure 3.8 below (Mehrotra *et al.,* 1996).

.

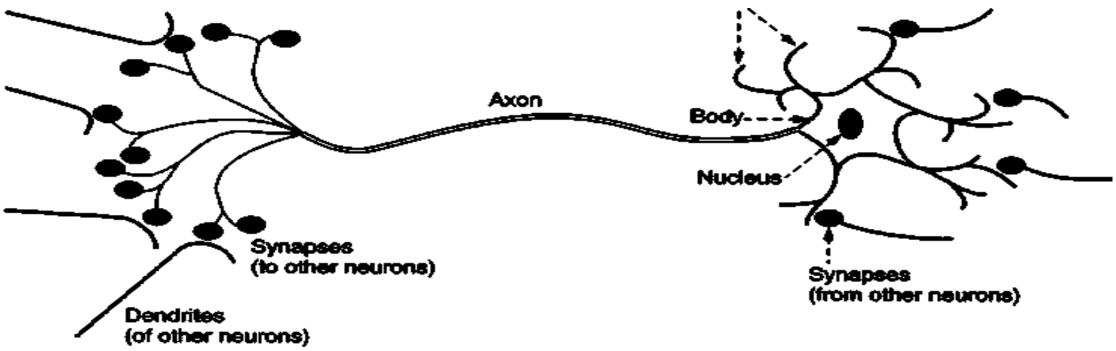


Figure 3.8: **Biological Neuron (Mehrotra *et al.,* 1996)**

3.3.5 Artificial Neuron

The inter-neuron arrangements and the nature of the connections determine the structure of a network. The processing elements each have a number of internal parameters called weights. Changing the weights of an element will alter the behaviour of the element and, therefore will also alter the behaviour of the whole network (Hagan *et al.,* 1996; Pham & Liu, 1996; Haykin; 1999, Choppin, 2005).

To make the ANN perform a particular task, the weights of the network are chosen to achieve the desired input/output relationship during a process known as training. The learning algorithm governs the process of choosing the weights (strengths) of the connections and adjusting/training it to achieve the desired overall behaviour of the network.

NN layers consist of neurons which are connected to other neurons through synaptic weights and biases. It is by adjusting the weights and biases of these connections that training is effected. ANNs are typically defined by three types of parameters:

1. The interconnection pattern between different layers of neurons.
2. The learning process for updating the weights of the interconnections.
3. The activation function that converts a neuron's weighted input to its output activation.

A neuron k with input U_k is represented by (Veelenturf, 1995; Hagan *et al.,* 1996; Choppin, 2005; Jones, 2008):

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \qquad 3.15$$

Given an input vector $x_1,\ldots,x_m$ the output y_k is obtained as:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right) \tag{3.16}$$

$$y_k = \varphi(u_k + b_k) \tag{3.17}$$

where weight w_{kj} connects the *kth* neuron with the *jth* input x_j, and $\varphi(.)$ is the activation function.

$w_{k1}, w_{k2}, \ldots, w_{kj}$ are the synaptic weights of the *kth* neuron respectively, and y_k is the output signal of the neuron. The bias b_k increases or lowers the net input of the activation function depending on whether it is positive or negative. A simple nonlinear model of a neuron is as shown in Figure 3.9.

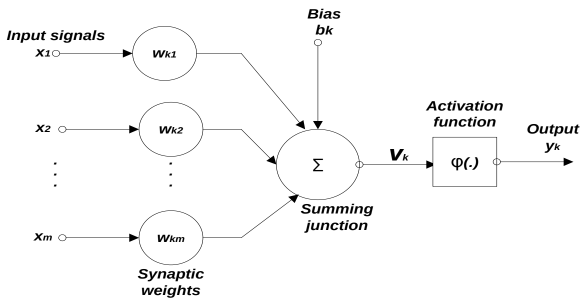


Figure 3.9: **Nonlinear model of a neuron**

3.4 Types of Neural Network

Neural networks can generally be classified according to their network topology. These are: Feed-Forward Neural Network (FFNN) and Feed-back Neural Network (FBNN).

3.4.1 Feed-Forward Neural Networks

In Feed-Forward Neural Networks (FFNNs), information moves only in the forward direction from the input layers, via the hidden layers (if any) and to the output layers. Feed-forward networks as shown in Figure 3.10 allow signals to travel in one direction from the input to the output. Thus, a layer is not affected by its own output. The FFNNs were the first and most widely used models in many practical situations. They can match the input vectors and the output vectors. The output at any particular time is dependent on only the corresponding input. They are the most popular and most widely used models in many practical situations. This is because FFNNs are easily to train and they produce a response to an input quickly.

Examples of feed-forward networks include the single-layer perceptron, ADALINE, multi-layer perceptron, radial basis function, learning vector quantization network, probabilistic neural network, generalized regression neural network, etc.

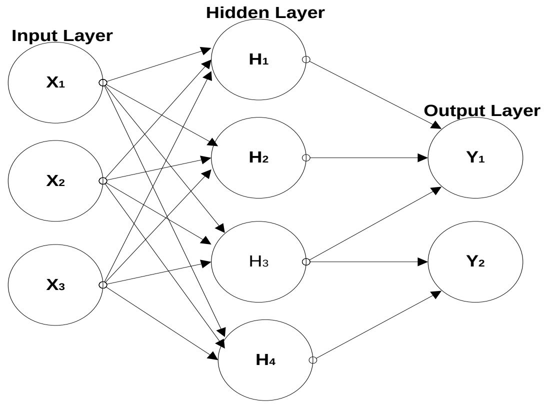


Figure 3.10: **Feed-Forward Neural Network topology**

3.4.2 Feed-Back Neural Networks

In the Feed-Back Neural Networks (FBNNs), there are feedback connections from one layer to another. That is, there is a bi-directional data flow and data are also propagated from the outputs to the input layers. These types of networks have a dynamic memory in that any particular output is a function of the current input and past inputs and outputs. FBNNs are best suited for optimization problems where the neural network looks for the best arrangement of interconnected factors. They are also used for error-correction and partial-contents memories where the stored patterns correspond to the local minima of the energy function.

Examples of feed-back networks include Boltzman machine, Elman networks, recurrent network, etc.

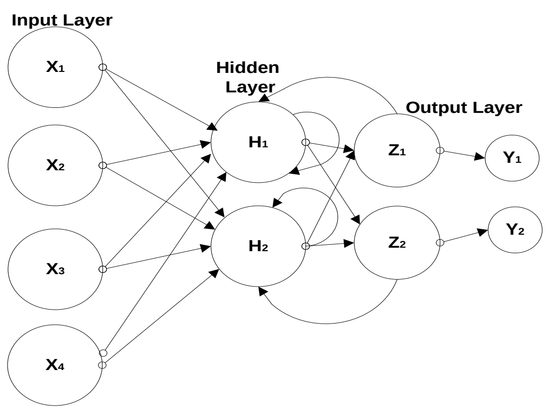


Figure 3.11: **Recurrent Neural Network topology**

3.5 Neural Network Architecture

3.5.1 Single Layer Perceptrons

The Rosenblatt perceptron was built around the McCulloch-Pitt model of a neuron (Hagan *et al.,* 1996). The Single Layer Perceptrons (SLPs) are suitable for simple linear separable or linear discriminants problem for pattern classification into one or two classes (Veelenturf, 1995; Hagan *et al.,* 1996; Choppin, 2005; Jones, 2008).The training technique used is called the perceptron learning rule and is capable of learning by generalizing from its training vectors and learning from randomly distributed connections.

The perceptron model is made up of a linear combiner and a hard limit transfer function. A high is produced if the net input is equal to or greater than 0; and 0 if otherwise. The perceptron learning rule is applied to each neuron in order to calculate the new weight and bias. Input vectors are classified by dividing the input space into two decision regions separated by a hyperplane defined by:

$$\sum_{i=1}^{m} w_i x_i + b = 0 \qquad (3.18)$$

where m is the number of input variables, $w \in \Re^m$ is the vector of the weight, $x \in \Re^m$ is the vector of the input stimulus, and b is the bias.

Perceptrons are trained on examples by using a set of inputs-output pairs where p is a vector of the input to the network and t is the corresponding correct output target vector as shown in Figure 3.12.

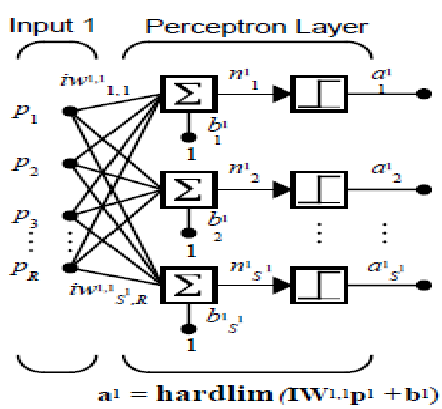


Figure 3.12: **Perceptrons (Demuth *et al.,* 2004)**

3.5.2 Multilayer Perceptrons

The multilayer perceptrons consist of the input layer, hidden layer, and an output layer. The input layer and hidden layer are referred to as source nodes, while the output layers are regarded as computational nodes. The input layer propagates signals through the network in a forward direction from layer to layer.

Multilayer perceptrons have been reported in the literature to be successful in complex problem application through supervised training based on the back-propagation learning algorithm (Veelenturf, 1995; Hagan *et al.,* 1996; Haykin, 1999; Choppin, 2005).

A typical example of the multilayer perceptron is Figure 3.13. From Figure 3.13, it is seen that an input signal propagates forward through the network and emerges at the output end. Also, an error signal is computed at the output of the network and is propagated backward through the network.

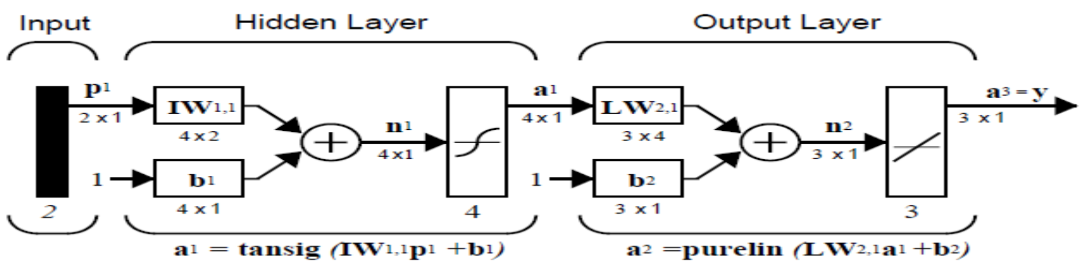


Figure 3.13: **Multi Layer Perceptron (Demuth *et al.,* 2004)**

This forms the basis of the error back-propagation algorithm. The back-propagation learning rule is implemented by adjusting the weights and biases of networks in order to minimize the Sum Squared Error (SSE) of the network. With this, the value of the network weights and biases are continuously changed in the direction of steepest descent with respect to the error.

3.5.3 Radial Basis Neural Network

A Radial Basis Function (RBF) network is an artificial neural network that uses radial basis functions as activation functions. It is made up of a feed-forward structure consisting of two layers: The input node, a nonlinear hidden layer with a radial basis function units and a linear output layer.

The radial basis transfer function for the hidden neurons can be spline, multiquadratic, or Gaussian function. However, the Gaussian RBF is the most commonly used. RBFNs find application in function approximation, time series prediction, and control problems.

Three parameters are required in a RBF network. These are: center vectors c_i, the output weights w_i, and the RBF width parameters β_i. In the sequential training, the weights are updated at each time step as the data are fed in. Figure 3.14 shows a *radbas* neuron. The activation type used in MATLAB is the *radbas*.

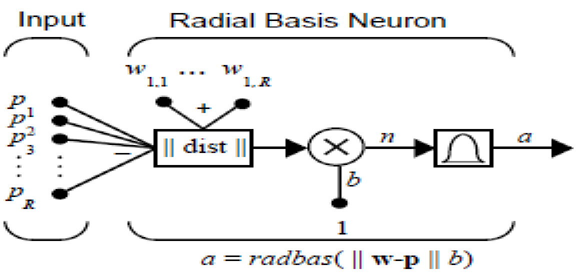


Figure 3.14: Radial basis neuron (Demuth *et al.,* 2004)

The output of the *jth* hidden layer neuron to an input P_R is (Hagan *et al.,* 1996):

$$\phi_j(x_k) = \exp\left(-\frac{1}{\sigma_j^2}\left\|P_R - \mu_j\right\|^2\right) \tag{3.19}$$

where σ_j is the spread of the Gaussian function, μ_j is the centre of the neuron and $\| \|$ is the Euclidean norm.

The response at the output layer is:

$$f_i(P) = \sum_{j=1}^{n_k} \phi_j\left(\left\|P_R - \mu_j\right\|\right)\theta_{ji} \tag{3.20}$$

The input is denoted by x_R, while θ_{ji} is the weight of the *jth* hidden layer to the *ith* output.

Two variants of radial basis networks are possible. Details of these networks are provided below.

3.5.4 Generalized Regression Neural Networks

This type of NN finds application in function approximation or regression problems. The GRNN is made up of radial basis and linear layers respectively. This is depicted in Figure 3.15 below.

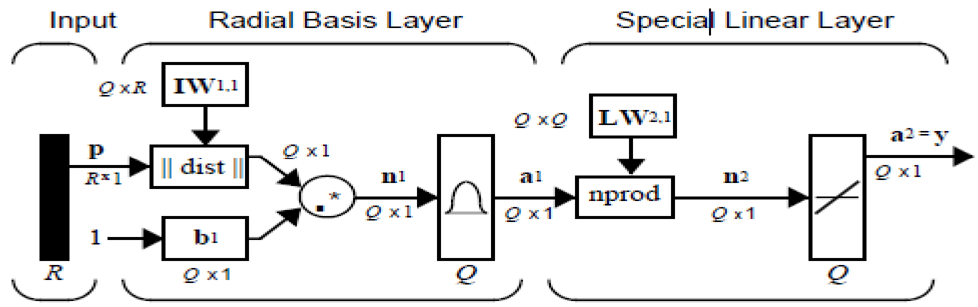


Figure 3.15: **GRNN architecture (Demuth *et al.,* 2004)**

The function '*newgrnn*' is used to create a GRNN in the MATLAB Neural Network Toolbox.

3.5.5 Probabilistic Neural Networks

Probabilistic Neural Networks (PNNs) are feed-forward neural networks which was derived from Bayesian network and the Kernel-Fisher discriminant analysis (Demuth & Beale, 2002). PNNs are similar to other feed-forward neural network. The main difference is the way that learning occurs. PNNs do not have weights in their hidden layer. Instead, the hidden node is used to represent an example vector.

PNN consists of an input layer with feature vectors interconnected with the hidden layer consisting of example vectors, and an output layer representing the possible classes. They are used in classification problems. A PNN is guaranteed to converge to a Bayesian classifier and their design does not depend on training (Hagan *et al.,* 1996). Figure 3.16 illustrates the PNN.

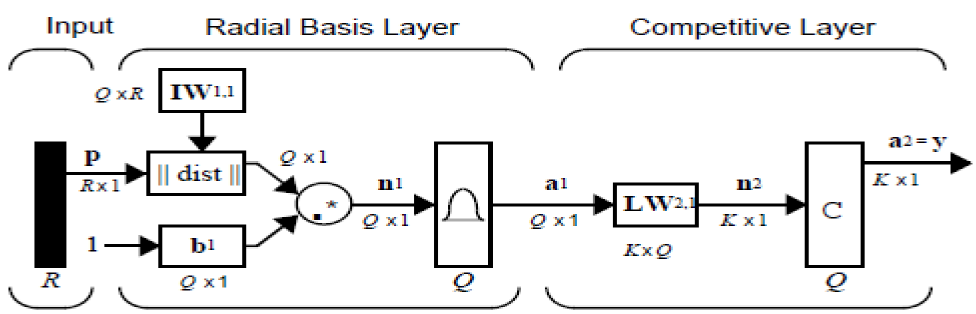


Figure 3.16: **Probabilistic Neural Network (Demuth *et al.,* 2004)**

3.5.6 Self-Organizing Networks

Self-organizing networks learn to detect similarities in their input and adjust their future responses to corresponding inputs.

3.5.6.1 Competitive Learning

Competitive network categorizes input vectors presented to it by distributing the neurons in the competitive layer to recognize frequently presented input vectors.

Figure 3.17 illustrate the structure of a competitive network.

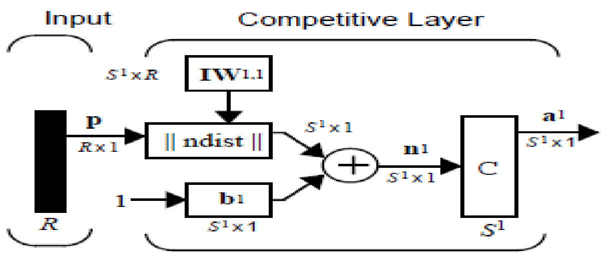
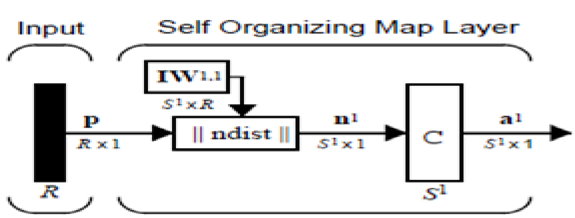


Figure 3.17: **Competitive Learning (Demuth *et al.,* 2004)**

3.5.6.2 Self-Organizing Maps

The Self-Organizing Feature Mapping (SOFM) is used to map unsupervised input vectors into a two dimensional space where the vectors are self-organized into clusters that represent the different classes. However, it is not recommended to be used by itself for pattern classification, but as a front-end to a classifier that employs supervised learning (Hagan *et al.,* 1996).

SOMs are typically used for classification problems, where the output neurons represent groups that the input neurons are to be classified into and they are usually trained with a competitive learning strategy. Figure 3.18 illustrates a typical architecture of the SOM.



$$n_i^1 = -\| {}_i\mathbf{IW}^{1,1} - \mathbf{p} \|$$

$$\mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1)$$

Figure 3.18: **Self Organizing Maps (Demuth *et al.,* 2004)**

3.5.7 Learning Vector Quantization Networks

Learning Vector Quantization (LVQ) uses supervised learning method. In LVQ networks, the hidden layer is replaced by a Kohonen layer. Classification using LVQ networks are done as specified by the trainer. Figure 3.19 illustrates a typical architecture of the LVQ.

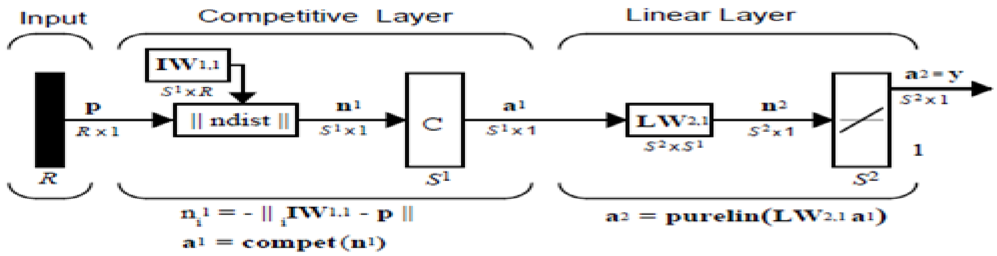


Figure 3.19: **Learning Vector Quantization (Demuth *et al.,* 2004)**

3.5.8 Recurrent Networks

3.5.8.1 Elman Networks

Elman networks are back-propagation networks with two layers. The only difference being the added feedback connection from the output of the hidden layer to its input. Elman networks can be created with the function '*newelm*', while Hopfield networks can be created with the function '*newhop*' in MATLAB. Figure 3.20 shows an Elman network, where *D* is the value of the delay in the feedback from the output of the hidden layer to its input.

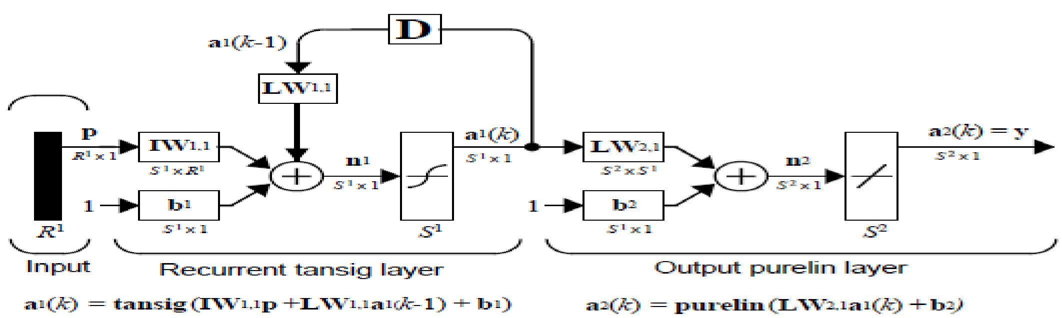


Figure 3.20: **Elman Network (Demuth *et al.,* 2004)**

3.5.8.2 Hopfield Network

The Hopfield network works by storing equilibrium points onto which the network revolve to. There is a feedback loop whereby the output is fed back to the input. This is as shown in Figure 3.21. The network stores target vectors and uses them as cues when provided with similar vectors.

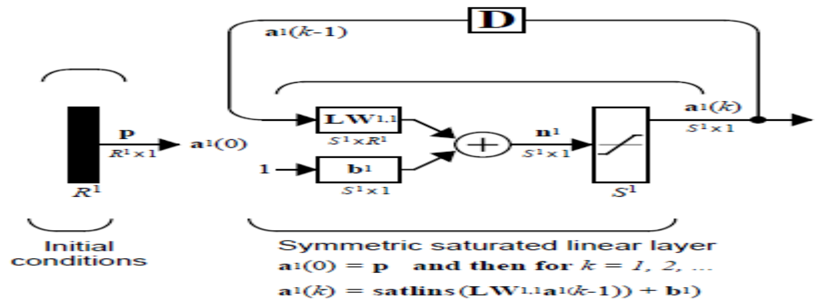


Figure 3.21: Hopfield Network (Demuth *et al.,* 2004)

3.6 Neural Network Training

The most significant property of a neural network is that it can learn, and can improve its performance through learning. Learning is a process by which some parameters of a neural network i.e. synaptic weights and thresholds are adapted through a continuous process of stimulation in the environment in which the network is embedded. The network becomes more knowledgeable about the task after each iteration. The objective of the training process is to adjust the ANN weights and biases to obtain minimal deviations between the target and the calculated ANN outputs in relation to the average of all input samples.

There are three types of learning paradigms namely, supervised learning, self-organized or unsupervised learning, and reinforced learning (Hagan *et al.,* 1995). The supervised and unsupervised learning are sometimes referred to as classification and clustering tasks respectively (Webb, 2002; Dunne, 2007).

3.6.1 Supervised Learning

In supervised learning, an external teacher, having the knowledge of the environment represents a set of input-output examples for the neural network which may not have any prior knowledge about that environment. When the teacher and the neural network are both exposed to a training vector drawn from the environment, by virtue of built-in knowledge, the teacher is able to provide the neural network with a desired response for that training vector. The network adjusts its weights and thresholds until the actual response of the network is very close to the desired response. The supervised learning requires a teacher or a supervisor to provide desired or target output signals. The difference (error) can then be used to change the network parameters, which results in an improvement in performance.

Examples of supervised learning algorithms include the perceptron learning algorithm, delta rule, the generalized delta rule or back-propagation algorithm, and the learning vector quantization algorithm.

3.6.2 Unsupervised Learning

Unsupervised learning is carried out by training vectors with similar properties to produce the same output. The input vectors automatically adjust the weights during training such that input vectors with the similar properties are clustered together.

Unsupervised learning includes Kohonen self-organizing maps, k-Means clustering algorithm, adaptive resonance theory, competitive learning algorithms, etc.

3.6.3 Reinforced Learning

Reinforcement learning is a hybrid form of the supervised and unsupervised learning. In this, the learning is by interaction whereby an action is performed on the environment and is reinforced by the response it receives from it. The training of ANN is done in order to associate certain output responses to particular input pattern thereby giving the ANN the ability to generalize.

Reinforcement learning system consists of three elements. These are:

- Learning element
- Knowledge base
- Performance element

A ‘critic’ is used instead of a teacher, which produces heuristic reinforced signal for the learning element. The input vector goes to the critic, learning element, and performance element at the same time. With the input vector and primary reinforced signal from the environment as inputs, the critic (predictor) estimates the evaluation function. Genetic algorithm is an instance of the reinforcement learning algorithm.

3.6.4 Learning Rules

Learning rules refer to a procedure for adjusting the weights and biases of neural networks with the aim of minimizing the discrepancies between the network outputs and the training targets (Hagan *et al.,* 1996; Dunne, 2008).

NN training involves adjusting the weights and biases of the network. In incremental training, the network elements (weights) are updated after each training vector has been presented. Whereas, batch training, involves updating the weights and biases after all of the input and target vectors have been presented to the network.

There are different kinds of learning algorithm. For example, the perceptron learning rule, back-propagation algorithm, Delta-Bar-Delta rule, conjugate gradient descent algorithm, quick propagation algorithm, Quasi-Newton algorithm, Levenberg-Marquardt algorithm, and Kohonen training. Some other learning algorithms are error correction learning, Boltzmann learning, Hebbian learning and competitive learning. Some of these learning rules are further enunciated below.

3.6.4.1 Perceptron Learning Rule

Perceptron learning is a supervised learning algorithm in which the weights are adjusted to classify the training set. Individual inputs from the training set are applied to the perceptron, and the error (the difference between the output and target) is used to adjust the weights.

The output of the perceptron can then be derived as (Jones, 2008):

$$y_j(t) = f\left[w(t) \cdot x_j\right] \tag{3.21}$$

$$y_j(t) = f\left[w_0(t) + w_1(t)\,x_{j,1} + w_2(t)\,x_{j,2} + \ldots + w_q(t)\,x_{j,q}\right] \tag{3.22}$$

The weights for each dataset are adjusted based on the error computed using equation (3.23). This is known as the Perceptron Rule.

$$w_i(t+1) = w_i(t) + \eta(d_j - y_j(t))\,x_{j,i}\,, \qquad i = \overline{0,n} \tag{3.23}$$

where j is the dimension of the input vector, n is the number of nodes of the jth input vector, η is the learning rate, d_j is the desired output, y_j is the calculated output, and x_{ji} is the input value for the current weight w_i.

The perceptron rule is used to adjust the weight by incrementing or decreasing the weight based on the direction of the error until $d_j - y_j(t)$ is less than a pre-defined threshold or if the maximum number of iterations has been completed.

The learning rate is used to moderate how the weights are changed at each step. The error is defined as 0 if the calculated output is correct; otherwise, it is positive if the calculated output is low and negative if the calculated output is high. In this way, if the output is too high, a decrease in weight is caused for an input that received a positive value. The drawback of the perceptron rule is that it is only suitable for linearly separable samples and would fail to converge if the samples are not linearly separable.

3.6.4.2 The Delta Rule

The Delta rule was introduced by Widrow and Hoff in 1959. It is an approximate gradient descent algorithm that updates its weight by minimizing the Mean-Square Error (Hagan *et al.,* 1996; Jones, 2008). Another name by which it is also known is the Widrow-Hoff rule or the Least Mean Square (LMS) algorithm. The iteration of the algorithm continues until it is capable of correctly classifying the training set.

In the Delta rule, the local minimum of the error is found by adjusting the weights proportionally to the negative of the gradient. The weights are also adjusted with a learning rate α to avoid oscillating around the MSE (Hagan *et al.,* 1996; Jones, 2008).
If the cost function is (Hagan *et al.,* 1996; Jones, 2008):

$$\varepsilon(w) = \frac{1}{2} e^2(q) \tag{3.24}$$

Differentiating equation (3.24) with respect to w yields:

$$\frac{\partial \varepsilon(w)}{\partial w} = e(q) \frac{\partial e(q)}{\partial w} \tag{3.25}$$

where $e(q)$ is the error measured at time q , w is the weight vector.

If $e(q) = d(q) - x^T(q) w(q)$, differentiating with respect to w gives:

$$\frac{\partial e(q)}{\partial w(q)} = -x(q) \tag{3.26}$$

Therefore:

$$\frac{\partial \varepsilon(w)}{\partial w} = -x(q) e(q) = \hat{g}(q) \tag{3.27}$$

But the steepest descent algorithm is:

$$w(q+1) = w(q) + \eta\, g(q) \tag{3.28}$$

Thus,

$$\hat{w}(q+1) = \hat{w}(q) + \eta\, x(q) e(q) \tag{3.29}$$

where η is the learning rate, $g(q)$ is the gradient vector at point $w(q)$, $x(q)$ is the input vector.

The major difference between the Delta rule and perceptron learning is that the former can operate with real-values unlike the perceptron learning that operates on binary inputs. Also, the Delta rule was implemented in the ADALINE network with a linear transfer function, while the perceptron used a hard-limiting transfer function. In addition, the perceptron rule converges to a solution which can be sensitive to noise because patterns lie close to the decision boundaries. On the other hand, Delta rule tries to move the decision boundaries away from the training patterns. Thus, they are robust against noise (Hagan *et al.,* 1996).

3.6.4.3 Back-Propagation Method

Back-propagation algorithm rule is the most frequently used algorithm for feedforward multilayer neural network. It is a supervised learning algorithm which requires a set of training data with known input and output vectors. It uses steepest gradient descent of error which propagates backwards for updating the synaptic weights and thresholds. The performance index in back-propagation is the same with that used in the Delta rule. The difference is the method used in calculating the derivatives of the MSE. In multilayer networks with nonlinear transfer function, the chain rule is used in the calculation of the MSE derivatives. The advantage of this algorithm is the simplicity of calculation during weight updates (Bishop, 1996).

This algorithm is implemented in two passes. In the forward pass, an input signal is propagated from the input layer through the network from one layer to another. During this process, the synaptic weights are unaltered and the activities of the nodes are calculated layer by layer. It should be noted that the computation for the forward pass begins at the first hidden layer by presenting it with the input vector and terminates at the output layer by computing the error signal for each neuron of this layer.

During the backward pass, the weights are adjusted according to the error-correction rule. That is, the output response is subtracted from the target. The computed error signal is afterwards fed back through the network.

The procedure for the back-propagation algorithm is summarized below (Bishop, 1996; Hagan *et al.,* 1996; Demuth *et al.,* 2004):

Step 1: Propagate the input forward through the network.

Step 2: Propagate the sensitivities (the output activations) backward through the network. The sensitivities are calculated by applying the chain rule.

Step3: Update the weights and biases using the approximate steepest descent rule. The weight update is done by multiplying the difference between the desired and calculated outputs with the input activation function. This gives the gradient of the current weight. Then, the learning rate is subtracted from the current weight.

The derivation of the back-propagation is given in Bishop (1996). It begins with the weighted sum of each unit of the network.

The error signal at the output j at iteration (q) is:

$$e_j(q) = d_j(q) - y_j(q) \tag{3.30}$$

$$\varepsilon(q) = \frac{1}{2}\sum_{j=C} e_j^2(q) \tag{3.31}$$

The weights are updated by using:

$$w(q) = w(q-1) + \eta \frac{\partial E(q)}{\partial w(q-1)} \tag{3.32}$$

where $E(q)$ is the error cost function and w is the weight vector.

The limitations of the back-propagation training algorithm are the training time, no guarantee for convergence to a global minimum, no exact rule for setting the number of hidden layers and neurons for the best performance, and instability if the learning rate is too large (Hagan *et al.,* 1996; Demuth *et al.,* 2004).

3.6.4.4 Improved Variations of the Back-Propagation Method

The back propagation algorithm suffers from slow rate of convergence as mentioned above, hence requires long training time for large network with large number of training patterns. However, some methods have been developed to overcome the slow rate of learning.

a. Newton's Method

Newton's method is based on the second order Taylor series and the requirement is that the error function ρ must be twice differentiable (Dunne, 2008).

With the Newton method, an attempt is made to locate the stationary point of the error function.

Expanding ρ to a minimum point in a Taylor series gives (Bishop, 1996; Dunne, 2008):

$$\rho(w+\Delta w) = \rho_{\min} = \rho(w) + \frac{\partial \rho^T}{\partial w}\Delta w + \frac{1}{2}\Delta w^T H \Delta w + \ldots, \tag{3.33}$$

where H is the Hessian matrix of second derivatives.

$$\langle H \rangle_{r1r2} = \left. \frac{\partial^2 \rho(w)}{\partial w_{r1} \partial w_{r2}} \right|_w \tag{3.34}$$

Taking the gradient of ρ with respect to Δw results to:

$$\frac{\partial \rho}{\partial w} + H\Delta w = 0 \tag{3.35}$$

Solving equation (3.35) for Δw gives:

$$\Delta w = -\frac{\partial \rho}{\partial w} H^{-1} \tag{3.36}$$

If $g = \frac{\partial \rho}{\partial w}$,

Hence, Newton's method is defined as:

$$w^{q+1} = w^q - \eta^q (H^q)^{-1} g^q \tag{3.37}$$

where η is the learning rate, and q is the number of iteration.

The term $\eta^{(q)}$ is found by a line search method in the search direction $(H^{(q)})^{-1} g^{(q)}$.

b. Quasi-Newton Method

Quasi-Newton is the most popular algorithm in nonlinear optimization with a reputation for fast convergence. It works by exploiting the observation that on a quadratic error surface, the minimum can be obtained by using the Newton step involving the Hessian matrix.

Quasi-Newton methods attempt to approximate the matrix H^{-1} using a matrix A (More information on this derivation can be obtained from Hagan *et al.,* (1996).

$$\lim_{q \to \infty} A^{(q)} = H^{-1} \tag{3.38}$$

The gradient information is used to determine A . If the search direction is $d = Ag(w)$ and $\Delta w = \eta d$, $\Delta g = g(w + \Delta w) - g(w)$,

$$A^{(q+1)} = A^{(q)} - \frac{g\, g^T}{g^T d} + \frac{\Delta g (\Delta g)^T}{\eta (\Delta g)^T d} \tag{3.39}$$

η is the learning rate and is determined by a line search.

Main drawbacks of this algorithm are that the Hessian matrix is difficult and expensive to calculate and Newton step would be wrong if the error surface is non-quadratic. It requires a huge memory and therefore it is not suitable for large networks.

c. Conjugate Gradient Method

Conjugate gradient descent is implemented by creating line searches across the error surface. It starts by working out the direction of steepest descent. An advantage of this method is that the calculation of the second derivatives is not necessary. This means that conjugate gradient is quite fast as it only involves searching in one dimension. Following this, further line searches are carried out.

The steps involved in the conjugate gradient algorithm are (Bishop, 1996; Hagan *et al.,* 1996):

Step 1: The first search direction p_0 is selected to be the negative of the gradient g_0 .

$$p_0 = -g_0 \tag{3.40}$$

where $g_q \equiv \nabla F(x)\big|_{q=q1}$

Step 2: Select learning rate η and minimize the function along the search direction.

$$x_{q+1} = x_q + \eta_q P_q \tag{3.41}$$

Step 3: Select the next search direction.

Step 4: Repeat step 2-3 if the algorithm has not converged.

p_0 is an assumed search direction, $F(x)$ is the performance index to be minimized, x is the scalar parameter being adjusted, and g_0 is the gradient of the performance index.

d. Scaled Conjugate Gradient

The scaled conjugate gradient algorithm is very similar to the quasi-Newton algorithms. It was introduced to correct the defect in the line-search method used in the conjugate gradient method. This is because other conjugate gradient methods require that the network outputs be calculated several times during the search. The scaled conjugate gradient uses a step size scaling method by combining the conjugate gradient approach with another method known as the model-trust region. The model-trust region method restricts the model around the present search point.

A unit matrix I is added to the Hessian matrix H as shown in equation (3.42) (Bishop, 1996):

$$H + \lambda I \tag{3.42}$$

where λ is a scaling coefficient. The step size is small for large values of the scaling coefficient.

Therefore, the step length is given by:

$$\alpha_j = -\frac{d_j^T g_j}{d_j^T H_j d_j + \lambda_j \left\| d_j \right\|^2} \tag{3.43}$$

where g_j is the gradient vector, d_j is the search direction.

e. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (L-M) algorithm is regarded as a variation of the Newton's method for minimizing nonlinear functions. L-M algorithm is said to be good for small- and medium-sized functions, and it has a stable convergence. This is because L-M algorithm combines the advantages of the steepest descent algorithm and the Gauss-Newton algorithm.

The derivation of the L-M algorithm begins with the Newton's method and would be too long to cover in the thesis. However, further information on L-M algorithm and its derivation can be obtained from Bishop, 1996; Hagan *et al.,* 1996.

The update rule for the Gauss-Newton algorithm is given by (Bishop, 1996; Hagan *et al.,* 1996):

$$w^{q+1} = w^q - (J^{qT} J^q)^{-1} J^q e^q \tag{3.44}$$

The Hessian matrix H and Jacobian matrix J are related by:

$$H = J^T J \tag{3.45}$$

In order to make the Hessian matrix invertible, an approximation factor known as the identity matrix I is added to it. Such that:

$$H = J^T J + \mu I \tag{3.46}$$

By combining equations (3.44, 3.46), the update rule for the L-M algorithm is:

$$w^{q+1} = w^q - (J^{qT} \mu I)^{-1} J^q e^q \tag{3.47}$$

J is the Jacobian matrix, I is the identity matrix, and μ is the combination coefficient.

3.7 Activation Functions

The activation function acting on the input vector determines the total signal a neuron receives and the output function operating on the activation determines the output.
The composition of the activation and the output functions is called the transfer function.

There is a wide range of activating functions in use with neural networks. For back-propagation learning, the activation function must be differentiable (Hagan *et al.,* 1996).

A brief overview of the characteristics of the most common activation functions in use (as implemented in the MATLAB NN Toolbox) is given below (Hagan *et al.,* 1996; Demuth *et al.,* 2004).

3.7.1 Hard-Limit Transfer Function

This transfer function operates by restricting the output to 0 if the net input *n* for the neuron is below 0; or gives a 1 if *n* is greater than or equal to 0.

Figure 3.22 depicts the *hard-limit* transfer function. The hard limit transfer function forces a neuron to output a 1 if its net input reaches a threshold, otherwise it outputs 0. This allows a neuron to make a decision or classification. It can say *yes* or *no*. This kind of neuron is often trained with the perceptron learning rule.

Mathematically,

$$\left.\begin{aligned} hard\lim(0) &= 1, if\ n >= 0; \\ &= 0\ if\ otherwise \end{aligned}\right\} \qquad (3.48)$$

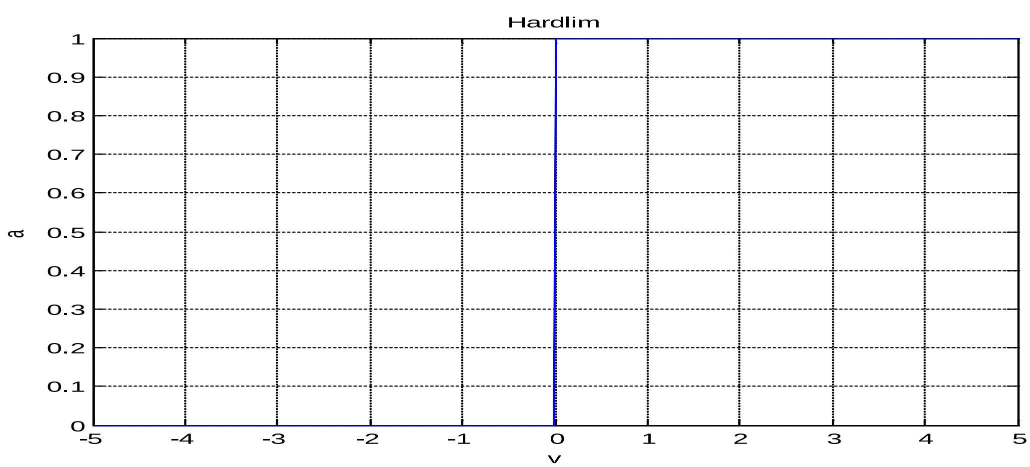


Figure 3.22: ***Hard limit* transfer function**

3.7.2 Hardlims Transfer Function

The symmetric hard limit transfer function forces a neuron to output a 1 if its net input reaches a threshold. Otherwise it outputs -1. Like the regular hard limit function, this allows a neuron to make a decision or classification. Figure 3.23 depicts the *hardlims* transfer function.

$$\left.\begin{array}{l} hard\lim(n) = 1, if\ n >= 0; \\ \qquad\qquad = -1 if\ otherwise \end{array}\right\} \qquad (3.49)$$

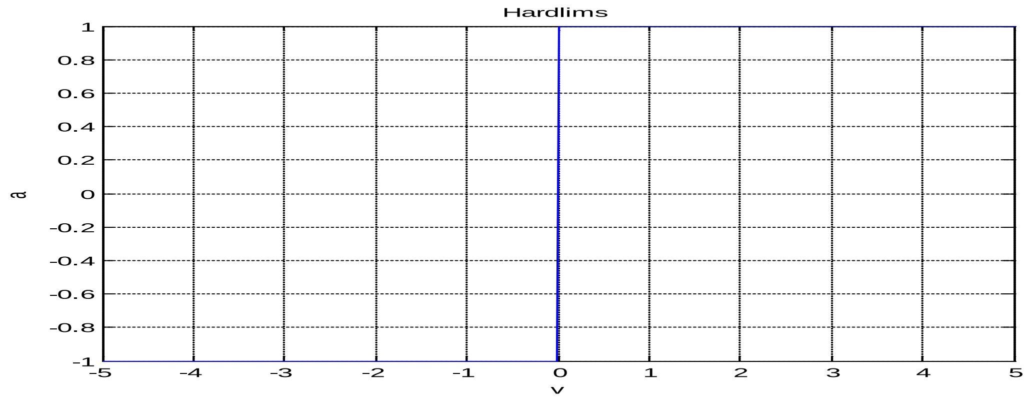


Figure 3.23: ***Hardlims*** **transfer function**

3.7.3 Logsig Transfer Function

Logsig (N) takes one input, *N-SxQ* matrix of net input (column) vectors and returns each element of *N* squashed between 0 and 1. Figure 3.24 depicts the *Logsig* transfer function.

A squashing function of the form shown below that maps the input to the interval (0,1) is:

$$\log sig(n) = \frac{1}{(1+\exp(-n))} \qquad (3.50)$$

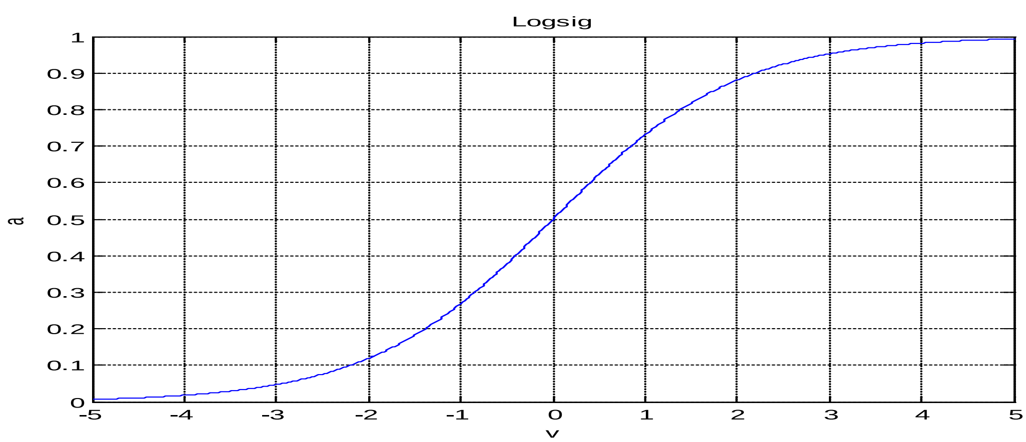


Figure 3.24: ***Logsig*** **transfer function**

3.7.4 Poslin Transfer Function

Poslin (N) takes one input, *N-SxQ* matrix of net input (column) vectors and returns the maximum of 0 and each element of *N*. Figure 3.25 depicts the *Poslin* transfer function.

$$poslin(n) = \left.\begin{array}{l} n, \text{ if } n >= 0; \\ = 0 \text{if } n <= 0 \end{array}\right\} \tag{3.51}$$

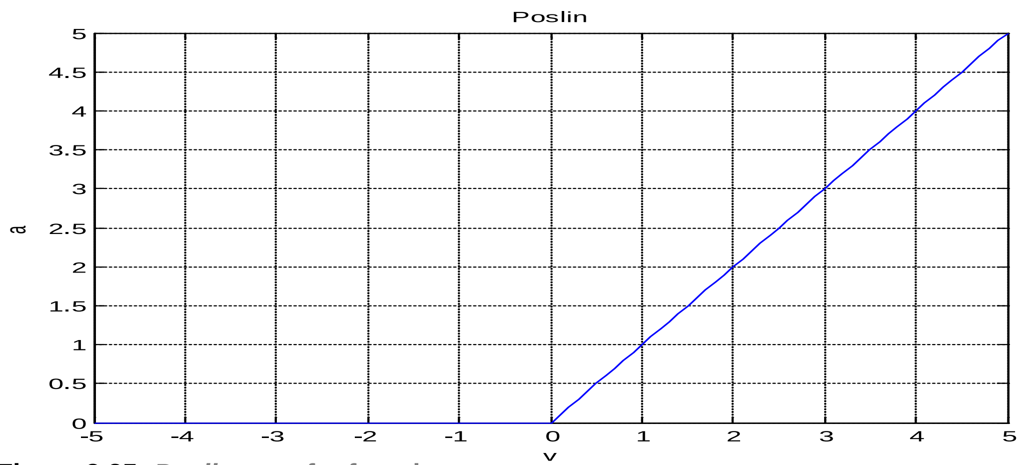


Figure 3.25: ***Poslin*** **transfer function**

3.7.5 *Purelin* Transfer Function

Purelin (N) takes an input, *N-SxQ* matrix of net input (column) vectors and produces an *N*. Figure 3.26 depicts the *Purelin* transfer function.

$$purelin(n) = n \tag{3.52}$$

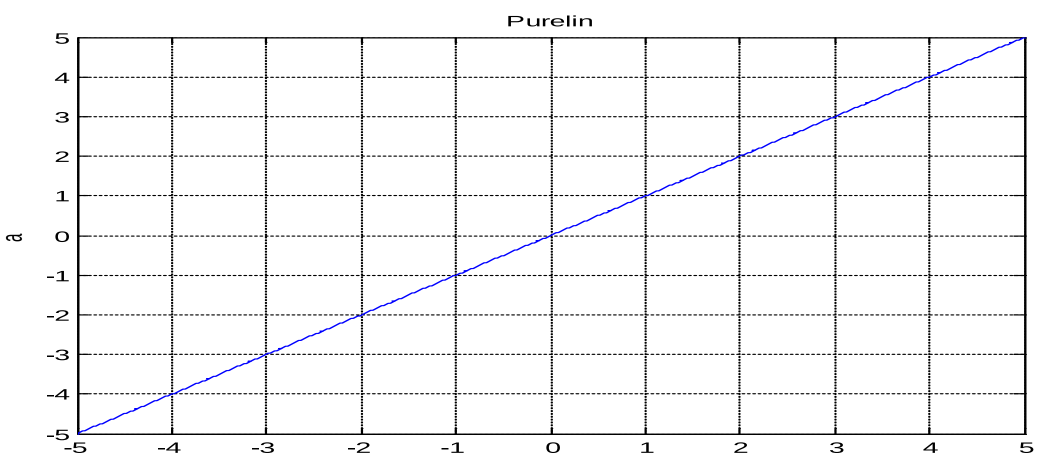


Figure 3.26: ***Purelin*** **transfer function**

3.7.6 Radbas Transfer Function

Radbas(N) takes one input, *N-SxQ* matrix of net input (column) vectors, and returns each element of *N* passed through a radial basis function. Figure 3.27 depicts the *Radbas* transfer function.

Radbas(N) calculates its output as:

$$a = \exp(-n^2) \tag{3.53}$$

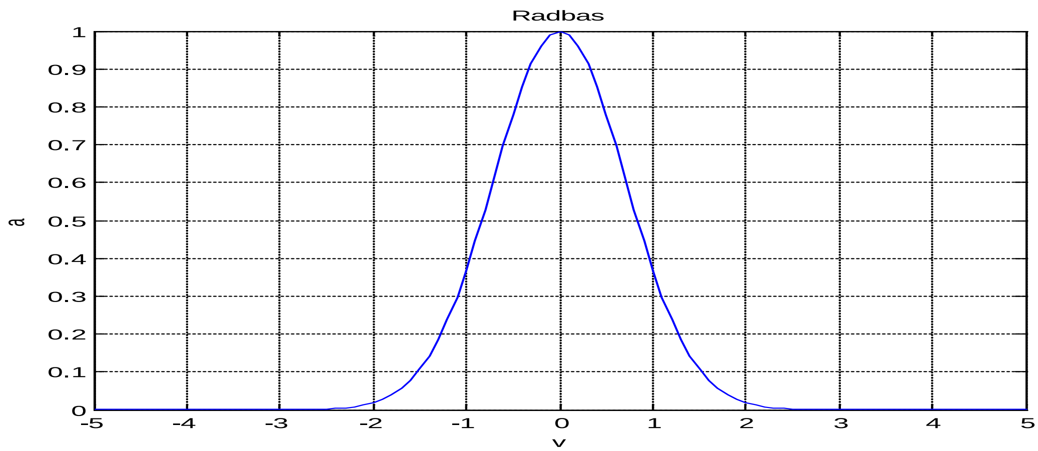


Figure 3.27: ***Radbas*** **transfer function**

3.7.7 Satlin Transfer Function

Satlin(N) takes an input, *N-SxQ* matrix of net input (column) vectors, and produces values of *N* in the interval [0, 1]. Figure 3.28 depicts the *Satlin* transfer function.

$$\left.\begin{aligned} satlin(n) &= 0, if\ n <= 0; \\ &= n, if\ 0 <= n <= 1; \\ &= 1, if\ 1 <= n \end{aligned}\right\} \tag{3.54}$$

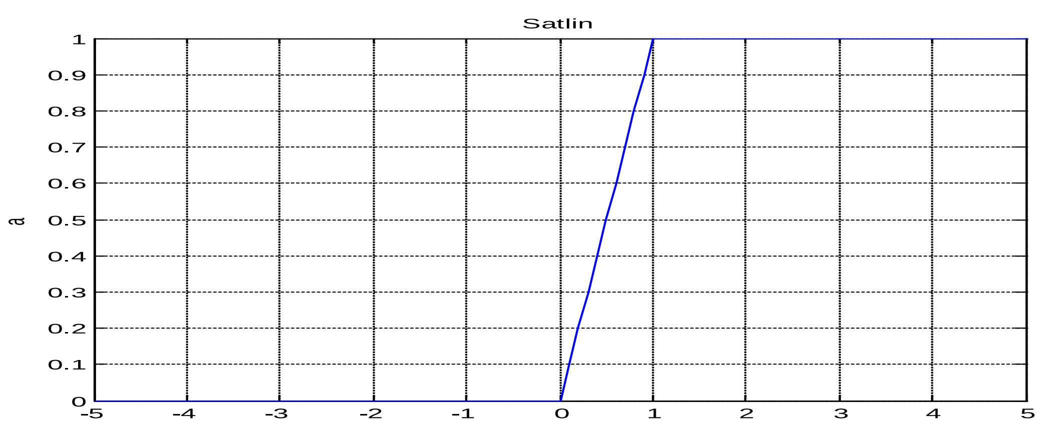


Figure 3.28: ***Satlin*** **transfer function**

3.7.8 Satlins Transfer Function

Satlins(N) takes one input, *N-SxQ* matrix of net input (column) vectors and returns values of *N* truncated into the interval [-1, 1]. Figure 3.29 depicts the *Satlins* transfer function.

$$\left.\begin{aligned} satlins(n) &= -1, if\ n <= -1; \\ &= n, if\ -1 <= n <= 1; \\ &= 1, if\ 1 <= n \end{aligned}\right\} \tag{3.55}$$

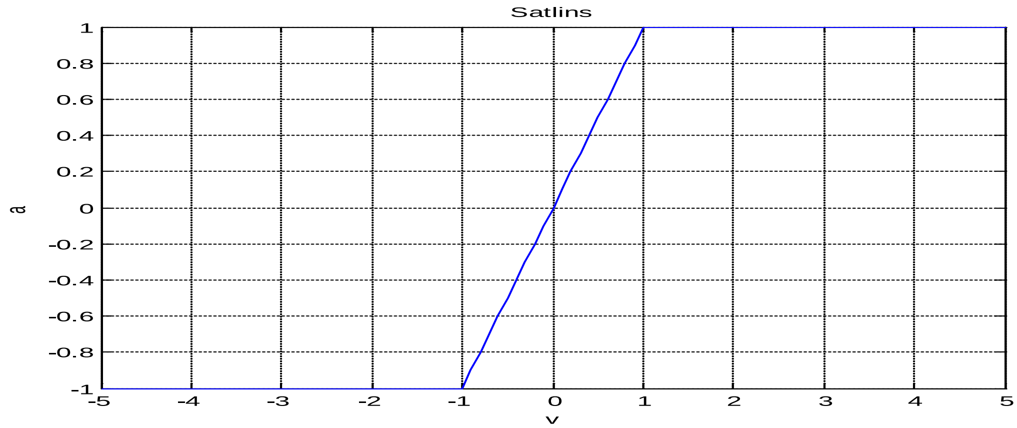


Figure 3.29: ***Satlins* transfer function**

3.7.9 Tansig Transfer Function

The *Tansig* transfer function takes an input *N-SxQ* matrix of net input (column) vectors, and gives each element of *N* squashed between -1 and 1. Figure 3.30 depicts the *Tansig* transfer function.

Tansig (N) calculates its output according to:

$$tansig = \frac{2}{(1 + \exp(-2 * n)) - 1} \tag{3.56}$$

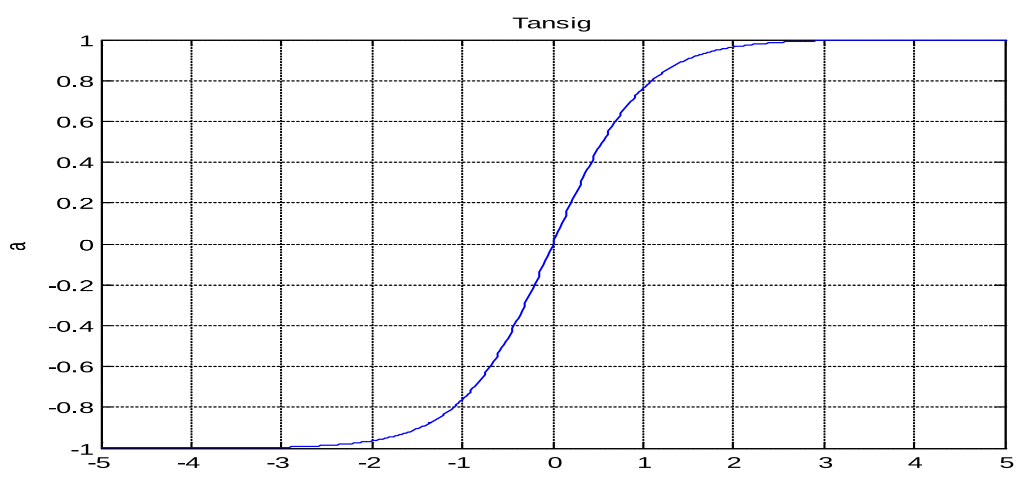


Figure 3.30: ***Tansig* transfer function**

3.7.10 Tribas Transfer Function

Tribas(N) takes one input, *N-SxQ* matrix of net input (column) vectors and returns each element of N passed through a radial basis function. Figure 3.31 depicts the *Tribas* transfer function.

Tribas(N) calculates its output with according to:

$$\left.\begin{aligned} tribas(n) &= 1 - abs(n), if\ -1 <= n <= 1; \\ &= 0, if\ otherwise \end{aligned}\right\} \tag{3.57}$$

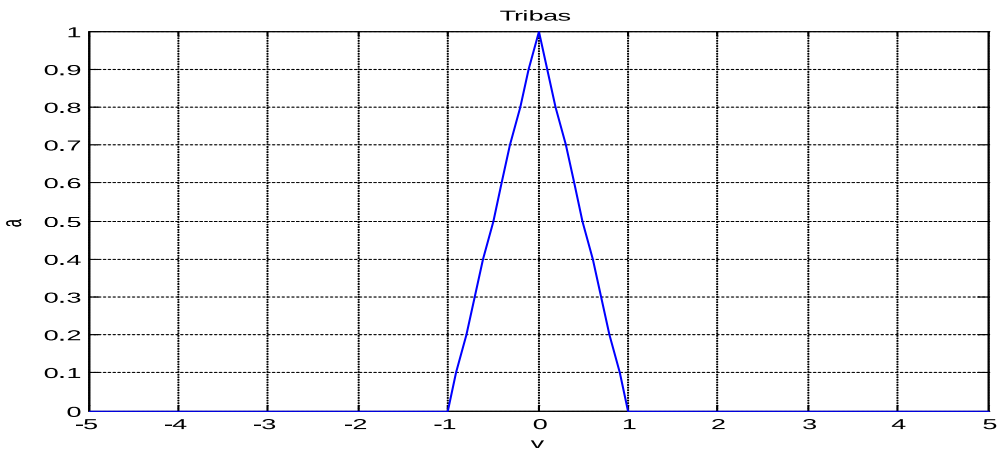


Figure 3.31: ***Tribas* transfer function**

3.8 Generalization

Generalization is the ability of a trained network to make a hypothetical response to an input that it has never seen before and make a reasonable response (Park, 1996).

It is of utmost importance that a NN be able to generalize. This is because in reality, it is somewhat impossible to obtain an exhaustive list of all situations that a system may be exposed to.

The methods that can be used to improve generalization include:

- Early stopping
- Regularization

3.8.1 Overfitting and Underfitting

Overfitting or underfitting is essentially how well the NN make predictions for cases that are not in the training set.

Three typical conditions necessary for good generalization are:

- The first condition is that the inputs to the network must contain sufficient information pertaining to the target.
- The second condition is that the function that is being approximated should be to some degree smooth such that a slight variation in the input vectors should produce a small change in the outputs.
- Thirdly, the training cases should be a sufficiently large and should represent likely scenarios where it would be used. This allows the network to interpolate cases that are in the neighbourhood of training vectors. If the cases are outside the range of the training vectors, then it requires extrapolation.

3.8.2 Early Stopping

In early stopping, training stops before the network begins to overfit. This is implemented by dividing the training vectors into three subsets. The training dataset, the validation dataset, and the test dataset. The validation set is used to determine the network performance on patterns which do not form part of the training set, but which determine when to terminate training.

3.8.3 Regularization

Regularization adds constraints to a neural network so that the results are more consistent. The performance function is modified in such a way that the algorithm prunes the network through driving irrelevant estimates to zero (Rech, 2002).

With regularization, the usual performance function for feed-forward networks (MSE) defined by equation (3.66) is replaced by *msereg* by adding the mean of the sum of squares of the network weights and biases (Demuth *et al.,* 2004):

$$f = mse \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \tag{3.58}$$

$$msereg = ymse + (1 - y)msw \tag{3.59}$$

where γ is the performance ratio, and *msw* is given by:

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \tag{3.60}$$

The main advantage of using regularization is that even if the neural network model is over parameterized, the irrelevant parameter estimates are likely to be close to zero and the neural network model behaves like a small network (Rech, 2002).

3.9 Conclusion

This chapter started with an introduction to signal analysis. Various signal processing techniques like the Fourier Transform, Short Time Fourier Transform, and Wavelet Transform were discussed. Wavelet analysis using Continuous Wavelet Transform and Discrete Wavelet Transform were further elaborated upon.

Also, a theoretical introduction to neural networks was presented. The relationship between the biological and artificial neuron were also highlighted. Historical background providing some perspective on the developments of the field of neural networks was given. Various activation functions were discussed. Furthermore, the types of neural network learning and learning algorithms were explained in detail. In addition, the various classes of neural networks based on topology and implementation were discussed.

The proposed method in this thesis makes use of multilayer perceptron neural networks. The tasks for which these networks are used for can be classified as pattern recognition for the fault section identification task, and function approximation for the fault location task. NNs were used because of their ability to learn, their immunity to noise, and their ability to generalize to new inputs.

Chapter 4 discusses the modelling and simulation of the IEEE benchmark model. Steady-state and transient simulations were carried out using the IEEE 34 node benchmark model for various fault conditions. Extensive simulation covering fault location, fault resistance, fault inception angle, load angle variation, load and capacitor switching would be carried out to generate the data needed for the proposed method. Modified case studies involving the integration of DGs and line extension of the test feeder would also be done in this chapter.

CHAPTER FOUR
NETWORK MODELLING AND SIMULATION

4.1 Introduction

Distribution feeders are characterized by having only one path for power flow from the source to the load centres. Feeders may consist of the following:

- Main feeder
- Laterals or tap-offs
- Loads
- Voltage regulators
- Shunt capacitor banks
- Distribution transformers

The IEEE Power Engineering Society Distribution Subcommittee published five benchmark distribution feeders with their configurations, parameters, and power flow results performed using the Radial Distribution Analysis Package from WH Power Consultants and/or Windmil® developed by Milsoft Integrated Solutions (Kersting, 2000).

Amongst these feeders, the IEEE 34 node test feeder (shown in Figure 4.1) was selected for this thesis. The chosen test feeder is an actual feeder located in Arizona. It is long and lightly loaded, with a rated voltage level of 24.9 kV. The IEEE 34 node test feeder is an unbalanced system with uneven load distribution and unequal lateral spacing.

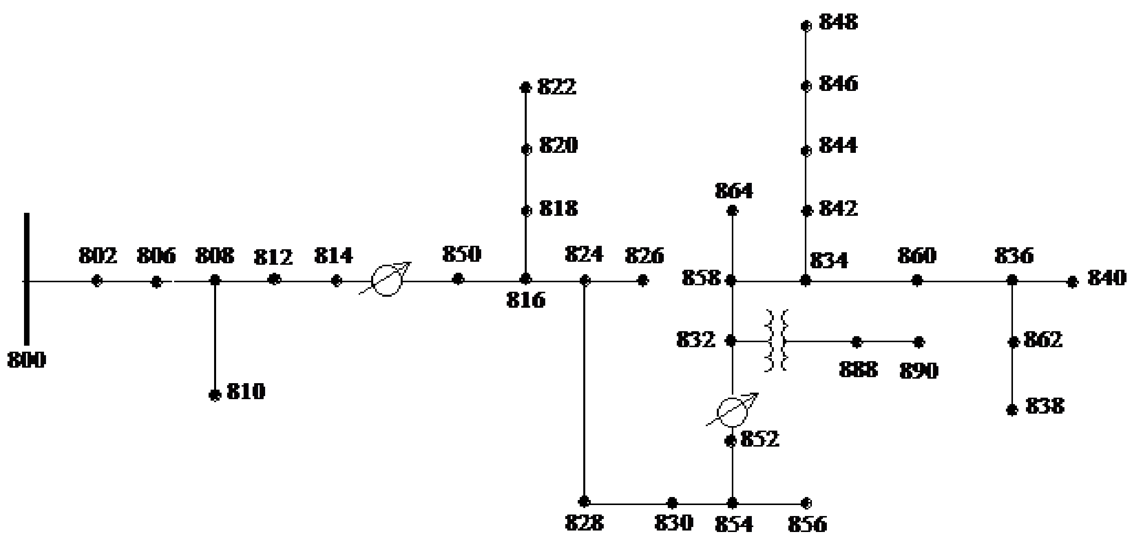


Figure 4.1: **The IEEE 34 Node radial test feeder**

The network elements in the IEEE 34 node test feeder include a substation transformer, an in-line transformer, shunt capacitors, voltage regulators, loads, and overhead distribution lines of various configurations.

This chapter covers the generation of the data required for the development of the Hybrid Fault Detection and Diagnosis (HFDD) method. A description of the modelling steps used is given herein. Also, simulation of fault and ‘no fault’ conditions in the IEEE 34 node test feeder is presented. Furthermore, post-simulation operations on the generated data are given.

4.2 Experiments for Data Generation

Extensive simulations were done in order to obtain data for steady state system operation and fault conditions. The steady state simulation covered variation in load angle and the switching on and off of loads and capacitors. Fault conditions involving ten fault types namely Single Line-to-Ground (SLG), Two Phase (2Ph.), Two Phase-Ground (2Ph.-G) and Three Phase (3Ph.) faults were simulated on the test feeder respectively.

The above-mentioned fault types were carried out using various locations in the feeder with different fault resistances and fault inception angles. Figure 4.2 is a chart showing the procedure followed.

Experiments done also involved the investigation of the effect of external disturbance. This was implemented by integrating Distributed Generations (DGs) in the test feeder. Another experiment conducted was by extending the line at the farthest point of the test feeder at node 840. Tables 4.1 and 4.2 presents a plan of the experiments carried out on the test feeder.

Table 4.1: Plan of experiments for ‘No Fault’ condition

Nr.	No Fault Conditions	System Parameter
1.	Load angle variation/(^{o})	0, 30, 45, 60,and 90
2.	Load switching	Switching on and off of 19 distributed loads and 6 spot loads
3.	Capacitor switching	Switching on and off of capacitors at nodes 844 and 846.

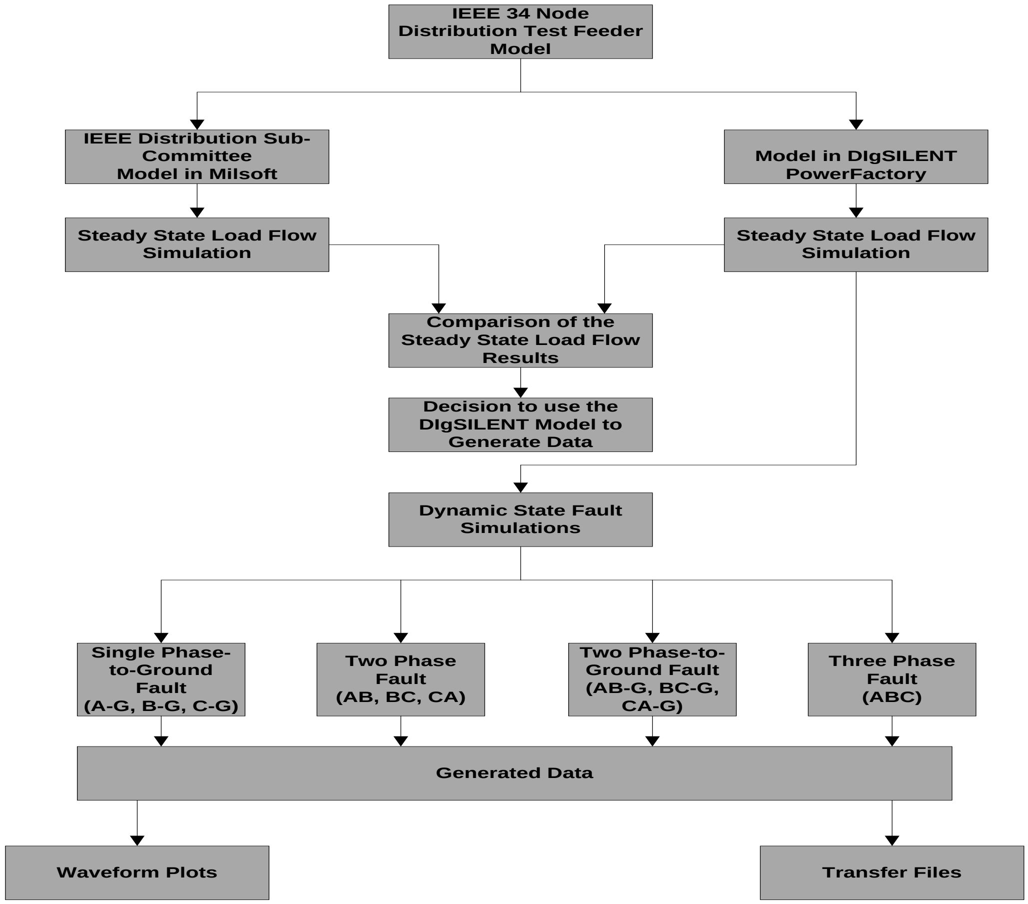


Figure 4.2: Summary of the modelling and simulation procedure used

Table 4.2: Plan of experiments for fault conditions

Nr.	Fault Condition.	Fault and System Parameter
1.	Fault location /(%)	10, 15, 20, 25, 30, 45, 50, 60, 70, 80, 85, 90, 95 of the main feeder, 15, 30, 60, 80, 90 of all the laterals
2.	Fault types	Single Phase-to-ground (1 Ph.-g), two Phase (2 Ph.), two phase-to-ground (2 Ph.-g), and three phase (3 Ph.)
3.	Fault resistance /(Ω)	0, 0.5, 2.5, 5, 10, 20, and 100
4.	Fault Inception angle/(^{o})	0, 30, 45, 60,and 90

4.3 Feeder Modelling

This distribution test feeder was modelled using DIgSILENT PowerFactory 13.2.341 and it is made up of thirty-two line segments, five overhead line configurations, and two transformers. It also includes two shunt capacitor banks, two voltage regulators, six spot loads, and nineteen distributed loads. The test feeder modelled in DPF showing various fault segments is depicted in Figure 4.3.

4.3.1 Nodes

The thirty-two line segments of the test feeder consist of a main feeder, nine laterals, and one sub-lateral.

There are a total of six single-phase laterals, labelled 808-810, 816-822, 824-826, 854-856, and 858-864. Three-phase laterals include laterals 832-890, 834-848, 836-862, and 836-840. 862-838 is a sub lateral. Also, lateral 832-890 is of special interest due to their unique characteristics with respect to the main feeder. It operates on a different voltage level and produces an undervoltage at its farthest node (node 890).

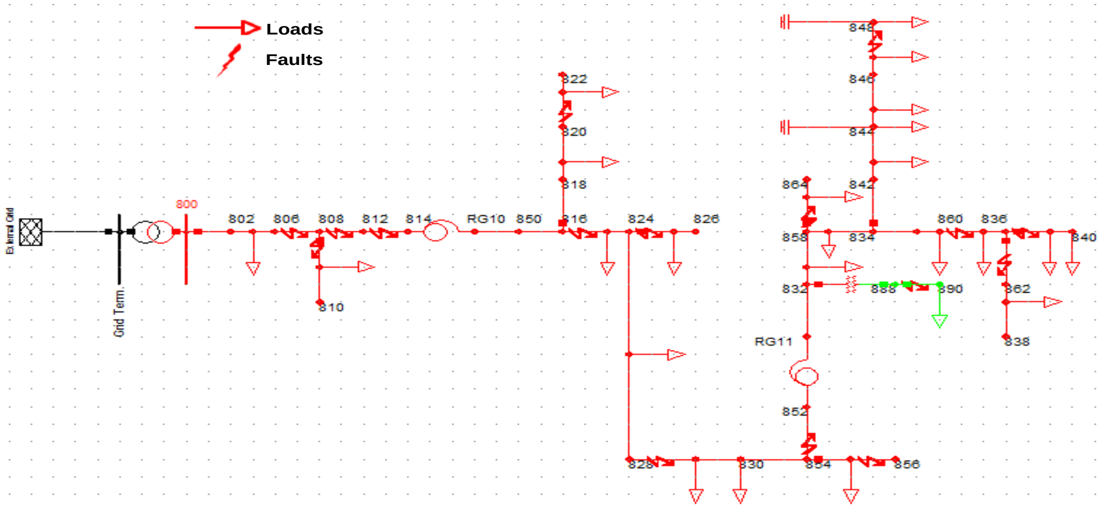


Figure 4.3: **Single line diagram of the IEEE 34 node test feeder in DIgSILENT PowerFactory**

The reason for the undervoltage is mostly due to the high magnitude of the constant current spot load and the line losses along the length of the feeder. Lateral 834-848, on the other hand, is the only location with reactive compensation. A summary of the laterals is given in Table 4.3.

Table 4.3: **Summary of laterals in the chosen test feeder**

Nr.	Feeder section	Lateral nomenclature used	Phase type	Length (ft)	Voltage level (kV)	Circuit component(s)
1.	808-810	L.808	B-N	5804	24.9	Loads
2.	816-822	L.816	A-N	63600	24.9	Load
3.	824-826	L.824	B-N	3030	24.9	Load
4.	854-856	L.854	B-N	23330	24.9	Load
5.	832-890	L.832	3 Ph.	10560	4.16	Transformer & Load
6.	858-864	L.858	A-N	1620	24.9	Load
7.	834-848	L.834	3 Ph.	5800	24.9	Loads & shunt capacitors
8.	836-838	L.836/1	3 Ph./B-N	5140	24.9	Load
9.	836-840	L.836/2	3 Ph.	860	24.9	Loads
10.	Main feeder	MF	3 Ph.	189,229	24.9	Transformer, loads, & regulators

4.3.2 Transformer Models

The voltage source used for the test feeder was an external grid provided in the DIgSILENT PowerFactory library. Two transformer models were made use of in this feeder. The first transformer is the substation transformer located at node 800, and the second is in-line within the feeder at node 832. The upstream transformer is a 2.5MVA 69/24.9kV transformer and was modeled with the *'three phase two winding transformer'* model of DIgSILENT PowerFactory. The parameters used are shown in Appendix A.1.

The 0.5MVA, 24.9/4.16kV in-line transformer was modeled with a *'three-phase two winding transformer'* model. The parameters for the configuration of the transformers are shown in Figures 4.4 and 4.5.

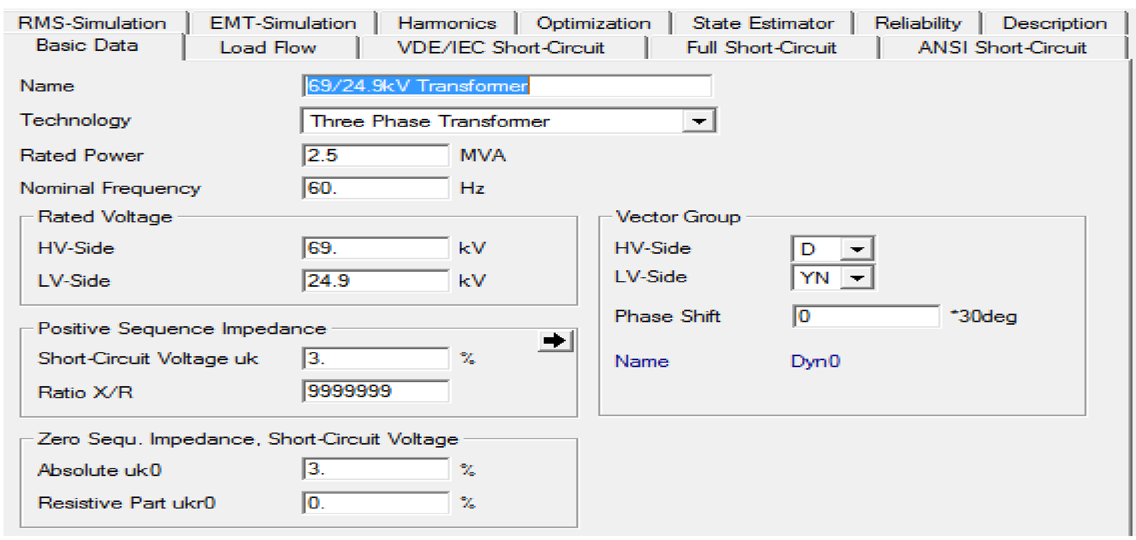


Figure 4.4: **Substation 2.5MVA 69/24.9kVA transformer parameters**

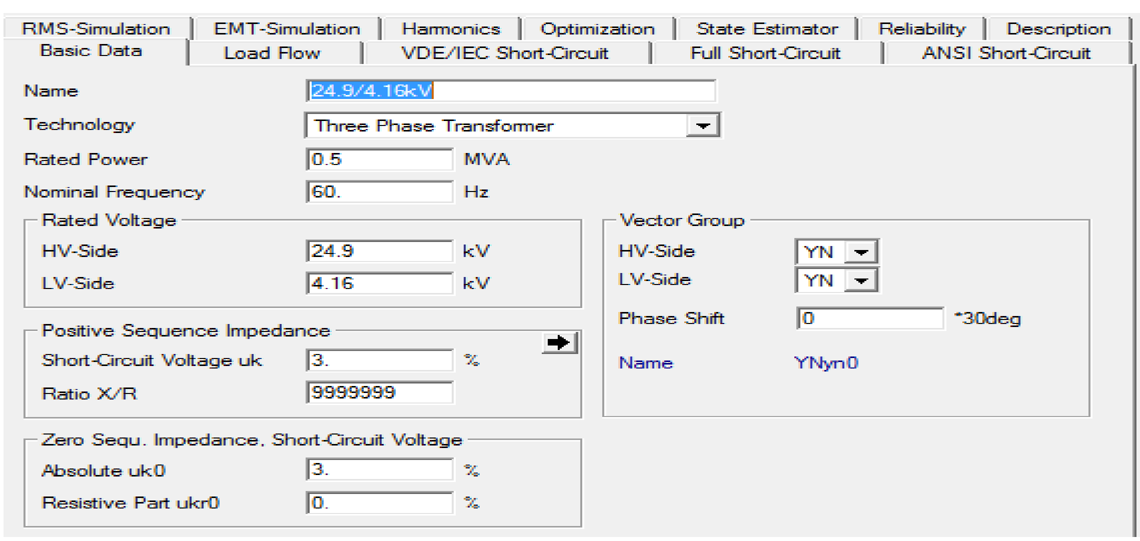


Figure 4.5: **In-line 0.5MVA 24.9/4.16kVA transformer parameters**

4.3.3 Line Models

The lumped *pi* model was used to model all the line segments since the feeder is made up of short lines. Five overhead line configurations (300-304) are specified in the IEEE 34 node test feeder parameters as shown in Appendix A.2. These line configurations determine the type of circuit (single phase or three phase), the type of overhead conductor, the spacing between the conductors, the impedance, and the susceptance. The main feeder has a total length of 189,230ft (57,677.30 metres) and it was modelled with configuration type 300 and 301 as given in Appendix A.2 respectively. Similarly, the various single phase laterals were modelled using the 302, 303, and 304 line configuration types.

There are three methods for modelling the lines in DIgSILENT PowerFactory.
These are:

- Line Type (*TypeLne*)
- Tower Type (*TypeTow*)
- Geometry Type (*TypeGeo*)

TypeLne was used in the thesis in the modelling of the lines. Carson's equation and Kron's reduction method were used to transform the impedance and susceptance matrices to positive and zero sequence impedances and shunt susceptances.
For a phase impedance matrix defined by (4.1),

$$[Z_{abc}] = \begin{bmatrix} Z_{aa} & Z_{ab} & Z_{ac} \\ Z_{ba} & Z_{bb} & Z_{bc} \\ Z_{ca} & Z_{cb} & Z_{cc} \end{bmatrix} \tag{4.1}$$

the self impedance Z_s and mutual impedance Z_m are given by (Kersting, 2002):

$$Z_s = \frac{1}{3}\left(Z_{aa} + Z_{bb} + Z_{cc}\right) \tag{4.2}$$

$$Z_m = \frac{1}{3}\left(Z_{ab} + Z_{bc} + Z_{ca}\right) \quad = \frac{1}{3}\left(Z_{ac} + Z_{ba} + Z_{cb}\right) \tag{4.3}$$

Thus, the phase impedance is transformed to:

$$[Z_{abc}] = \begin{bmatrix} Z_s & Z_m & Z_m \\ Z_m & Z_s & Z_m \\ Z_m & Z_m & Z_s \end{bmatrix} \tag{4.4}$$

From the above, the sequence impedance is given as:

$$Z_{00} = Z_s + 2Z_m \tag{4.5}$$

$$Z_{11} = Z_{22} = Z_s - Z_m \tag{4.6}$$

Table 4.4 lists the values of the positive and zero sequence impedances (Z_{11} and Z_{00}) and susceptances (B_1 and B_0) calculated from equations (4.5) and (4.6) respectively.

The obtained positive and zero sequence impedances and susceptances were used as the parameters for the *pi* sections as shown in Figure 4.6. The lengths of the lines are specified in Appendix A.3.

Table 4.4: **Line parameters for the test feeder**

Line Configuration	$R_{11} + jX_{11}$	$R_{00} + jX_{00}$	B_1	B_0
300	1.1911+j1.0005	1.6078+j2.0373	5.8218	3.6693
301	1.7687+j1.0309	2.2301+j2.1988	5.5911	3.5595
302	0.9332+j0.4952	0.9332+j0.4952	1.4084	1.4084
303	0.9332+j0.4952	0.9332+j0.4952	1.4084	1.4084
304	0.6406+0.4737	0.6406+0.4737	1.4546	1.4546

4.3.4 Load Model

The IEEE 34 Node Test feeder has six spot loads and nineteen distributed loads. The sum of the entire load in the feeder is 1,769 kW as given in Appendix A.4.
The load types specified either as wye or delta loads are modelled as constant power (PQ) load, constant current (I) load, and constant impedance (Z) load.

The spot loads were all three-phase loads and were connected to their respective nodes. Distributed loads are placed at the midpoint of the line. The loads were modelled as constant power, constant current, constant impedance as per (Kersting, 2002).

DIgSILENT PowerFactory (DPF) does not include models of constant power (*PQ*) load, constant current (*I*) load, and constant impedance (*Z*) load. In lieu of this, a derivation by Subrahmanyam, (2009); Subrahmanyam and Radhakrishna, (2010) which relates the load type to the voltage dependency factors obtainable in DPF was used.

The load current at the *qth* bus for a three phase star connected load or single phase load connected line to neutral for star and delta load models are defined in (Subrahmanyam, 2009; Subrahmanyam and Radhakrishna, 2010):

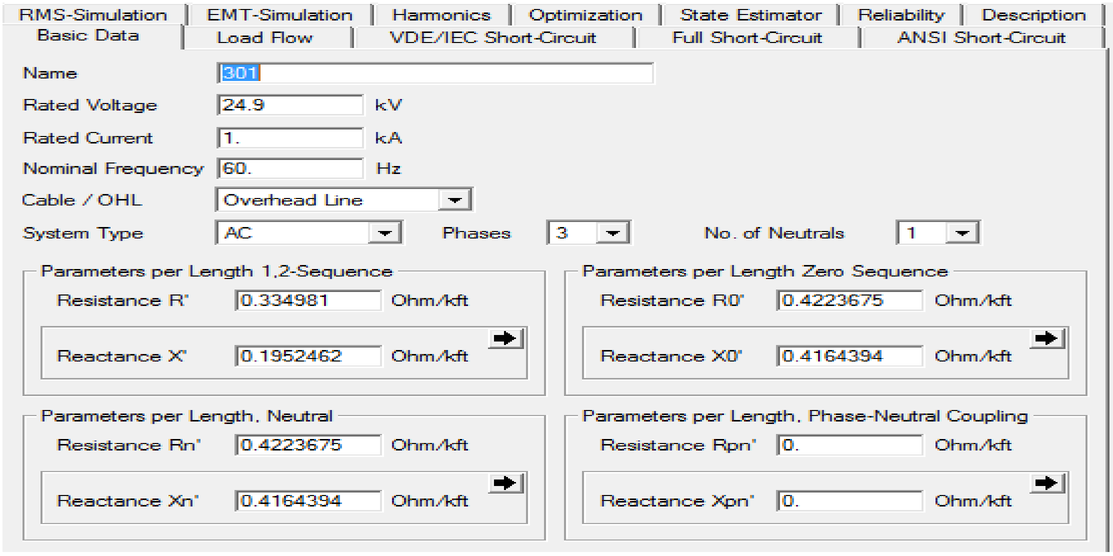


Figure 4.6: Line parameters

$$\begin{bmatrix} IL_q^a \\ IL_q^b \\ IL_q^c \end{bmatrix} = \begin{bmatrix} \left(\dfrac{SL_q^a}{V_q^a}\right)^* * \left|V_q^a\right|^n \\ \left(\dfrac{SL_q^b}{V_q^b}\right)^* * \left|V_q^b\right|^n \\ \left(\dfrac{SL_q^c}{V_q^c}\right)^* * \left|V_q^c\right|^n \end{bmatrix} = \begin{bmatrix} \left(\dfrac{PL_q^a + jQL_q^a}{V_q^a}\right)^* * \left|V_q^a\right|^n \\ \left(\dfrac{PL_q^b + jQL_q^b}{V_q^b}\right)^* * \left|V_q^b\right|^n \\ \left(\dfrac{PL_q^c + jQL_q^c}{V_q^c}\right)^* * \left|V_q^c\right|^n \end{bmatrix} \tag{4.7}$$

where IL_q^p is the complex load current at the qth bus, V_q^p is the voltage at bus q, PL_q^p, QL_q^p, SL_q^p are the real, reactive and complex power loads for the qth bus. $p \in a,b,c$, and n is the neutral.

Similarly, the load current at the qth bus for a delta connected three-phase or a line-to-line single phase load is:

$$\begin{bmatrix} IL_q^{q} \\ IL_q^{b} \\ IL_q^{c} \end{bmatrix} = \begin{bmatrix} \left(\dfrac{SL_q^{ab}}{V_q^{ab}}\right)^* * \left|V_q^{ab}\right|^n - \left(\dfrac{SL_q^{ca}}{V_q^{ca}}\right)^* * \left|V_q^{ca}\right|^n \\ \left(\dfrac{SL_q^{bc}}{V_q^{bc}}\right)^* * \left|V_q^{bc}\right|^n - \left(\dfrac{SL_q^{ab}}{V_q^{ab}}\right)^* * \left|V_q^{ab}\right|^n \\ \left(\dfrac{SL_q^{ca}}{V_q^{ca}}\right)^* * \left|V_q^{ca}\right|^n - \left(\dfrac{SL_q^{bc}}{V_q^{bc}}\right)^* * \left|V_q^{bc}\right|^n \end{bmatrix} \tag{4.8}$$

$$= \begin{bmatrix} \left(\dfrac{PL_q^{ab} + jQL_q^{ab}}{V_q^{ab}}\right)^* * \left|V_q^{ab}\right|^n - \left(\dfrac{PL_q^{ca} + jQL_q^{ca}}{V_q^{ca}}\right)^* * \left|V_q^{ca}\right|^n \\ \left(\dfrac{PL_q^{bc} + jQL_q^{bc}}{V_q^{bc}}\right)^* * \left|V_q^{bc}\right|^n - \left(\dfrac{PL_q^{ab} + jQL_q^{ab}}{V_q^{ab}}\right)^* * \left|V_q^{ab}\right|^n \\ \left(\dfrac{PL_q^{ca} + jQL_q^{ca}}{V_q^{ca}}\right)^* * \left|V_q^{ca}\right|^n - \left(\dfrac{PL_q^{bc} + jQL_q^{bc}}{V_q^{bc}}\right)^* * \left|V_q^{bc}\right|^n \end{bmatrix} \tag{4.9}$$

where n = kpu = kqu and is defined as follows:

Constant power: kpu = kqu = 0; Constant current: kpu = kqu = 1; Constant impedance: kpu = kqu = 2.

In order to implement this, the values of voltage dependency factors kpu and kqu were set as shown in Table 4.5 (DIgSILENT Manual, 2005):

Table 4.5: **Voltage dependency factors**

Nr.	**Load Type**	***kpu.***	***kqu.***
1.	Constant Power (PQ)	0	0
2.	Constant Current (I)	1	1
3.	Constant Impedance (Z)	2	2

4.3.5 Voltage Regulator Models

Two voltage regulators located at nodes 814-850 and 852-832 are used in the feeder. No voltage regulator component is available in DIgSILENT PowerFactory. Therefore, the regulators were implemented using three-phase autotransformers. The primary side voltages were set equal to the regulator's input voltages, while the secondary side voltages were set equal to the regulator's output voltages. The voltage can be varied by changing the autotransformer taps.

Figure 4.7 shows the concept used to model the voltage regulators by using the autotransformer component in DIgSILENT Power Factory. Appendix A.5 presents the parameters used for this component.

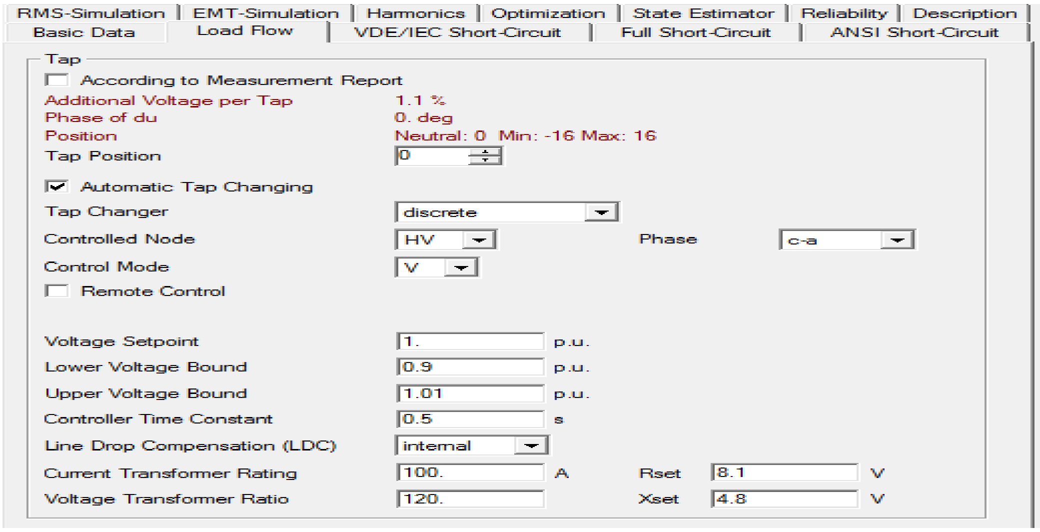


Figure 4.7: **Voltage regulator model**

A line drop compensator is used as a control circuit to determine the position of the tap. Most standard step regulators contain a switch within a range of +10% (Kersting, 2002). The voltage regulator was configured for 32 steps. That is, 16 positive and 16 negative steps, and these were used to regulate the line voltage within predefined limits.

4.3.6 Shunt Capacitor Models

The two capacitor banks identified on the feeder are located at nodes 844 and 848 respectively. Shunt capacitor banks are mainly utilized to assist in voltage regulation and for power factor correction (reactive power support). The capacitor banks are modelled as three-phase banks and connected in either wye or delta configuration. The parameters used are given in Appendix A.6.

4.3.7 Distributed Generator (DG) Models and Line Extension

The effects of the Integration of DGs into the test feeder were considered. Their effect over the predictive capabilities of the proposed algorithms is investigated.

Distributed generation is the generation of electric power (usually between 5kW and 10MW) at consumption end of the distribution network. The generated power is integrated to the distribution network at the substation, feeder, or customer load levels (Barker & de Mello, 2000).

Conditions for the addition of DG sources are: It must be reliable, dispatchable, of the proper size or penetration level, and at the proper locations (Barker & de Mello, 2000). DGs can be implemented with wind turbine, hydro, PV, fuel cells, etc.

Renewable energy sources are finding more usage because of the environmental implications (Silva *et al.,* 2007). The positive impacts of DG are said to provide system support benefits to the distribution system in the form of loss reduction, improved utility system reliability, voltage support, and improved power quality (Barker & de Mello, 2000). The studies carried out in the thesis did not assign any specific energy source to the DG.

The parameters of the synchronous generators were based on previous work carried out by Silva *et al.,* (2007) as shown in Appendix A.7. The connection of the generator to the grid was via a 500kVA step-up transformer. The transformer impedances were also set equal to the transformer at Node 832-888 (XFM-1). However, the transformer winding was changed to delta-star type based on the recommendations by Barker & de Mello, (2000) on optimal transformer winding types for DGs. The placement and sizing of the DGs were based on recommendation given by Dugan & Kersting, (2006), Samaan *et al.,* (2006), Santoso & Zhou, (2006), Silva *et al.,* (2007).

In addition to the above, the effect of line extension on the network was investigated by extending the network by 5,270ft. at Node 840. The impedances and other line parameters were assumed to be similar to that of lateral 834-846. Also, distributed load of 54kW was added to the network.

4.4 Comparison of Steady-State Load Flow Calculation Results with IEEE Results

In order to prepare the radial test feeder for further studies, the steady state-load flow results of the modelled feeder was compared with the IEEE's published results (Kersting, 2004). The load flow calculation was done for the considered unbalanced three phase network.

The following options were selected for this load flow calculation:

a. Basic Option > *Unbalanced, 3 Phase (ABC)*
b. Basic Option > *Automatic Tap Adjust of Transformer*
c. Basic Option > *Consider Coincidence of Low-Voltage Loads*
d. Active Power Control > *as Dispatched*
e. Advanced Options > *Current Approach (Kirchoffs law)*

The *'Unbalanced 3 Phase (ABC)'* option was used to perform load-flow calculation for multi-phase systems and the associated unbalances as a result of non-transposed lines, single-phase and two-phase loads.

Using the *'Automatic Tap Adjust of Transformer'* option adjusts the tap changers of the transformers automatically during the simulation. This was needed to implement voltage regulation using the autotransformer component.

Similarly, in order to model constant power, constant current, constant impedance loads, the *'Consider Coincidence of Low-Voltage Loads'* option was used. Consequently, the voltage dependency factors kpu and kqu was set to 0, 1 and 2 respectively for the load models according to the parameters.

'Active Power Control > *as Dispatched'* was used so that the total power balance will be established by the external grid. The nodal equations for solving power flow can be implemented in two ways. The *P, Q-balance* or energy conservation method and the *ir,li-*balance or Kirchoff's law method. The 'Advanced Options > *Current Approach (Kirchoffs law)'* was used because unbalanced distribution systems usually converge better using the *'Current Approach'* (DIgSILENT Manual, 2005). Figures 4.8 and 4.9 shows the load flow settings for basic and advance options in DPF.

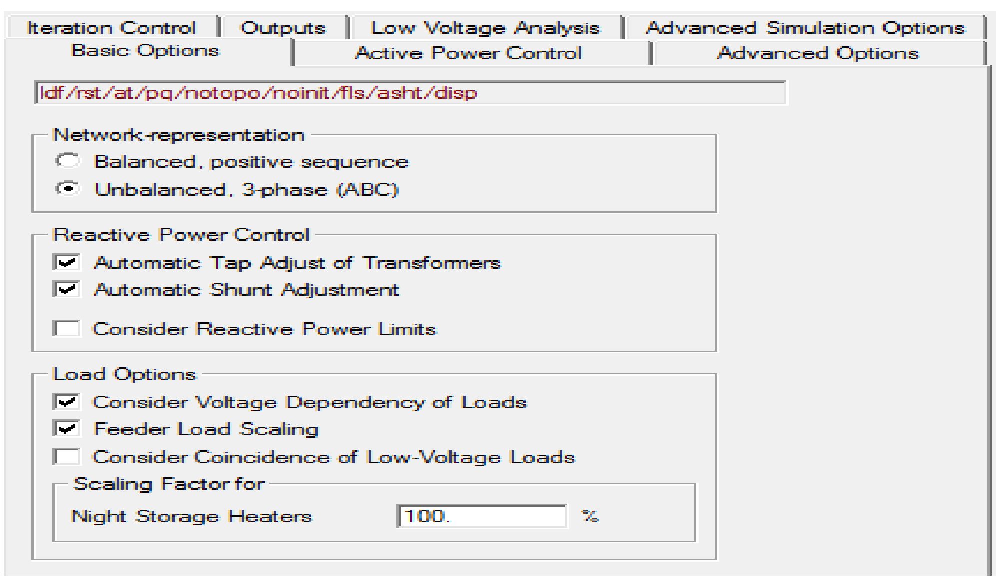


Figure 4.8: **Basic option load flow settings in DIgSILENT PowerFactory**

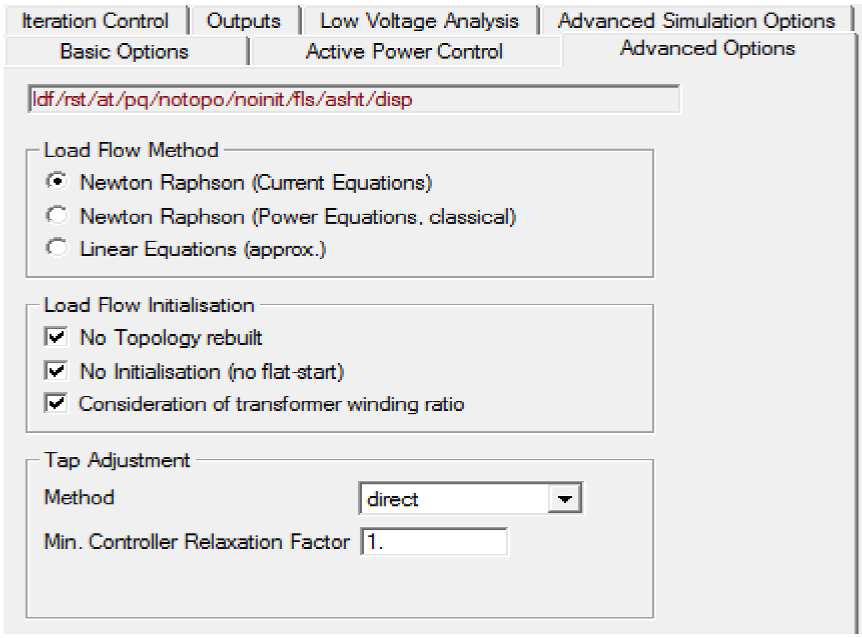


Figure 4.9: **Advanced options load flow settings in DIgSILENT PowerFactory**

Tables 4.6 and 4.7 present the load flow result obtained and the comparison of the node voltage and line current obtained from DIgSILENT PowerFactory simulation with the published reference results from the IEEE Distribution System Analysis Subcommittee (Kersting, 2004). The variation in results can be attributed to the modelling difference in DIgSILENT PowerFactory and that used by the IEEE subcommittee.

The IEEE reference results were modelled in a software package developed by Milsoft. This model took into account the unequal mutual coupling in the line impedances because it made use of the line geometry configuration, whereas the modelling implemented for this thesis used the line impedance values.

Also, the implementation of the various types of loads was different. DIGSILENT PowerFactory does not have models for constant power, constant current, constant impedance, and the voltage dependency factors were improvised upon and used to simulate constant power, constant current and constant impedance loads respectively.

Furthermore, Table 4.6 showed that the maximum error recorded for node voltages was -5.2032% on phase A. Table 4.8 presents the summarized analysis between the model in DIgSILENT PowerFactory and the IEEE benchmark result. The maximum error recorded for line current was -5.9389% also on phase A. The average error for the node voltage is as given in Table 4.8. From Tables 4.6, the relative errors for the node voltages in phases B and C were observed to be low. For phase A, majority of the node voltage relative errors were below -2.9689. The close agreement and low relative error between the IEEE results and the average error obtained provides the validation of the result acquired from the DIGSILENT PowerFactory modelling.

Thus, the modelled network in DIgSILENT PowerFactory is adequate for the power system studies required for the generation of data in the thesis.

4.5 Short Circuit Studies

Short circuit analysis studies finds application in network planning and in solving operation problems. In network planning, the ratings of network equipment, expected maximum and minimum currents are checked. This is done in order to select the switchgears, determine the rating of the components, and to ensure that the protection philosophy is viable. Solution to operational problem has to do with the precise evaluation of the fault current levels. For instance, in order to determine if the malfunction of a protection device was as a result of failure or wrong settings.

The short circuit studies for this thesis were implemented using DIgSILENT PowerFactory (DPF) to calculate the minimum and maximum fault currents for various segments of the feeder in order to study the fault current level in the feeder.

Two calculation options available in DPF were considered. These are the IEC 60909 standard of 2001 and the superposition method. The steady-state short circuit current (*Ikss*) calculation using IEC 60909 standard is independent of the load flow of a system and does not consider existing system conditions. However, it is based on the nominal values of the system. The method also uses correction factors for voltages and impedances (DIgSILENT Manual, 2005).

In the superposition method, the fault currents of the short circuit are determined by first computing the steady-state load flow before the inception of the short circuit. The fault current calculation of the second stage of the superposition method is done by determining the Thevenin equivalent of the network (Nedic *et al.,* 2007).

The minimum fault current for each lateral was obtained by calculating for a single-phase-to-ground fault with a non-zero fault resistance (20Ω) placed at the farthest point in the lateral away from the main feeder.

Table 4.6: **Comparison of DIgSILENT PowerFactory node voltage results with IEEE benchmark results (Kersting, 2004)**

	Comparison of IEEE 34 Node Test Feeder Results								
	DIgSILENT PowerFactory results/voltage p.u			IEEE published results/voltage p.u			Relative error (Erel.)/%		
Node	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**
800	1.047	1.05	1.052	1.0500	1.0500	1.0500	-0.2857	0.0000	0.1905
802	1.045	1.048	1.05	1.0475	1.0484	1.0484	-0.2387	-0.0382	0.1526
806	1.043	1.048	1.049	1.0457	1.0474	1.0474	-0.2582	0.0573	0.1528
808	1.012	1.024	1.029	1.0136	1.0296	1.0289	-0.1579	-0.5439	0.0097
810		1.024			1.0294			-0.5246	
812	0.976	0.999	1.005	0.9763	1.0100	1.0069	-0.0307	-1.0891	-0.1887
814	0.947	0.979	0.987	0.9467	0.9945	0.9893	0.0317	-1.5586	-0.2325
RG10	0.991	1.032	1.039	1.0177	1.0255	1.0203	-2.6236	0.6338	1.8328
850	0.991	1.032	1.039	1.0176	1.0255	1.0203	-2.6140	0.6338	1.8328
816	0.987	1.029	1.036	1.0172	1.0253	1.0200	-2.9689	0.3609	1.5686
818	0.987			1.0163			-2.8830		
820	0.979			0.9926			-1.3701		
822	0.978			0.9895			-1.1622		
824	0.983	1.024	1.032	1.0082	1.0158	1.0116	-2.4995	0.8072	2.0166
826		1.024			1.0156			0.8271	
828	0.982	1.023	1.031	1.0074	1.0151	1.0109	-2.5213	0.7782	1.9883
830	0.965	1.005	1.014	0.9894	0.9982	0.9938	-2.4661	0.6812	2.0326
854	0.964	1.005	1.013	0.9890	0.9978	0.9934	-2.5278	0.7216	1.9730
852	0.935	0.973	0.983	0.9581	0.9680	0.9637	-2.4110	0.5165	2.0027
RG11	0.982	1.024	1.037	1.0359	1.0345	1.0360	-5.2032	-1.0150	0.0965
832	0.982	1.024	1.037	1.0359	1.0345	1.0360	-5.2032	-1.0150	0.0965
858	0.98	1.023	1.036	1.0336	1.0322	1.0338	-5.1858	-0.8913	0.2128
834	0.978	1.02	1.033	1.0309	1.0295	1.0313	-5.1314	-0.9228	0.1648
842	0.978	1.02	1.033	1.0309	1.0294	1.0313	-5.1314	-0.9132	0.1648
844	0.978	1.02	1.033	1.0307	1.0291	1.0311	-5.1130	-0.8843	0.1843
846	0.978	1.02	1.034	1.0309	1.0291	1.0313	-5.1314	-0.8843	0.2618
848	0.978	1.02	1.034	1.0310	1.0291	1.0314	-5.1406	-0.8843	0.2521
860	0.977	1.02	1.033	1.0305	1.0291	1.0310	-5.1917	-0.8843	0.1940
836	0.977	1.019	1.033	1.0303	1.0287	1.0308	-5.1733	-0.9429	0.2134
840	0.977	1.019	1.033	1.0303	1.0287	1.0308	-5.1733	-0.9429	0.2134
862	0.977	1.019	1.033	1.0303	1.0287	1.0308	-5.1733	-0.9429	0.2134
838		1.019			1.0285			-0.9237	
864	0.98			1.0336			-5.1858		
XF10	0.968	1.011	1.023	0.9997	0.9983	1.0000	-3.1710	1.2722	2.3000
888	0.968	1.011	1.023	0.9996	0.9983	1.0000	-3.1613	1.2722	2.3000
890	0.885	0.927	0.939	0.9167	0.9235	0.9177	-3.4581	0.3790	2.3210
856		1.005			0.9977			0.7317	

Table 4.7: **Comparison of DIgSILENT PowerFactory line current result with IEEE benchmark results (Kersting, 2004)**

		Comparison of IEEE 34 Node Test Feeder Results								
		DIgSILENT PowerFactory results			IEEE published results			Relative error (Erel.)/%		
		Current (Amps)			Current (Amps)			(%)		
Node		Phase A	Phase B	Phase C	Phase A	Phase B	Phase C	Phase A	Phase B	Phase C
800	802	49.39	44.03	41.63	51.56	44.57	40.92	-4.2087	-1.2116	1.7351
802	806	49.40	44.03	41.63	51.58	44.57	40.93	-4.2264	-1.2116	1.7102
806	808	49.91	41.92	39.93	51.59	42.47	39.24	-3.2564	-1.2950	1.7584
808	810		1.19			1.22			-2.4590	
808	812	49.55	40.88	39.95	51.76	41.30	39.28	-4.2697	-1.0169	1.7057
812	814	49.69	40.95	39.98	51.95	41.29	39.33	-4.3503	-0.8234	1.6527
814	RG10	49.79	41.01	39.99	52.10	41.29	39.37	-4.4338	-0.6781	1.5748
RG10	850	46.37	38.39	37.02	48.47	40.04	38.17	-4.3326	-4.1209	-3.0128
850	816	46.37	38.39	37.02	48.47	40.04	38.17	-4.3326	-4.1209	-3.0128
816	818	13.07			13.02			0.3840		
816	824	33.79	38.42	37.02	35.83	40.04	38.17	-5.6936	-4.0460	-3.0128
818	820	13.07			13.03			0.3070		
820	822	10.80			10.62			1.6949		
824	826		2.96			3.10			-4.5161	
824	828	33.79	35.46	36.89	35.87	36.93	38.05	-5.7987	-3.9805	-3.0486
828	830	33.79	35.46	36.62	35.87	36.93	37.77	-5.7987	-3.9805	-3.0447
830	854	32.23	34.83	35.43	34.22	36.19	36.49	-5.8153	-3.7579	-2.9049
854	852	32.24	34.56	35.43	34.23	35.93	36.49	-5.8136	-3.8130	-2.9049
852	RG11	32.31	34.60	35.44	34.35	35.90	36.52	-5.9389	-3.6212	-2.9573
RG11	832	32.31	34.60	35.44	31.77	33.59	33.98	1.6997	3.0068	4.2966
854	856		0.32			0.31			3.2258	
832	XF10	11.01	11.09	11.22	11.68	11.70	11.61	-5.7363	-5.2137	-3.3592
832	858	20.69	22.20	23.14	21.31	23.40	24.34	-2.9094	-5.1282	-4.9302
858	834	20.16	21.97	22.85	20.73	23.13	24.02	-2.7496	-5.0151	-4.8709
858	864	0.14			0.14			0.0000		
834	860	10.51	8.93	10.01	11.16	9.09	10.60	-5.8244	-1.7602	-5.5660
834	842	13.99	15.60	14.85	14.75	16.30	15.12	-5.1525	-4.2945	-1.7857

		DIgSILENT PowerFactory results			IEEE published results			Relative error (Erel.)/%		
		Current (Amps)			Current (Amps)			(%)		
	Node	Phase A	Phase A	Phase B	Phase C	Phase A	Phase B	Phase C	Phase A	Phase B
842	844	13.99	15.60	14.85	14.74	16.30	15.12	-5.0882	-4.2945	-1.7857
844	846	9.45	9.18	9.30	9.83	9.40	9.40	-3.8657	-2.3404	-1.0638
846	848	9.39	9.21	9.65	9.76	9.40	9.78	-3.7910	-2.0213	-1.3292
860	836	4.07	5.94	3.58	4.16	5.96	3.60	-2.1635	-0.3356	-0.5556
836	840	1.42	2.22	1.66	1.50	2.33	1.75	-5.3333	-4.7210	-5.1429
836	862		2.14			2.09			2.3923	
862	838		2.14			2.09			2.3923	
888	890	65.96	66.75	66.78	69.90	70.04	69.50	-5.6366	-4.6973	-3.9137

Table 4.8: **Node voltage relative error vs. IEEE result**

Relative Error (%)	Phase A	Phase B	Phase C
Minimum	-0.0307	0.0000	0.0097
Maximum	-5.2032	-1.5586	2.3210
Average	3.1509	0.7719	0.8746

Similarly, the maximum branch fault current was obtained for a three-phase fault with a 0Ω fault resistance placed at the closest point in the lateral section. In situations where the line segments were single phase, a single phase-to-ground fault was calculated as appropriate. Table 4.9 shows the short circuit current values obtained for the main feeder and at the laterals.

The maximum and minimum short circuit current obtained using the IEC 60909 method is compared to the superposition method. From Table 4.10, it was observed that there were differences in the short circuit currents obtained from using the IEC 60909 and that of the superposition method.

Although, the superposition method is said to give realistic result because a load flow is conducted prior to the calculation (Manual DIgSILENT, 2005), the IEC 60909 method makes use of an equivalent voltage source and voltage/impedance correction factors to calculate the maximum and minimum short-circuit current. These currents correspond to the total curve and minimum melt time required for the fuse-relay coordination in the next sub-section. Thus, the IEC 60909 method was used in this thesis.

Table 4.9: **Short circuit currents for the main feeder and at the laterals**

Nr.	Section nomenclature	Min. fault current/(A)	Max. fault current/(A)
1.	Main Feeder	641.58	2027.98
2.	L.808	390.54	823.34
3.	L.816	163.93	294.16
4.	L.824	189.39	284.38
5.	L.854	156.02	242.76
6.	L.832	91.81	718.76
7.	L.858	107.46	146.04
8.	L.834	102.93	154.13
9.	L.836/1	103.05	151.47
10.	L.836/2	102.66	151.47

4.6 Simulation

After the modelling of the test feeder, 1780 fault cases were simulated on the feeder. These 1780 cases represent steady-state conditions and fault cases involving different fault types at different line sections simulated with varying fault resistance, fault inception angles, and load angles.

Also, 35 steady state cases comprising of various load angles, load switching, and capacitor switching were also simulated. In addition, a total of 492 simulations were done for the modified case studies (DG1, DG2, DG3, and line extension).

Table 4.10: **Comparison of the short circuit currents using IEC 60909 and superposition method**

Nr.	Section	IEC 60909		Superposition Method	
		Min. Fault Current/(A)	Max. Fault Current/(A)	Min. Fault Current/(A)	Max. Fault Current/(A)
1.	Main Feeder	641.58	2027.98	672.74	1993.21
2.	L.808	390.54	823.34	423.71	794.33
3.	L.816	163.93	294.16	183.54	283.80
4.	L.824	189.39	284.38	210.95	274.14
5.	L.854	156.02	242.76	175.07	233.70
6.	L.832	91.81	718.76	85.86	714.90
7.	L.858	107.46	146.04	119.16	137.70
8.	L.834	102.93	154.13	114.52	149.83
9.	L.836-1	103.05	151.47	114.19	147.09
10.	L.836-2	102.66	151.47	114.62	147.08

4.6.1 Base Case

Steady-state and faulted case simulations were carried out and monitored at node 800. The result was exported in ASCII format and was used as the input to the wavelet transform decomposition algorithm for subsequent implementation of the fault detection and diagnosis algorithm.

The parameters used to simulate the steady state of the network include:

- Steady-state simulations at different load angles (0^o, 30, 45, 60^o, and 90^o).
- Steady-state simulations with load switching
- Steady-state simulations with capacitor switching

These simulations were carried out to discriminate between transients due to faults and that due to switching conditions like load switching, capacitor switching, and load angle variation. The fault cases involved dynamic electromagnetic transient simulation of different fault types including Single Phase-to-ground (1 Ph.-G), two Phase (2 Ph.), two phase-to-ground (2 Ph.-G), and three phase (3 Ph.) faults. These simulations were carried out at different locations of the main feeder, and on the laterals. Various fault resistances and fault inception angle *(* θ_f *)* were used. The fault inception angle θ_{fA} is the phase angle of phase A voltage at the fault inception time. Total number of fault cases was 1780.

Since the waveform is periodic, it is sufficient to study the effect of fault inception angle by using values between 0^o and 90^o.

The ten fault types include:

1. Single-phase to ground faults (SL-G)
    a. Phase A to ground fault (A-G)
    b. Phase B to ground fault (B-G)
    c. Phase C to ground fault (C-G)
2. Double-phase to ground faults (2Ph.-G)
    a. Phase A-B to ground fault (A-B-G)
    b. Phase A-C to ground fault (A-C-G)
    c. Phase B-C to ground fault (B-C-G)
3. Phase to phase faults (2Ph.)
    a. Phase A-B (A-B)
    b. Phase A-C (A-C)
    c. Phase B-C (B-C)
4. Three-phase fault (3 Ph.) A-B-C

In the transient simulation procedure, the system conditions were initialized by making use of the *‘calculate initial conditions’* function of DIgSILENT PowerFactory. Afterwards, electromagnetic transient simulation for unbalance three phase was started. The simulation initialized at 0.00s and lasted for 10 cycles. Fault inception was on the 4th cycle. The fault duration used was 3 cycles. The time duration during which the fault current flows in the device can be estimated directly from the faulted voltage and current waveforms by looking at the voltage sag or the high current magnitude.

The waveforms are generated with 128 samples per cycle based on recommendations in Power System Relaying Committee Report, (2006) and a total of 1780 fault cases were simulated on the main feeder and at the laterals in order to build a database that truly represents the characteristics of the network. These test cases were used to develop, train, and evaluate the proposed method.

The top plot in Figure 4.10 shows the three phase line current values during capacitor switching. The bottom plot shows the zero sequence plot. From this figure, spikes due to the generation of transients as a result of the switching on and off of the capacitor are noticeable. The effect is quite pronounced at time t = 0.10833s. t corresponds to the time the capacitor(s) are being switched off.

Thus, the method to be developed in this thesis should be able to distinguish normal operation events like load switching and capacitor switching.

Figure 4.11 shows an AB-G fault with fault resistance of 20Ω at lateral 846-848. The three phase line currents and zero sequence components of the various simulations were exported to MATLAB for signal processing using Discrete Wavelet Transform (DWT).

The zero sequence components are used to differentiate between ground and aerial faults since 2Ph. and 2Ph.-G faults have similar features. Voltage sags and overcurrent are usually experienced at the faulted phase(s) during fault. The magnitude of these sags or overcurrent depends on the type of fault and the system parameters.

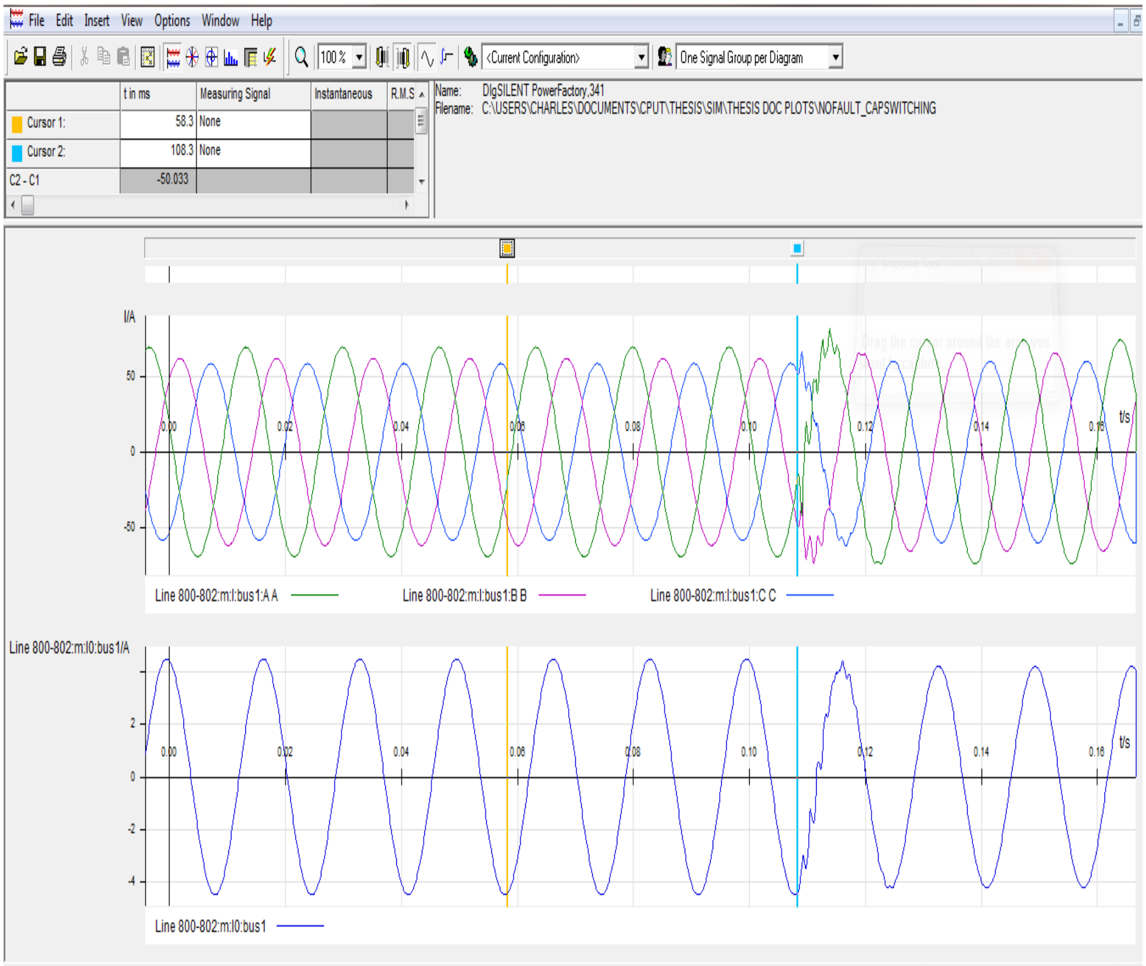


Figure 4.10: Current waveforms for capacitor switching at nodes 844 and 848

The start of recordings can be prompted either by using edge trigger or duration trigger (Power System Relaying Committee report, 2006; Davilla, 2011). Edge triggering is often times associated with fixed length recording from the rising edge of the trigger for a pre-defined length. This is as shown in Figure 4.12(a), while duration triggers capture the length of the entire fault duration within a record.

For edge triggering, a device will typically start a recording on the rising edge of the pulse. The recording will continue for a pre-determined length of time. The length is the sum of the amount of pre-fault data, fault length, and the post-fault data (Power System Relaying Committee Report, 2006; Davilla, 2011).

An illustration of duration triggering method is shown in Figure 4.12(b). A recording is initiated on the rising edge of a trigger. Post fault data are captured once the trigger de-asserts.

The total length is decided by the length of pre-fault data, the event duration, the post-fault data, and the preset total record length. Duration triggering is adopted in this thesis for all the simulations.

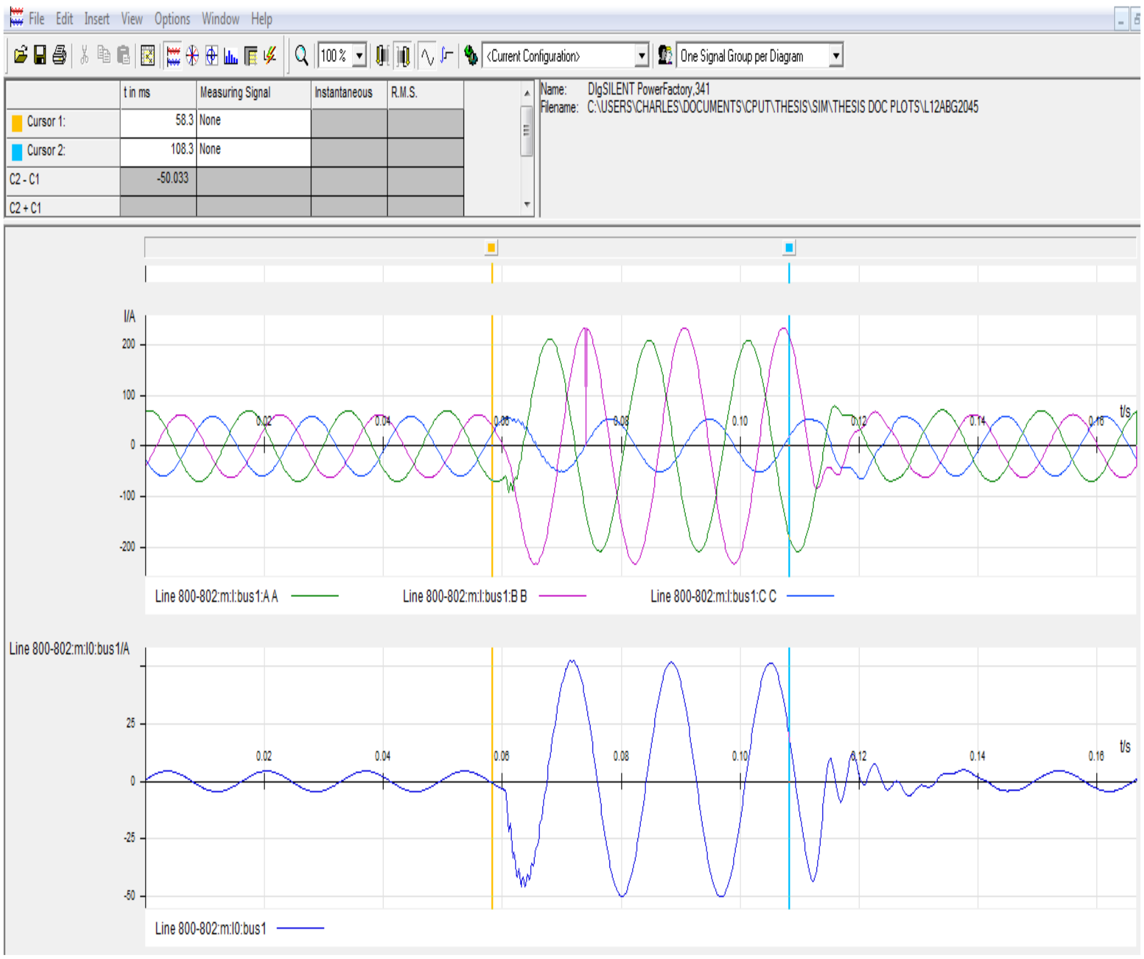


Figure 4.11: **Current waveforms for AB-G fault at lateral 846-848**

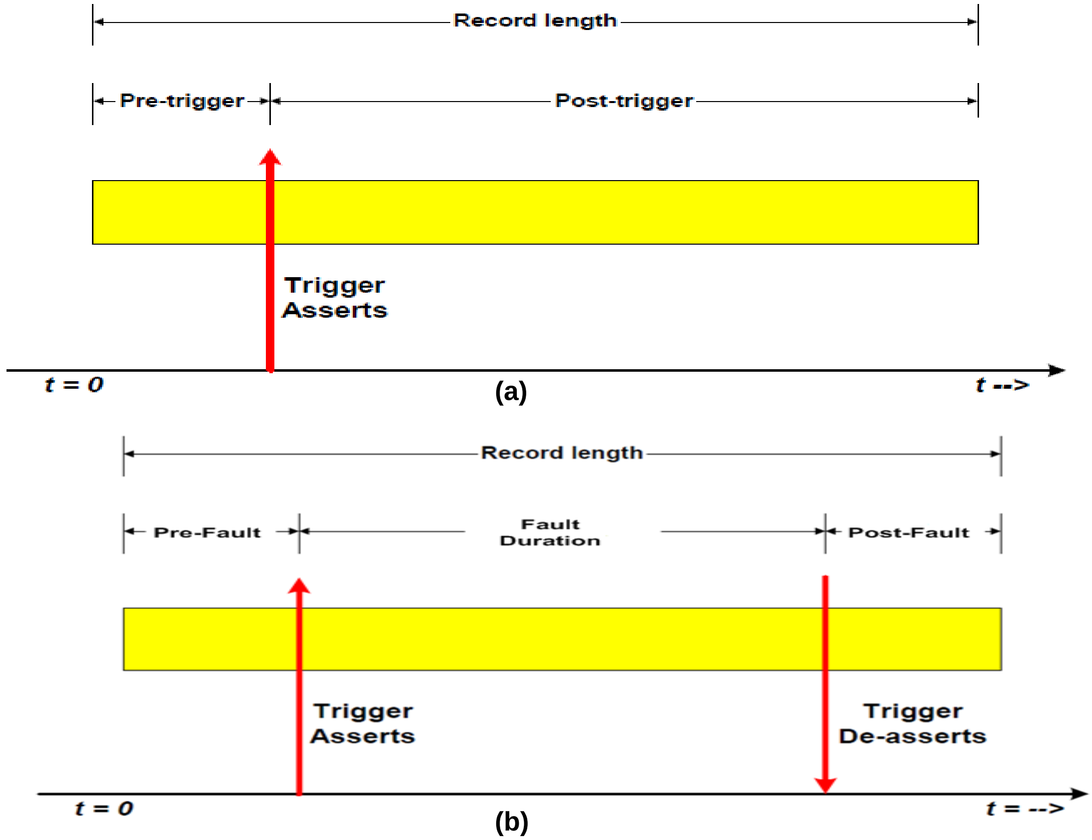


Figure 4.12: **Disturbance record triggering (a) Edge triggered; and (b) Duration triggered (Power System Relaying Committee Report, 2006)**

4.6.2 DG Case Studies

The modified case studies involving the integration of DGs into the IEEE 34 node test feeder and the effect of line extension are:

- DG1 case study: maximum load + 20% DG penetration level installed at node 840
- DG2 case study: maximum load + 20% DG penetration level installed at node 844
- DG3 case study: maximum load + 10% DG penetration level installed at nodes 840 and 844 respectively.

Thus, node 840 (along the main line) and node 848 (one of the laterals) were used with a 20% penetration level one at a time. Furthermore, another test case was simulated with smaller DGs co-located in the network at nodes 840 and 848 respectively. A total of 113 fault cases and 10 steady state simulations were done per DG case study. Figure 4.13 is a model of the DG3 case study. The blue components in this figure indicate the distributed generators.

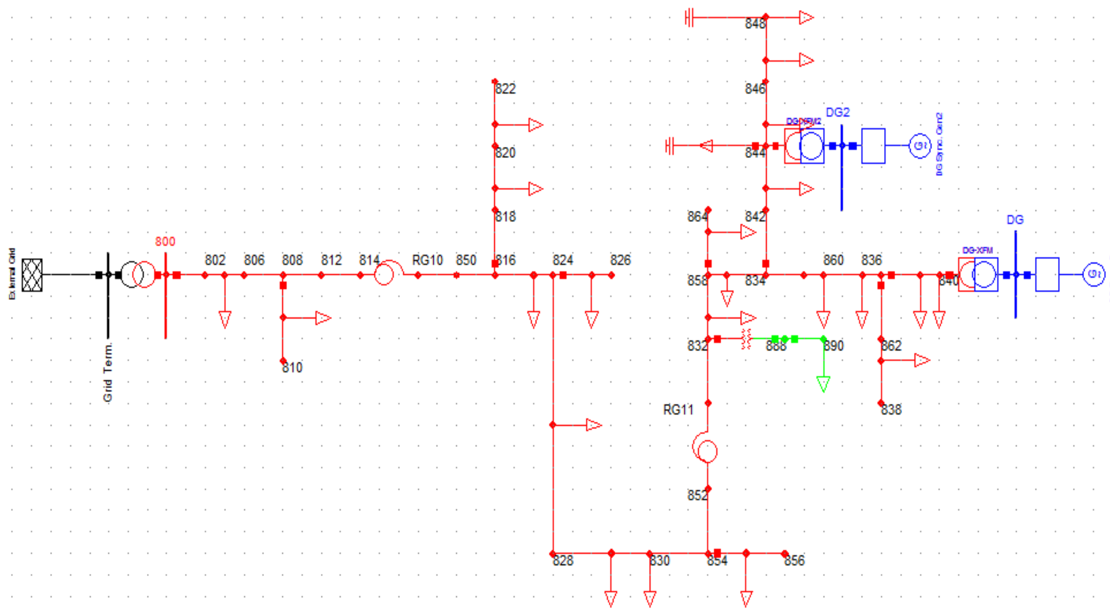


Figure 4.13: **Single line diagram of the DG3 case study in DPF**

4.6.3 Line Extension Case Study

This case study involved the extension of the line at node 840. Load flow analysis was conducted and the result was compared to the load flow result of the base case. Also, short circuit studies were done for this case study.

Afterwards, simulations comprising of 113 fault cases and 10 steady-state cases were carried out.

4.6.4 Results for the modified case studies

Load flow calculation of the case studies (DG1-3), line extension case study, and the base case were analyzed and compared. A plot of the voltage profile is given in Figure 4.14. The voltage profile plot shows the impact of the integration of DG and line extension on the feeder.

Similarly, a plot of the maximum short circuit currents at various nodes is shown in Figure 4.15. Nodes 836/1 refer to lateral 836-862, while Node 836/2 refers to lateral 836-840. The short circuit plot shows an increase in the maximum short circuit current at various nodes in the feeder as a result of the integration of DGs. The integration of two generators in DG3 case study had a far reaching influence and the highest maximum short circuit current for nodes 800, 808, 816, 824, 854, 832, and 858 was recorded in this case study. DG2 case study with a generator integrated to the test feeder at node 844 recorded the highest maximum short circuit currents for the nodes in its vicinity i.e. nodes 834, 836/1, and 836/2.

Table 4.11: Comparison of DIgSILENT PowerFactory node voltage result for DG1, DG2, and DG3

		IEEE 34 Node Test Feeder Results										
		DIgSILENT PowerFactory load flow results for modified case studies								**DG3 relative error (erel.)**		
	DG1 voltage (p.u)			**DG2 voltage (p.u)**			**DG3 voltage (p.u)**			**%**		
Node	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**
800	1.047	1.048	1.054	1.047	1.048	1.054	1.047	1.048	1.054	0.0000	0.1905	-0.1901
802	1.045	1.046	1.052	1.045	1.047	1.053	1.045	1.046	1.052	0.0000	0.1908	-0.1905
806	1.044	1.045	1.052	1.044	1.046	1.052	1.044	1.045	1.051	-0.1043	0.2863	-0.1907
808	1.020	1.028	1.035	1.022	1.030	1.037	1.020	1.027	1.035	-0.8096	-0.2930	-0.5831
810		1.028			1.030			1.027		0.0000	-0.2930	
812	0.992	1.009	1.016	0.996	1.012	1.020	0.992	1.008	1.015	-1.5616	-0.9009	-0.9950
814	0.970	0.993	1.001	0.975	0.998	1.006	0.970	0.992	0.999	-2.1781	-1.3279	-1.2158
RG10	1.000	1.029	1.036	0.996	1.025	1.031	0.991	1.006	1.015	0.0000	2.5194	2.3099
850	1.000	1.029	1.036	0.996	1.025	1.031	0.991	1.006	1.015	0.0000	2.5194	2.3099
816	0.997	1.027	1.034	0.993	1.023	1.029	0.997	0.992	0.999	-0.9870	3.5957	3.5714
818	0.997			0.993			1.026			-3.8493		
820	0.989			0.985			0.989			-0.9790		
822	0.988			0.984			0.988			-0.9780		
824	0.994	1.023	1.030	0.990	1.019	1.026	0.994	1.022	1.028	-1.0813	0.1953	0.3876
826					1.019			1.022		0.0000	0.1953	
828	0.993	1.023	1.029	0.990	1.019	1.025	0.993	1.021	1.028	-1.0802	0.1955	0.2910
830	0.981	1.008	1.015	0.979	1.006	1.012	0.982	1.007	1.013	-1.6405	-0.1990	0.0986
854	0.981	1.008	1.015	0.979	1.005	1.012	0.981	1.006	1.013	-1.6388	-0.0995	0.0000
852	0.961	0.984	0.990	0.961	0.983	0.989	0.962	0.982	0.987	-2.5245	-0.9250	-0.4069
RG11	0.992	1.011	1.021	0.983	1.002	1.011	0.992	1.009	1.018	-0.9820	1.4648	1.8322
832	0.992	1.011	1.021	0.983	1.002	1.011	0.992	1.009	1.018	-0.9820	1.4648	1.8322
858	0.991	1.010	1.020	0.983	1.001	1.010	0.992	1.008	1.002	-1.1760	1.4663	3.3205
834	0.990	1.009	1.018	0.983	1.002	1.011	0.991	1.006	1.015	-1.2714	1.3725	1.7425
842	0.990	1.009	1.018	0.982	1.000	1.009	0.991	1.006	1.015	-1.2714	1.3725	1.7425
844	0.990	1.008	1.018	0.982	1.000	1.009	0.991	1.006	1.014	-1.2714	1.3725	1.8393
846	0.990	1.008	1.018	0.983	1.000	1.009	0.991	1.006	1.015	-1.2714	1.3725	1.8375

		IEEE 34 Node Test Feeder Results										
		DIgSILENT PowerFactory load flow results for modified case studies								DG3 Relative error (Erel.)		
	DG1 VOLTAGE (P.U)			DG2 VOLTAGE (P.U)			DG3 VOLTAGE (P.U)			%		
Node	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**	**Phase A**	**Phase B**	**Phase C**
848	0.990	1.008	1.018	0.983	1.000	1.009	0.991	1.006	1.015	-1.2714	1.3725	1.8375
860	0.990	1.008	1.018	0.982	0.999	1.008	0.990	1.005	1.014	-1.2701	1.4706	1.8393
836	0.990	1.008	1.018	0.981	0.999	1.008	0.990	1.005	1.014	-1.2701	1.3739	1.8393
840	0.990	1.009	1.018	0.981	0.999	1.008	0.990	1.005	1.014	-1.2701	1.3739	1.8393
862	0.990	1.008	1.018	0.981	0.999	1.008	0.990	1.005	1.014	-1.2701	1.3739	1.8393
838		1.008			0.999			1.005		0.0000	1.3739	
864	0.990			0.983			0.992			-1.1760		
XF10	0.989	1.008	1.018	0.983	1.002	1.011	0.989	1.006	1.016	-2.0328	0.4946	0.6843
888	0.989	1.008	1.018	0.980	0.990	1.008	0.989	1.006	1.016	-2.0328	0.4946	0.6843
890	0.905	0.924	0.933	0.897	0.915	0.924	0.906	0.922	0.930	-1.8585	0.5394	0.9585
856		1.008			1.005			1.006			-0.0995	

The maximum short circuit currents for the line extension case study did not show any remarkable deviation from that obtained for the base case.

Table 4.11 shows the comparison of the load flow result of DG1, DG2, and DG3 case studies with the base case. Only the relative error of DG3 is included in this table because the highest relative deviation was recorded for this case study. The maximum deviation recorded as shown in Table 4.11 was 3.5957% for phase B of node 816.

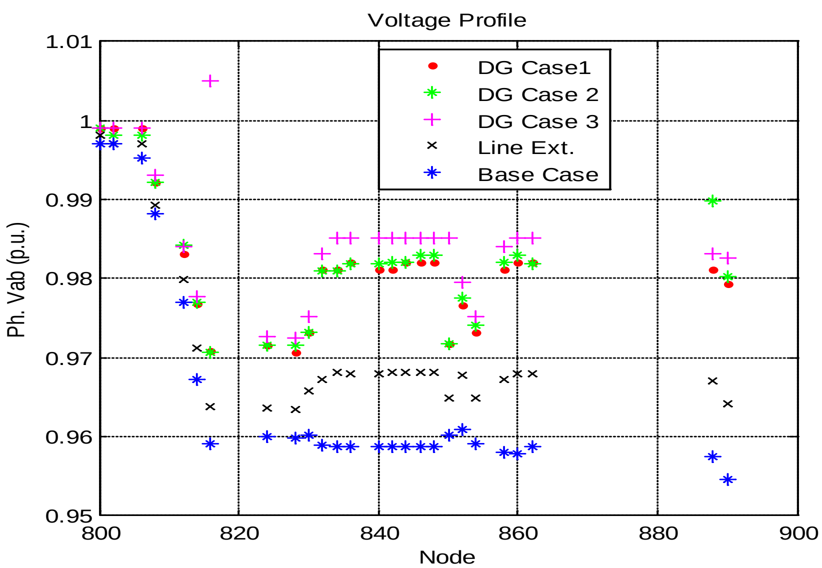


Figure 4.14: **Node voltage profile for the base case and modified case studies**

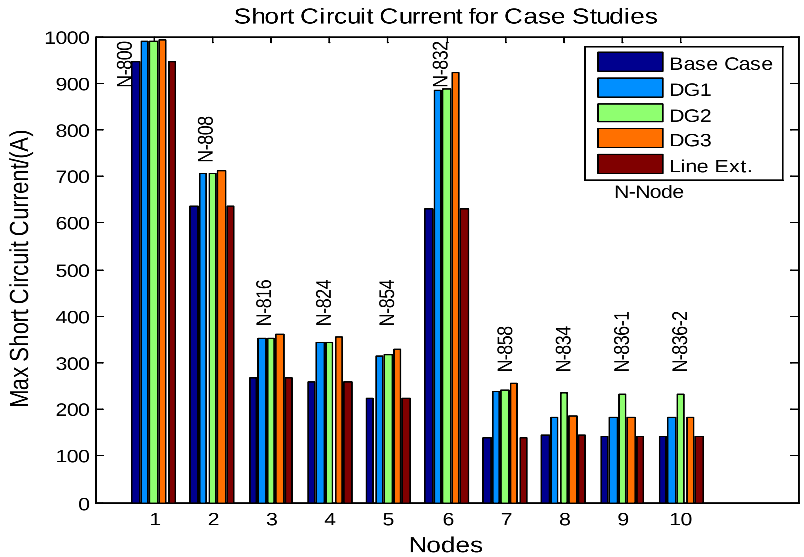


Figure 4.15: **Short circuit fault current for various the base case and modified case studies**

Figure 4.16 is a plot showing the fault current waveform for phase A during A-G fault on Line 808-812 (30% of the length of the feeder). The fault resistance was 10Ω, while the fault inception angle was 0^o. The fault was simulated for various load angles of 0^o, 30^o, 45^o, 60^o, and 90^o.

From the figure, it can be seen that the load angle influences the minimum, maximum, and RMS values of the line current. Figures 4.17-4.19 show similar plots for DG1, DG3, and Line extension case studies respectively.

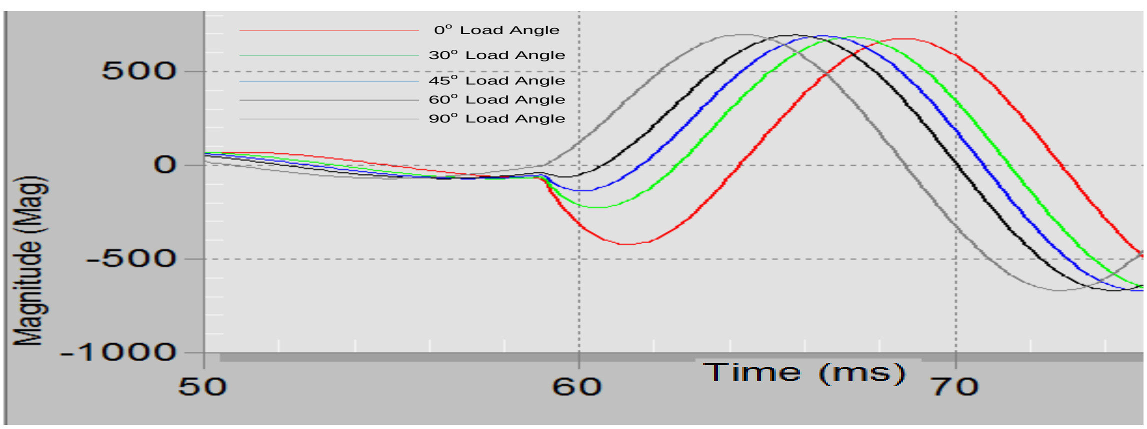


Figure 4.16: Base case: **A-G fault at Line 808-812 (30% of the length of the feeder)**

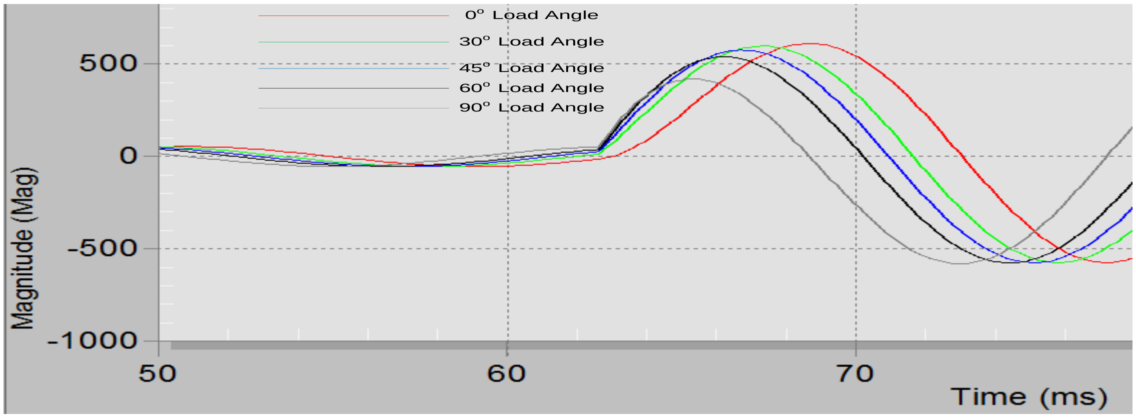


Figure 4.17: DG1 case study: A-G fault at Line 808-812 (30% of the length of the feeder)

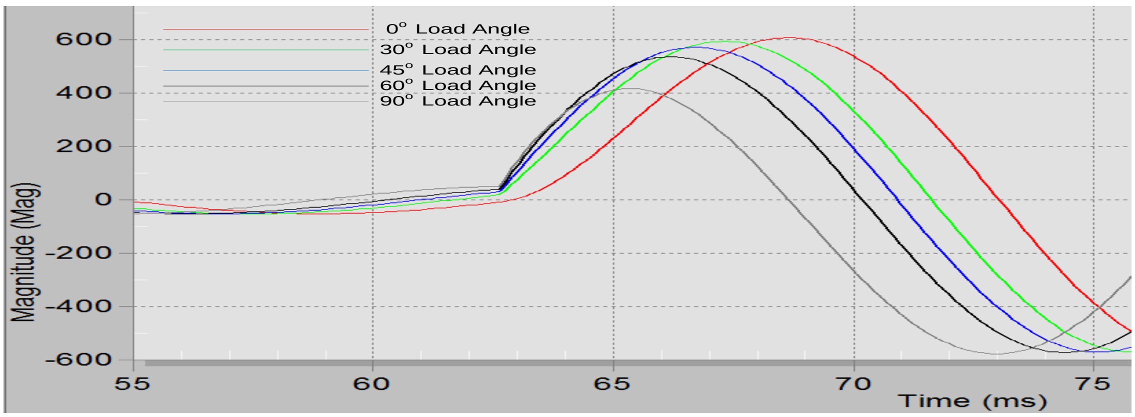


Figure 4.18: DG3 case study: A-G fault at Line 808-812 (30% of the length of the feeder)

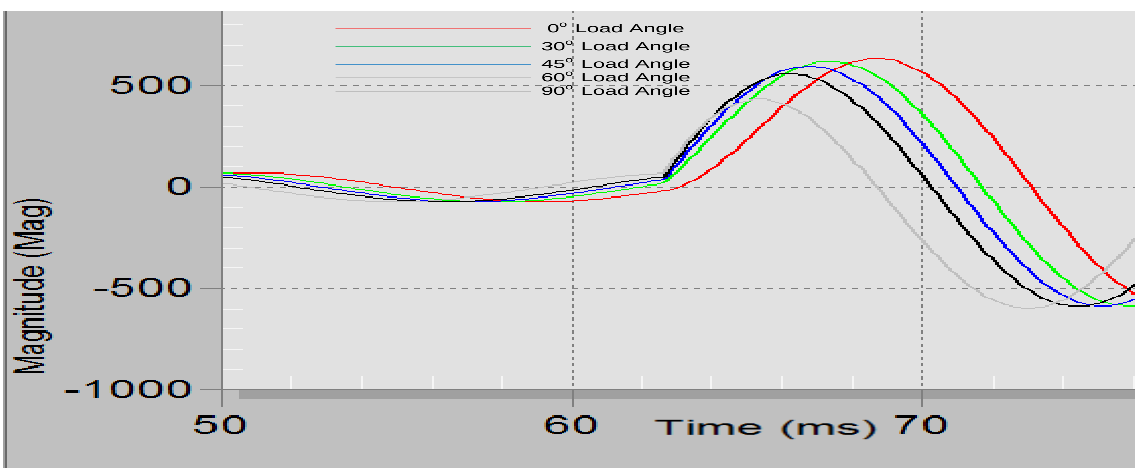


Figure 4.19: Line extension case study: A-G fault at Line 808-812 (30% of the length of the feeder)

Simulation results showing a summary of the faulted phase for a B-G fault at various fault resistances with fault inception angles of 0^o at Line 860-836 is given in Table 4.12. Note, Line 860-836 was chosen because it is at the extreme end of the feeder.

Table 4.12: **Waveform summary for a B-G fault at line 860-836 (95% of the length of the main feeder) with varying fault resistances**

Fault resistance/Ω	Min fault current	Max fault current	Abs max fault current	Avg fault current	RMS fault current
0.0	-249.243	234.54	249.243	73.506	98.8159
0.5	-247.833	233.87	247.833	73.391	98.6008
2.5	-243.040	231.48	243.040	72.936	97.7548
5.0	-238.541	228.71	238.541	72.373	96.7242
10.0	-230.653	223.68	230.653	71.279	94.7338
20.0	-217.366	214.32	217.366	69.252	91.0534
100.0	-159.095	161.14	161.135	58.603	72.2123

Similarly, simulation results showing a summary of the faulted phase for a B-G fault at various fault inception angles with fault resistances of 0Ω at Line 860-836 is given in Table 4.13. The result obtained shows that the fault inception angle influences the magnitude of the fault current, with 90^{o} fault inception angle having the highest fault current magnitude.

Table 4.13: **Waveform summary for a B-G fault at line 860-836 (95% of the length of the main feeder) with varying fault inception angles**

Fault inception angle/^{o}	Min fault current	Max fault current	Abs max fault current	Avg fault current	RMS fault current
0	-238.541	228.71	238.541	72.373	96.7242
30	-240.081	228.81	240.081	72.396	97.0393
45	-240.485	228.84	240.485	72.374	97.0177
60	-239.788	228.94	239.788	72.103	96.6715
90	-228.728	230.11	230.112	76.2	101.05

4.7 Post-Simulation Operations for Data Transfer

The simulation results obtained in DIgSILENT PowerFactory can be exported to other software packages for further analysis. In this thesis, it was necessary to export the simulation results to MATLAB for the development of the Hybrid Fault Detection and Diagnosis (HFDD) method.

The results in DIgSILENT PowerFactory can be obtained by creating a virtual instrument panel to record the waveform of the quantity of interest before running the simulation. In this case, three phase and zero sequence fault currents.

The procedure followed in creating a virtual instrument panel is given below:

- Right click on the ‘Grid’ tab in DIgSILENT
- *Insert page > create new page*
- Select *‘Virtual Instrument Panel’*
- Right click on the new page created > *Create VI > Subplot*
- Double click on the subplot sheet created > *Insert the result file* that contains the quantities of interest > Choose the element and variables to be plotted

Afterwards, the data in the waveform can be exported by right-clicking on the simulation waveform plot and choosing the *‘Export’* function. The file transfer methods used in this thesis was ASCII and COMTRADE and can be selected by using the *‘Export to’* menu as shown in Figure 4.20.

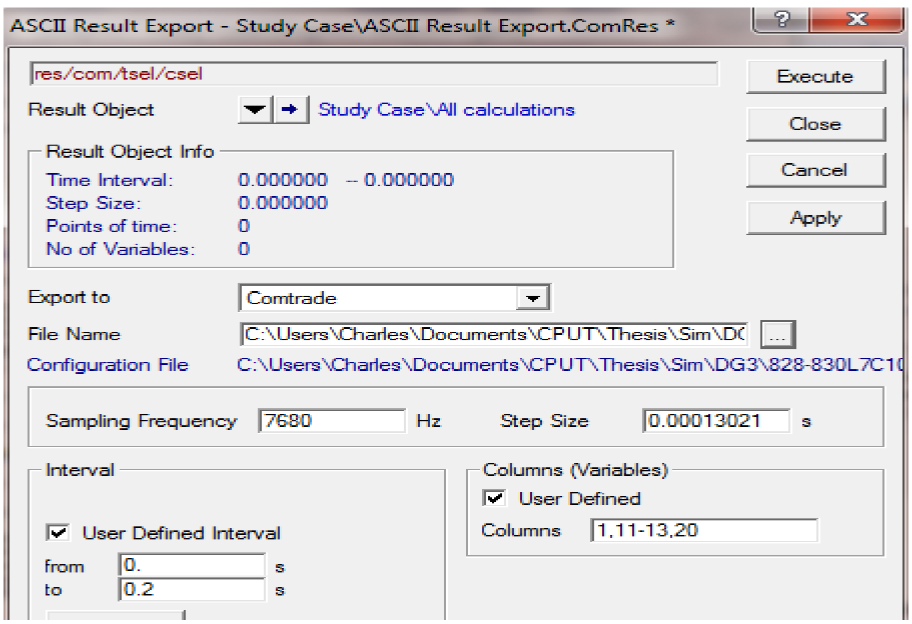


Figure 4.20: File transfer of results from DIgSILENT PowerFactory to other software environment

MATLAB command *‘tdfread’* was used afterwards to read the ASCII file into MATLAB workspace. The *‘tdfread’* command reads in text and numeric data from tab-delimited files. This was perfect for the task because of the nature of the ASCII file that was exported from DPF. This gives an option to select the file to be read. The file was then read and variables were created in the MATLAB work space.

After the variables of the exported waveform are created in MATLAB workspace, the decomposition of these variables is done through the use of the developed script file in Appendix C.1. This algorithm executes the Discrete Wavelet Transform (DWT) processing and results in the generation of detail and approximate coefficients. These coefficients would be used in Chapter Five in the design of the various algorithms in the Hybrid Fault Detection and Diagnosis (HFDD) Method.

4.8 Discussion of the Results

The results in Tables 4.6 and 4.7 showed close agreement with that obtained by the IEEE subcommittee despite the difference in the modelling software and the consequent difference in components used. Phases B and C showed a very low relative deviation of 0.7719% and 0.8746% respectively.

The load flow result comparison of the base case and DG1, DG2, and DG3 case studies in Table 4.11 showed that the DGs did provide voltage support to the network. This impact is noticeable in the node voltages especially at nodes 888 and 890 which previously had undervoltage respectively. This positive impact is known as the system support benefits of the integration of DGs. The consequence of which is a disturbance to the model and also, the contribution to fault current by the DG especially for faults in close proximity to it. This increase in fault current is shown in Figure 4.15.

Furthermore, Figure 4.16 shows plots of A-G fault at various load angles. The plot showed that load angle variation would have an impact on the maximum and minimum current magnitudes including the RMS current. The same observation applies to Figures 4.17-4.19.

Table 4.12 presents a summary of the effect of varying fault resistances. As expected, there is an inverse proportionality (Ohm’s law) between fault resistance and fault current. As the fault resistance increases, the fault current decreases. Thus, the fault signature reduces. Faults at high resistances might go undetected and would require a special algorithm capable of detecting high impedance fault. Faults with high resistances would form part of the dataset for the developed method in Chapter Five. This also applies to fault location. Faults closer to the substation would have a high fault current compared to the fault current at line 836-840 for instance.

Similarly, the variation of fault inception angle had a noticeable effect on the fault current especially at fault inception angle of 90^o.

The above-mentioned parameters were chosen in order to have a wide diversity of data incorporating possible and worse case scenarios. Figure 4.10 showed that the on and off action of capacitors also generate transients which could be mistaken as fault. Thus, it is required that provisions be made to differentiate such actions from faults, and tag them as ‘no-fault’ conditions.

4.9 Conclusion

The chosen test feeder was modelled in DPF to generate data to be used for the development of the Hybrid Fault Detection and Diagnosis (HFDD) method. Steady state and transient simulation studies were conducted on the feeder. Load flow calculations were derived from the steady state studies and were validated with the published IEEE load flow results.

Furthermore, short circuit studies were conducted and used to determine the fault current level. Ten types of faults were simulated at various three-phase line segments, while single phase-to-ground faults were simulated at various single-phase line segments. For each type of fault at each fault location, simulation was done for various fault resistances and fault inception angles. Also, the effect of the switching on and off of loads and capacitor banks was investigated.

In Chapter Five, the waveforms of the simulation cases were decomposed using wavelet transform techniques in MATLAB environment. The entropy and entropy per unit of the resulting detail coefficients served as the input to a rule-based decision algorithm for fault detection and classification, and also to artificial neural networks for fault section identification and fault location respectively.

CHAPTER FIVE

DEVELOPMENT OF A HYBRID FAULT DETECTION AND DIAGNOSIS (HFDD) METHOD

5.1 Introduction

This chapter describes the development of a Hybrid Fault Detection and Diagnosis (HFDD) method using the IEEE 34 node benchmark test feeder. The proposed method is based on the combination of Discrete Wavelet Transform (DWT), algorithm based on decision-taking rules, and Neural Networks.

Current and voltage signals contain characteristic signatures which signify the existence or absence of fault, the fault type, and the fault location (Butler *et al.,* 1997). The proposed method in this thesis makes use of entropy and entropy per unit of DWT detail decomposition coefficients obtained from three phase and zero sequence currents. These currents contain important information capable of giving details about the status of the network.

The HFDD method is implemented in a modular form by means of software subroutines implemented in MATLAB. The selection of the three-phase and the zero sequence currents as the data input for the proposed method is based on the fact that overcurrent protection is the most used form of protection in distribution networks. Thus, line current quantities are readily available. Also, the spike/surge in fault current waveforms is usually easier to visualize than voltage sag/collapse during faults.

The HFDD method is divided into five (5) algorithms:

- DWT and entropy calculation
- Fault detection
- Fault classification
- Fault section identification
- Fault location

The inputs to the various algorithms of the HFDD method are derived from DWT decomposition of the waveform signals exported from DIgSILENT PowerFactory. Level-1 coefficients of the detail decomposition are used to calculate the wavelet entropy which is then used as input to the fault detection algorithm. The use of level-1 detail coefficients was based on several experimentations carried out to determine the best decomposition level suitable for fault detection. The other algorithms make use of a formulation proposed in the thesis. This formulation is known as the wavelet entropy per unit (using level-5 detail coefficients), and it serves as inputs to the fault classification, fault section identification, and fault location algorithms respectively.

The fault detection algorithm is activated first and on detection of a fault condition, the fault classification algorithm is triggered to perform the fault type and faulted phase(s) tasks. The selection of the appropriate fault type can be difficult with evolving fault characteristics and magnitudes (IEEE Guide, 2005). Thus, it is very important to have an accurate process for the fault classification task since the output at this stage would be used as the trigger to activate the appropriate fault section identification and fault location ANNs respectively.

The fault detection and fault classification modules are based on rules written in MATLAB to take decisions. The inputs to the fault detection module are entropy values from the three phase and zero sequence currents obtained from the DWT decomposition. Entropy per unit is used as the input to the fault classification module. Similarly, the fault section identification and fault location modules are based on artificial neural network pattern classifiers and function approximators respectively. The implementations of the algorithms are illustrated in Figure 5.1.

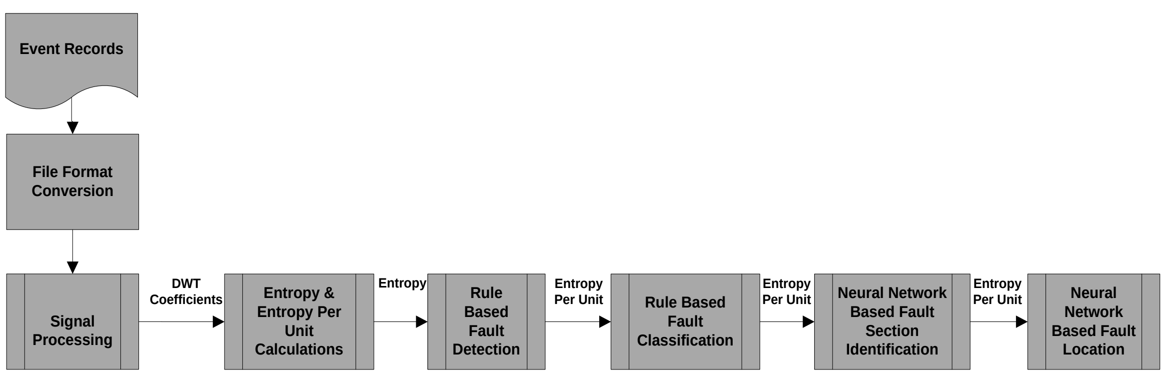


Figure 5.1: Breakdown of the fault detection and diagnosis method

The input currents required for the HFDD method can be extracted from event records or waveform plots from simulations. Event records can be acquired from a myriad of equipment. A couple of such equipment is illustrated in Figure 5.2.

Level-1 DWT decomposition coefficients from the three phase and zero sequence currents serve as the input to the fault detection module, the output from this module triggers the fault classification module if a fault event has occurred.

The output from the fault classification module determines which of the ANNs to use for the fault section identification and fault location tasks respectively. The fault classification, fault section identification, and fault location modules make use of entropy per unit of level-5 DWT detail coefficients.

Figure 5.2: **Digital devices for recording events**

The process followed in the design of the various algorithms in the thesis is shown below in Figure 5.3. The essential processes involved include system design, software design and development, and system integration/verification.

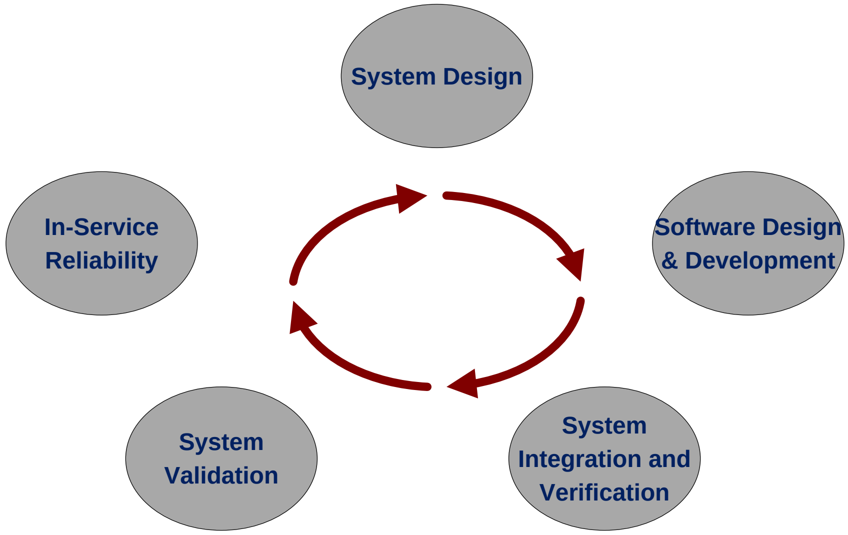


Figure 5.3: **System design and implementation process**

This chapter presents the development of the various algorithms that make up the Hybrid Fault Detection and Diagnosis (HFDD) method. Discrete wavelet decomposition of the inputs is carried out with several wavelet families in order to determine the most suitable family to use. Also, the best levels of decomposition were selected. Furthermore, the calculation of the statistical measures to use was done. Other procedures presented are the data pre-processing, feature extraction, and feature selection tasks respectively. Lastly, the development of the algorithms for fault detection, fault classification, fault section identification, and fault location is detailed herein.

5.2 Discrete Wavelet Transform and Statistical Computations

5.2.1 Introduction

Wavelet Transform has been proven to be very efficient in power system transient analysis (Borghetti *et al.,* 2006; Salim *et al.,* 2008; Bhowmik *et al.,* 2008; Panigrahi and Panda, 2009; Ekici *et al.,* 2009; Malathi *et al.,* 2011; Zhengyou *et al.,* 2011; Baqui *et al.,* 2011).

Signal analysis using Discrete Wavelet Transform (DWT) involves the processing of the signal of interest by using a combination of low pass (h) and high pass (g) filters. The coefficients of these filters have been pre-determined in various works by the proponents of such wavelet family. For instance, the filter coefficients for the Daubechies wavelet family was determined in previous works spanning several years by Ingrid Daubechies (Daubechies, 1992). Thus, the number of filter coefficients for the db4 low pass and high pass filters has been predetermined by her to be four each.

5.2.2 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) decomposition of a signal using the low pass and high pass filters results to two components. The approximate coefficients correspond to the low pass filter, while the detail coefficients correspond to the high pass filter.

The detail and approximate coefficients for the next level of decomposition is obtained by passing the downsampled coefficients of the previous decomposition through the high pass and low pass filters respectively.

Signal analysis using waveform transform has been said to be influenced by the mother wavelet used (Peretto *et al.,* 2005).

Research results by Bollen *et al.,* 2006; Zhang and Kezonovic, 2007 show that the best mother wavelet for power system analysis is the Daubechies family. Bhowmik *et al.,* 2008 used Daubechies db3 in their proposed method. Ekici *et al.,* 2009; Panigrahi and Panda, 2009; Zhengyou *et al.,* 2011 used db4. Baqui *et al.,* 2011 used db5, while Salim *et al.,* 2008 utilised db8. Malathi *et al.,* 2011 carried out an analysis with db1-db4 Daubechies mother wavelets, with db2 showing the best performance. However, Borghetti *et al.,* 2006 used the Mortlet mother wavelet. The best mother wavelet to use depends on which of the mother wavelet exhibits similar characteristics with the signal of interest (Malathi *et al.,* 2011).

This thesis would analyse the Daubechies family mother wavelets of db2, db3, db4, db5, and db8. Other mother wavelets considered are the symlet and Coiflet families. The most suitable mother wavelet will be selected and would form the basis for the computation of the statistical measures used. In the same vein, the best level of decomposition to use would also be determined. Software sub-routines for these are given in Appendices C.1, C.2, and C.3.

5.2.3 Statistical Computations

As earlier mentioned, the criteria that were used in the thesis are the wavelet entropy and the entropy per unit calculations. These calculations are carried out by software sub-routine as given in Appendices C.1 and C.2.

Given that the wavelet energy of the signal at scale (level) j instant k is:

$$E_{jk} = \left| D_j(k) \right|^2 \tag{5.1}$$

The sum of all the signal's energy at scale j, $k = 1,2,\ldots,N$.

$$E_j = \sum_{k=1}^{N} E_{jk} \tag{5.2}$$

The relative wavelet energy is:

$$P_{jk} = \frac{E_{jk}}{E_j} \tag{5.3}$$

WEE for each scale or level can be defined as:

$$WEE_j = -\sum_k P_{jk} \log P_{jk} \tag{5.4}$$

where D_j are the DWT detail coefficients at scale j.

The criterion proposed for fault classification, fault section identification, and fault location in the thesis is the phase entropy per unit λ_{p5}.

This is given as:

$$\lambda_{p5} = \frac{-\sum_k P_{jkp} \log P_{jkp}}{(-\sum_k P_{jkA} \log P_{jkA}) + (-\sum_k P_{jkB} \log P_{jkB}) + (-\sum_k P_{jkC} \log P_{jkC})}, \qquad p \in (A,B,C) \tag{5.5}$$

where r_{a5}, r_{b5}, r_{c5} are the computed WEE_j of level-5 details for phases A, B, C respectively, and $p \in (A,B,C)$. λ_{p5} is the phase entropy calculated per unit for a given fault.

Similarly, the Standard Deviation σ and Mean Absolute Deviation (MAD) of the detail coefficients of the selected mother wavelet would be calculated. The accuracy of the wavelet entropy formulation would be compared with that obtained from Standard Deviation σ and Mean Absolute Deviation (MAD).

Standard Deviation σ is a statistical measure of distribution or spread in a data set and it is derived from the square root of the variance in a data set. Mean Absolute Deviation (MAD) is the mean of the absolute deviations of the data set from the mean of the data. It shows the statistical dispersion of a data set.

The standard deviation of the signal at scale j, instants k is:

$$\sigma_j = \left(\frac{1}{N-1} \sum_{k=1}^{N} (D_{jk} - \mu_j)^2 \right)^{1/2} \tag{5.6}$$

The Mean Absolute Deviation of a signal is given as:

$$MAD = \frac{1}{N} \sum_{k=1}^{N} \left| D_{jk} - \mu_j \right| \tag{5.7}$$

where D_{jk} is the detail coefficient at scale j, instant k, μ_j is the mean at scale j, and N is the number of instants.

Figure 5.4 is a breakdown showing the DWT and calculations used in the HFDD method.

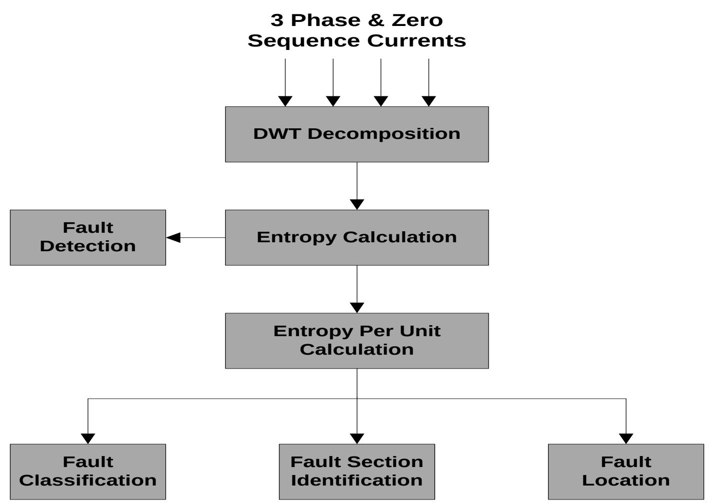


Figure 5.4: **Breakdown of the discrete wavelet transform and calculations used**

5.3 Data Pre-processing and Feature Extraction

5.3.1 Data Pre-Processing

Data pre-processing operates on raw data by transforming the data into a format that will be more easily and effectively processed. Pre-processing simplifies the input signal by reducing the volume of the input data for the process. This helps to reduce the training time and complexity of neural networks.

When a fault occurs on a transmission or distribution line, different frequency component signals like the DC offset, fundamental frequency and non-fundamental frequency components are produced due to the fault transients. It has been shown that all these frequency components change as the fault position and fault type vary (Song, 1997).

Data in real-life recordings are usually noisy. The use of DWT algorithms ensures that there will be only negligible variations of noise components in the fifth level. This is because the magnitudes of the transient signals by the faults are much larger than whatever noise may be present in the fifth level of the DWT decomposition.

5.3.2 Feature Extraction

The main idea of feature extraction is to reduce the amount of information, either from the original waveform or from its transformation format. The process of feature extraction consists of finding the distinctive waveform parameter, with significant information, that can represent the fundamental characteristics of the signal.

The waveform plot of the three phases and zero sequence currents from the distribution system is obtained from the simulation. The options in DIgSILENT PowerFactory for exporting the simulation results are: ASCII, COMTRADE, and elm files. The ASCII format is used to export the results in this part of the thesis. The exported waveforms are read into MATLAB by using the *tdfread* function and decomposed using db4 level-6 discrete wavelet transform into its detail and approximation coefficients respectively.

Daubechies-4 (db4) is one of the most used wavelet in power system disturbance analysis and it was chosen for this research empirically. The experiments done demonstrates its orthogonality, compact support in the time domain, and its good performance in power system studies as reported by (Ekici *et al.,* 2008; Perera & Rajapaske, 2008; Panigrahi & Pandi, 2009; Costa *et al.,* 2010).

The filters high pass (g) and low pass (h) of the db4 have four coefficients as earlier mentioned. These are:

$$g_1 = 0.1294,\ g_2 = 0.2241, g_3 = 0.8365, g_4 = 0.4830 \tag{5.8}$$

$$h_1 = -0.4830,\ h_2 = 0.8365, h_3 = -0.2241, h_4 = -0.1294 \tag{5.9}$$

WT is an effective tool for analyzing transient signals. Its features of extraction and representation properties can be used to analyze various transient events in power signals. These transient features are accurately captured and localized in both the time and the frequency domains. Table 5.1 presents the frequency range existing in the wavelet decomposition levels.

Table 5.1: Frequency range existing in wavelet decomposition levels for 7.68kHz Sampling frequency

DB4 Decomposition Level	**Frequency Range/(Hz)**	
Nr.	**Approximation *(cA)***	**Detail *(cD)***
1	0-1920	1929-3840
2	0-960	960-1929
3	0-480	480-960
4	0-240	240-960
5	0-120	120-240
6	0-60	60-120

5.3.3 Feature Selection using Wavelet Energy Spectrum Entropy

Feature selection involves the selection of the relevant information from the relevant features that summarize the most important aspects of the available data. Wavelet combined entropy is said to make full use of localized feature at time-frequency domains thereby analyzing fault signals more efficiently (Zhengyou *et al.,* 2011). Thus, the wavelet energy spectrum entropy is used as feature selection.

Entropy has found application in thermodynamics, physical chemistry, statistical mechanics, quantum mechanics, information theory, etc. In recent times, the use of entropy in power system research is growing (Ekici, 2008; Zhengyou *et al.,* 2011; Adewole & Tzoneva, 2012). Input data processed by this method through a neural network could save a great deal of training time, and it makes the classification more precise and faster. Wavelet energy entropies have shown excellent results in power system fault detection and classification over other statistical methods like standard deviation and mean absolute deviation (Adewole & Tzoneva, 2012).

In order to determine at which level of decomposition these high frequency appear for the considered cases, several experiments were performed using waveforms obtained from the simulations with different fault types and fault locations. The experiments were carried out for decomposition levels-1 through level-6. Level-1 detail was chosen afterwards as the decomposition level of interest for fault detection features. Conversely, level-5 detail was selected for fault classification because the best classification was obtained with it. This is supported by the fact that at level-5, the dominant non-frequency transient that is generated by the faults is observable within the frequency range 120Hz to 240Hz. Hence, it was unnecessary to utilize coefficients from other scales.

Similarly, level-5 was also selected for the FSI and FL tasks. Extensive simulation studies show that the most relevant detail in the DWT decomposition for this network is at level-5 since they provide the relevant characteristics needed to analyze the information in the waveforms. However, the db4 level-5 details from the decomposition of these waveforms resulted to 47 coefficients. The direct use of these coefficients as inputs would result to a complex and time consuming analysis. This thesis utilized the wavelet energy spectrum entropy as the feature selection method. The derivation is given in equations (5.1)-(5.4).

5.4 Fault Detection Algorithm

5.4.1 Feature Selection for the Fault Detection Algorithm

The Wavelet Energy Entropy (WEE) per phase obtained from equation (5.4) is used for the fault detection task. The WEE computed for the three phase and zero sequence currents serves as input to the rule based algorithm. It should be noted that the decomposition level to use are peculiar to each particular system. Thus, in order to optimize performance, a rules-based system should be designed with the ability to modify the rules used for the analysis and decision making (Bekker & Keller, 2008). The rules for the fault detection task are based on thresholds determined by simulations based on the benchmark model for the cases with and without fault.

5.4.2 Rules for the Fault Detection Algorithm

Rule-based classifier consists of a set of rules, used in a given order during the prediction process to classify the likely event that might have occurred. By using a set of rules, a complex task is broken down into a collection of simple decisions.

The rules perform comparisons of the WEE per phase of the three phase and zero sequence current, and take decisions based on pre-determined logic.

This can involve some of the following (Bekker, 2008):

- Algebraic and Calculus (Differentiation and Integration) on scalars, vectors and matrices.
- Complex algebraic operators for complex scalars, complex vectors and complex matrices.
- Data lookup, reference and format manipulation operators.
- Wavelet Transform, Fourier Transform, and harmonic component operators.
- Boolean logic and Fuzzy Logic (influence) operators.
- Statistical computations.

In the proposed method in the thesis, the fault detection module makes use of MATLAB subroutines to perform level-6 daubechies-4 (db4) mother wavelet decomposition on the obtained waveform. The WEE of phases A, B, and C (r_{a1}, r_{b1}, r_{c1}) of the level-1 detail coefficients are computed based on equations (5.1)-(5.4). Level-1 is used for fault detection because its coefficients are influenced by the high frequency noise components of the signal.

Fault or ‘no fault’ condition is then established on comparison with a predetermined threshold (ζ_d). The comparison is done for each phase. ζ_d is carefully chosen to ensure that the algorithm would be able to accurately discriminate between faults and normal switching events. In this particular case, ζ_d is set to 1.75. The choice of 1.75 was done after extensive simulation and comparison of the entropy values for faulted cases and for 'no fault' cases.

The rules for taking decision are:

If $r_{a1} > \zeta_d$ or $r_{b1} > \zeta_d$ or $r_{c1} > \zeta_d \rightarrow$ Fault is detected.

Else → No Fault.

The above rules show that fault would be detected once any of the phases exceeds the pre-defined threshold. The software subroutines are given in Appendices C.4 and C.5 respectively. Figure 5.5 presents the block diagram of the fault detection algorithm.

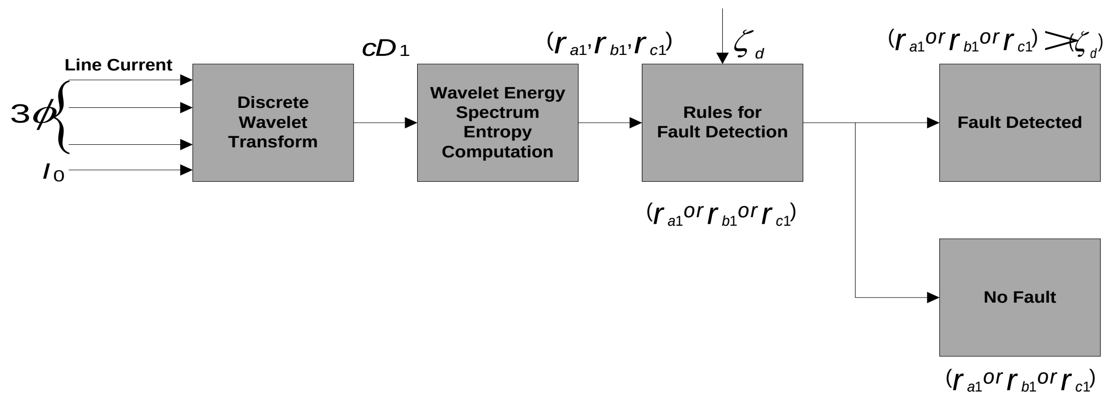


Figure 5.5: **Block diagram of the fault detection algorithm**

In this thesis, the results of the proposed method are obtained for the base case fault conditions, the case studies involving systems parameter changes, introduction of Distributed Generation (DGs), and overhead line extension.

5.5 Fault Classification Algorithm

5.5.1 Feature Extraction for Fault Classification Algorithm

When fault is detected, the fault classification module is triggered. This is implemented in two stages. The first is fault type classification, followed by faulted phase(s) determination. The rule-based classification is based on the fact that each fault has its own characteristic feature or signature by which the faulted phases can be identified.

Therefore, the fault types are grouped into four categories: Single Phase-to-ground (1 Ph.-g), Two Phase (2 Ph.), Two Phase-to-ground (2 Ph.-g), and three phase (3 Ph.) faults.

5.5.2 Rules for Fault Classification Algorithm

Rules are then formulated to define which particular category the fault belongs to. For fault classification, the three phase and zero sequence currents are decomposed into 6 levels using db4 mother wavelet. The coefficients from level-5 detail are used to compute the entropies $r_{a5}, r_{b5}, r_{c5}, r_{I05}$ based on equations (5.1)-(5.4).

where $WEE_{a5} = r_{a5}$, $WEE_{b5} = r_{b5}$, $WEE_{c5} = r_{c5}$, and $WEE_{I05} = r_{I05}$.

The patterns observed through exhaustive simulations were used to draw up the rules for the algorithm. After the identification of the fault type, the phase(s) included in the fault is determined. To do so, fault indices based on level-5 decomposition are used.

The rules used for the fault classification are detailed below:

R1: if ($\lambda_{a5} < \lambda_{b5}$) & ($\lambda_{a5} < \lambda_{c5}$) → A-g Fault

R2: if ($\lambda_{b5} < \lambda_{a5}$) & ($\lambda_{b5} < \lambda_{c5}$) → B-g Fault

R3: if ($\lambda_{c5} < \lambda_{a5}$) & ($\lambda_{c5} < \lambda_{b5}$) → C-g Fault

R4: if ($\lambda_{a5} < \lambda_{c5}$) & ($\lambda_{b5} < \lambda_{c5}$) & $r_{I0a5} > \zeta_{ca}$ → AB Fault

R5: if ($\lambda_{b5} < \lambda_{a5}$) & ($\lambda_{c5} < \lambda_{a5}$) & $r_{I0b5} > \zeta_{cb}$ → BC Fault

R6: if ($\lambda_{a5} < \lambda_{b5}$) & ($\lambda_{c5} < \lambda_{b5}$) & $r_{I0c5} > \zeta_{cc}$ → CA Fault

R7: if ($\lambda_{a5} < \lambda_{c5}$) & ($\lambda_{b5} < \lambda_{c5}$) & $r_{I0a5} < \zeta_{ca}$ → AB-G Fault

R8: if ($\lambda_{b5} < \lambda_{a5}$) & ($\lambda_{c5} < \lambda_{a5}$) & $r_{I0b5} < \zeta_{cb}$ → BC-G Fault

R9: if ($\lambda_{a5} < \lambda_{b5}$) & ($\lambda_{c5} < \lambda_{b5}$) & $r_{I0c5} < \zeta_{cc}$ → CA-G Fault

R10: *else* (rules 1-9) → 3 Ph. Fault.

where ζ_{ca}, ζ_{cb}, and ζ_{cc} are the classification thresholds (derived from the zero sequence entropy) for classifying 2Ph. and 2Ph.-G faults. ζ_{ca} = 3.85; ζ_{cb} = 3.3; ζ_{cc} = 3.5.

Figure 5.6 is a block diagram of the fault classification algorithm from the line current inputs to the fault type and faulted phase(s) classification.

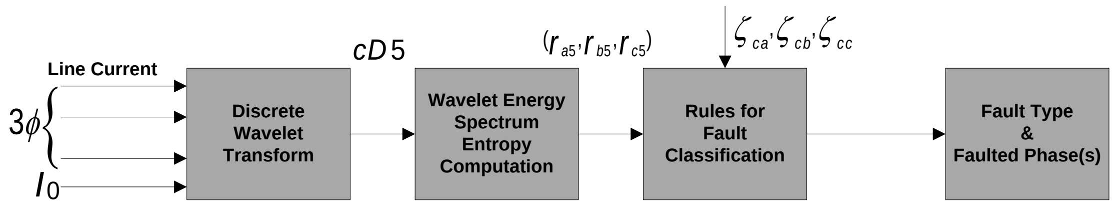


Figure 5.6: Block diagram of the fault classification algorithm

On the basis of the above considerations and rules, the overall flow chart of the proposed algorithm for fault detection and classification is shown in Figure 5.7. The reason for going through a classification sequence as shown in the flowchart is based on the probability of the occurrence of the various fault types, with single phase-to-ground faults having the highest occurrence. The software subroutines are given in Appendices C.6, C.7, and C.8 respectively.

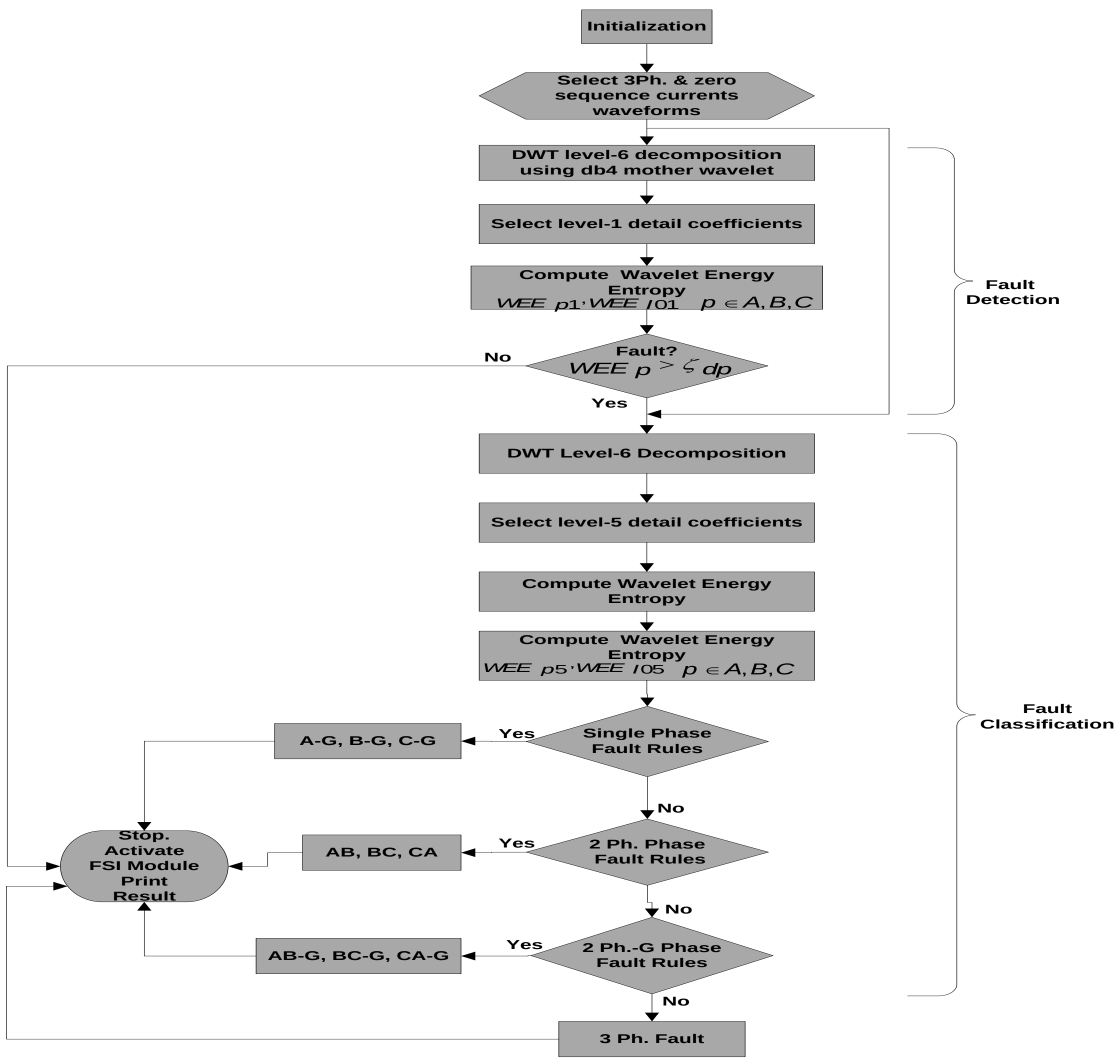


Figure 5.7: Flowchart of the proposed fault detection and classification algorithms

5.6 Fault Section Identification Algorithm

As highlighted in Chapter One, the existing methods for the determination of the faulted section and fault location in distribution networks are usually determined by trial and error. Usually, the trouble shooting starts from the information provided by the customer. In order to isolate the faulty segment, the line is energized section by section until the protective relay trips the feeding circuit breaker and the faulty section is identified. This procedure may be repeated several times which is time consuming and also exposes the equipment to additional stress from faults. It is vital that fault analysis and identification be carried out quickly for a fast restoration of the system after fault. For the fault section identification task in this thesis, neural networks are used as shown in Figure 5.8.

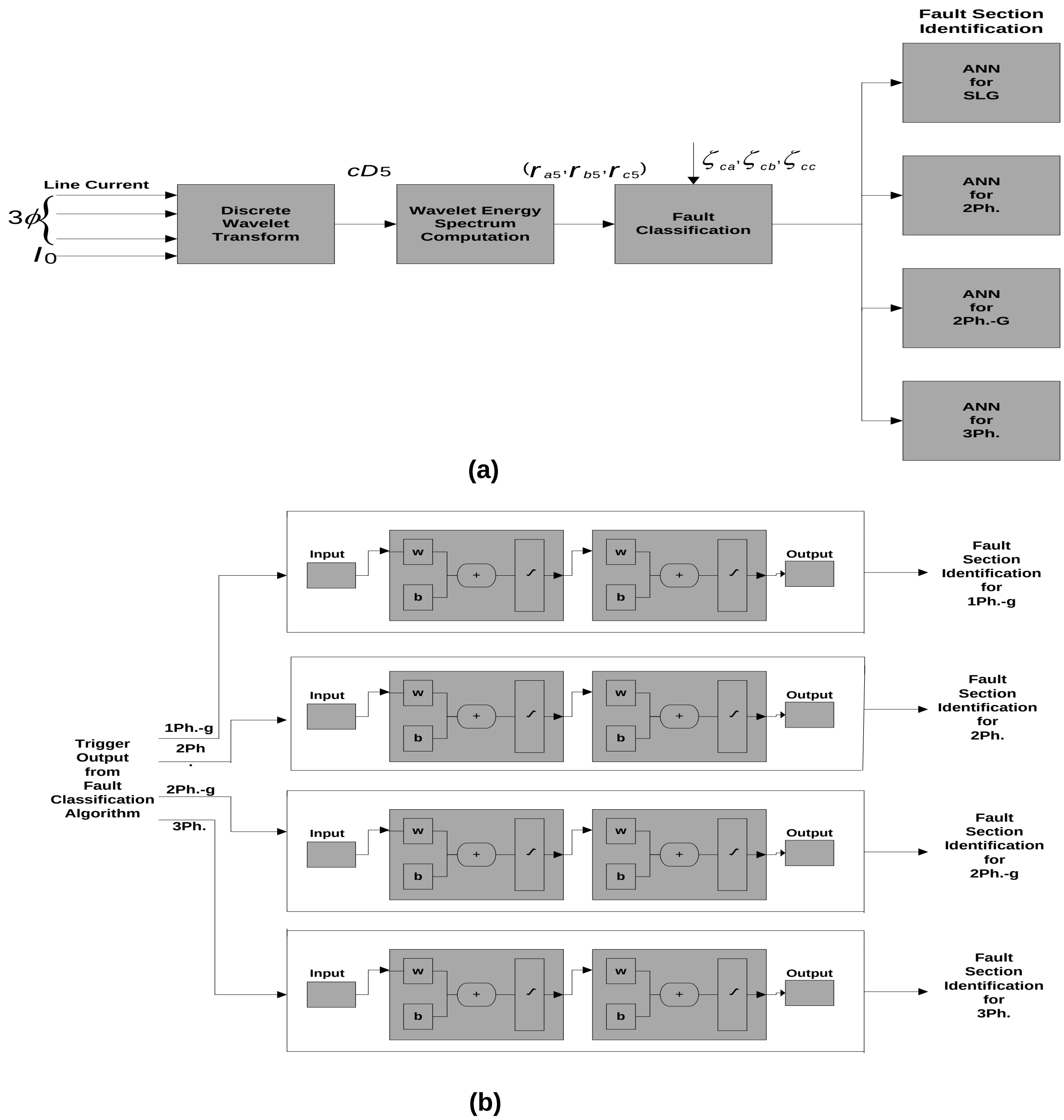


Figure 5.8: (a) Functional blocks of the proposed fault section identification algorithm; and **(b)** Implementation of the fault section identification algorithm

Four ANNs are utilized for fault section identification, each corresponding to the fault classes (i.e. 1Ph-G, 2Ph., 2Ph.-G., 3Ph.). The use of ANN has been proven to be appropriate in power system fault analysis since white noise does not have any adverse effect on the performance of neural networks (Aggarwal *et al.,* 1994, Jain *et al.,* 2009).

5.6.1 Design Process for the Fault Section Identification ANNs

The neural networks for the proposed fault section identification task are designed to solve a classification problem.

The design process of the ANN fault section identification module goes through the following procedure:

1. Extraction and selection of the appropriate features.
2. Preparation of the training vectors.
3. Definition of the classes for the classification task.
4. Determination of the ANN structure.
5. NN training.
6. Post-training evaluation of the performance of the trained ANN.
7. Use the network.

In this research, three methods were used in the implementation of the NN fault section identification module.

These are:

- Split sample or hold-out validation method
- Early-stopping method
- Regularization method

The Split-sample or hold-out validation is carried out by retaining part of the dataset as a test set. This test set must be a representative sample of the cases that must be generalized to. After training, the test set is used to test the ability of the trained network to generalize, and the error on the test set provides an unbiased estimate of the generalization error. 70% of the dataset is used for training, while 30% is used for testing.

In the early stopping method, the dataset is divided into three parts: Training set (70% of the dataset), validation set (15% of the dataset), and test set (15% of the dataset).

Regularization is a method for improving generalization and it involves modifying the performance function. This method is also tested by an independent ‘test set’. This test set must also be a representative sample of the cases.

5.6.2 Feature Selection

Generally, normalization of the input and target vectors in the dataset is carried out in order to make the ANN training faster and also avoid outputs that will be stuck in the local minima. The neural network can be considered as having a pre-processing block between the input and the first layer of the network and a post-processing block between the last layer of the network and the output. This is as shown in Figure 5.9.

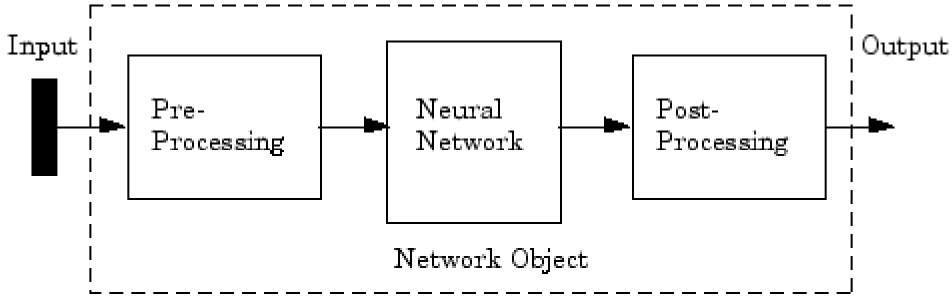


Figure 5.9: **Processing stages in neural network implementation**

The neural network training is performed by matrix manipulation of the inputs, outputs, weights, and biases. Vectors from a training set are presented to the networks sequentially. If the output of the network is correct, no change would be made. Otherwise, the weights and biases are updated according to the network's training algorithm.

Since this is a multi-task problem, it is preferable to decompose the problems into individual sub-problems where each sub-problem is dedicated to one task. Using an ANN for only one sub-problem makes it more powerful and increase the learnability of the ANN. Thus, the proposed fault section identification algorithm consists of multiple ANNs. This makes the algorithm faster, efficient, robust, and accurate.

Four different networks were finally selected to process different fault types. The particular fault section identification network is activated by the fault type classification module based on the fault type. The Neural Network Toolbox in MATLAB Version 7.12.0.635 (R2011a) was used in the design of the various ANNs of the proposed HFDD method. The initial weights and initial biases are taken as random values. The steps followed are given in Figure 5.10.

The four neural network classifiers are discussed in subsequent sub-sections. Emphasis is placed on the performance comparison of the neural networks in terms of the size of the neural network, the learning process, confusion matrix, Receiver Operating Characteristics (ROC) plot, regression coefficient, classification accuracy, and robustness.

The wavelet energy entropies obtained from the preceding section were collected and organized for training using Microsoft Excel. The task of fault section identification was accomplished by properly designing and developing the appropriate networks. Several architectures were investigated and a comparative report of the various architectures was carried out.

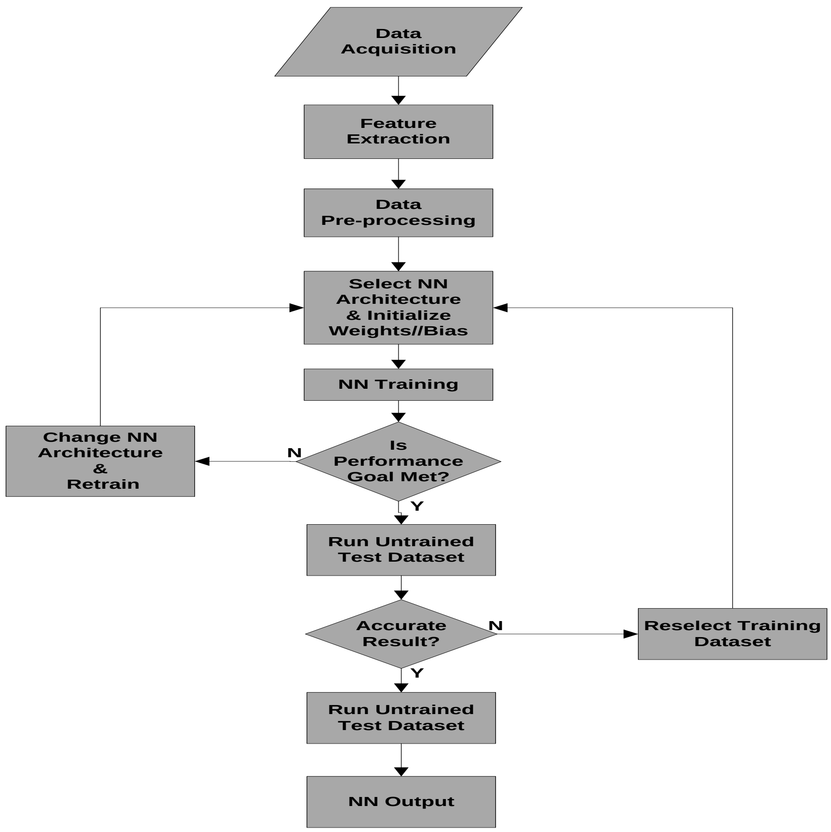


Figure 5.10: **Flow chart of NN training for fault section identification**

5.6.3 Network Architecture

The selection of the right architecture is of utmost importance in neural network implementation. At present, there are no formal ways of determining the size of a neural network. As the size of input and output were specified already, only the size of hidden layer was concerned. Choosing the number of hidden layers and the number of neurons is therefore difficult, because there are no generally acceptable theories. The solutions normally are found by means of empirical testing. One practical consideration is to try and minimize the size of the network while maintaining good performance. Although, smaller NNs cannot be trained to very small errors, they are however less prone to noise in the training data set and give better approximations (generalize better) for new patterns.

The rule of thumb in selecting the number of hidden layer/hidden layer neurons in neural networks is to ensure that the number of equations (N_{eq}) for a particular problem is far greater than the number of weights (N_w). This is necessary for an accurate and stable weight estimates that will generalize well to untrained data used for testing.

For an Input-Hidden-Output (*I-H-O)* node topology, the number of unknown weights (N_w) is given by:

$$N_w = (I+1) * H + (H+1) * O \qquad (5.10)$$

The number of training equations (N_{eq}) is:

$$N_{eq} = N_t * O \qquad (5.11)$$

For an *I-H-H-O* node topology, the number of unknown weights is given by:

$$N_w = \{(I+1) * H\} + \{(H+1) * H\} + \{(H+1) * O\} \qquad (5.12)$$

where I is the number of input vector(s) to the neural network, H is the number of hidden layer neurons, and O is the output of the network.

Different ANN structures were tested for various training strategies. A single hidden layer would suffice normally except if the task is complex, then more than one hidden layer would be necessary. A summary of the various ANN structures considered for 1Ph.-G, 2Ph., 2Ph.-G, and 3ph. faults is given in Chapter Six.

In MATLAB, the implementation of the neural networks is performed by means of matrix manipulation of the inputs, outputs, weights and biases vectors. Vectors from a training set are presented to the networks sequentially. The weights and bias of the network would be updated if the output of the network is different from the target dataset. The input datasets are arranged in a matrix form. Table 5.2 gives a breakdown of the training and test dataset used for 1Ph.-G, 2Ph., 2Ph.-G, and 3ph. ANNs respectively.

Thus, the training dataset for single phase-to-ground fault section identification consists of 500 four-element (4x500 matrix) input vectors and four-element target vectors. There are four elements in each target vector because four categories are associated with each input vector. The association table representing the faults sections is given in Table 5.3. This table shows the details of the classification desired.

Table 5.2: **Summary of the training and test datasets for fault section identification**

S/N	Fault Type	Size of Training Dataset	Size of Test Dataset
1	Single Phase-to-Ground	500	100
2	Two Phase	400	100
3	Two Phase-to-Ground	400	100
4	Three Phase	150	30

A coarse search for hidden layer (H) = 5:5:55 is used to determine the number of hidden layer neurons for the 1Ph.-G, 2Ph., 2Ph.-G, and 3ph. ANNs depending on the maximum number of neurons possible based on equations (5.10)-(5.12).

Table 5.3: **FSI association table representing the fault section class**

Nr.	Section	Classification Class			
1	Main Feeder	0	0	0	1
2	Lateral 808-810	0	0	1	0
3	Lateral 816-822	0	0	1	1
4	Lateral 824-826	0	1	0	0
5	Lateral 854-856	0	1	0	1
6	Lateral 832-890	0	1	1	0
7	Lateral 858-864	0	1	1	1
8	Lateral 834-848	1	0	0	0
9	Lateral 836-838	1	0	0	1
10	Lateral 836-840	1	0	1	0

5.6.4 Neural Network Training

The datasets selected for the neural network training are normalized to the range [-1, 1] before the training and testing. This is done by using equation (5.13).

$$p = 2x\frac{[p - \min(p)]}{[\max(p) - \min(p)]} - 1 \qquad (5.13)$$

where min ***(p)*** and max***(p)*** are the minimum and maximum values of the entire input space of input ***p***.

The initial weights and bias are of utmost importance in the neural network training. Incorrectly initialized weights lead to the saturation of neurons (Demuth *et al.,* 2004). Before the commencement of the training, the network was randomized in order to ensure that different weights and bias are used. The batch method of training was used.

The initialization of the weights and biases are repeated for 10 trials. The mean of these ten trials are tabulated and presented in Chapter Six. Also, performance plots of the best trial are presented for visualization.

The activation function used was the *tansig* for the hidden and output layers respectively. The selected activation functions are in accordance with the recommendations for neural network pattern recognition/classification tasks as given in Demuth *et al.,* 2004. The *logsimoid* and *purelin* activation function was also experimented upon, but were found to give poor performance.

Several training algorithm were experimented with. The conventional back propagation uses gradient descent algorithm for updating the weights. However, the gradient descent converges slowly. Only the results obtained for Resilient Back Propagation (RP), Scaled Conjugate Gradient (SCG), One Step Secant (OSS), Bayesian Regularization (BR), and Levenberg Marquardt (L-M) algorithms are included in this thesis due to space considerations. It is not possible to consider for training all the cases that an ANN-based fault section identification algorithm may actually encounter. Therefore, it is necessary to consider some representative fault situations and to train the network with data corresponding to these cases, such that the ANN gives correct output on testing even for those cases which it has never encountered before.

Network error and computational time were noted after each training trial. Table 5.4 gives a summary of the parameters used to generate the training dataset. During network training, the output of the network is computed and compared with the desired targets. Based on the deviation between these two, the network weights and bias are adjusted and updated.

Table 5.4: Parameters for training the fault section identification ANNs

Parameter	**Main Feeder**	**Lateral Section (%)**
Fault Location (%)	10, 20, 30, 40, 50, 60, 70, 80, 90, 95	5, 25, 50, 75, 95
Fault Resistance (Ω)	0, 20, 100	0, 20, 100
Fault Inception Angle(˚)	0, 30, 60, 90	0, 30, 60, 90

Also, the number of epochs was determined by experimentation. The number of epoch for the training was set between the range of 500 and 3500. An epoch of 500 means the weights are updated with the learning rule continuously until input dataset has been presented for 500 times. In addition, a MSE rate of e-03 was used. The final learning rate used for the SCG algorithm was 0.6 after several experiments with learning rates ranging from 0.1 to 1.0. When the learning rate is large, learning occurs quickly. If the learning rate is too large, it leads to instability and the errors increase.

In order to ensure that the NN is able to generalize, it is sometimes necessary to stop the training (learning) before the network converges to a local minimum. The learning or training phase of a NN can be interrupted on three conditions.
These are based on if:

- the maximum number of iteration is reached;
- a pre-set time is reached;
- the mean squared error has converged to a minimum; or
- the learning algorithm has reached a minimum gradient.

The training can also be terminated during training if it is observed that the error stays constant for a sufficiently long number of training epochs.

Network pruning as defined in Haykin, (1999), is used to determine the final number of neurons/layer in the hidden layer. This involves starting the neural network training with large hidden layer neurons with subsequent decrements of the neurons until the right number of neurons with good network performance is obtained. Network growing was also used in training some neural networks.

Before the training trial, the network datasets were randomized in order to obtain well spread out training and validation datasets. The number of neurons in the hidden layer was decided by experimentation which involves different network configurations. The process started with the lowest number of neurons in the hidden layer and was increased until a suitable network having satisfactory performance is established.

Preference was given to the network with the smallest size in terms of the lowest number of hidden layer weights. The final architectures and structures selected are presented in Chapter Six. The software subroutines are given in Appendices C.9 and C.10 respectively.

5.6.5 Performance Analysis

A number of performance analyses were carried out on the network to determine its viability. First is the performance training curve. The performance training curve serves as a tool for viewing the network during training in order to monitor the progress of the NN training.

The Receiver Operating Characteristic (ROC) plot can also be used to check the quality of the NN classifier. For each class, threshold values are applied across the interval [0, 1] to the outputs. Two values are calculated for each of the thresholds; the True Positive Ratio (the number of outputs greater or equal to the threshold, divided by the number of one targets), and the False Positive Ratio (the number of outputs less than the threshold, divided by the number of zero targets). The ROC plot with the curve tending to the left and top edges of the plot signifies that the classification is good.

Another means of testing the performance of the neural network is the confusion matrices for the various types of errors that occurred for the trained neural network. The diagonal cells in green indicate the number of cases that have been classified correctly by the neural network and the off diagonal cells which are in red indicate the number of cases that have been wrongly classified by the ANN. The last cell in blue in each of the matrices indicates the total percentage of cases that have been classified correctly in green while the red signifies incorrectly classified cases.

The correlation coefficient is used to measure the relationship between the neural network's targets and the outputs, and how well the targets track the variations in the outputs. A correlation of 0 signifies no correlation, while 1 means complete correlation.

The final validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. Fault cases with parameters different from that used in the training dataset were simulated at different location at different fault resistance and fault inception angles. The parameters used in generating this test dataset are given in Table 5.5.

In addition, the notation used in Chapter Six for the trained fault section identification networks is in the form [*I*-*H*-*O*]. *I* is the number of input neurons, *H* is the number of hidden layer neurons, and *O* is the number of output layer neurons. Hence, a network notation of [4-10-4] means 4 input layer neurons, 10 hidden layer neurons, and 4 output layer neurons respectively. Testing of the trained network was done using the codes in Appendix C.13.

Table 5.5: **Simulation parameters for testing the fault section identification ANNs**

Parameter	Main Feeder	Lateral Section (%)
Fault Location (%)	15, 25, 45, 50, 60, 85	15, 30, 60, 80, 90
Fault Resistance (Ω)	0.5, 2.5, 5	0.5, 2.5, 5
Fault Inception Angle(˚)	0, 30, 45, 60, 90	0, 30, 45, 60, 90

5.7 Fault Location Algorithm

In the design of the fault location algorithm, an approach similar to that highlighted for the fault section identification is used.

Neural networks corresponding to four classes of faults are developed (i.e. 1Ph.-G, 2Ph., 2Ph.-G., 3Ph.). The designed network takes in sets of four inputs (entropy per unit of the three phase and zero sequence current values) and one output node that gives the estimated distance of the fault in km. Table 5.6 is a summary of the parameters used in generating the ANN training dataset. Figure 5.11 is a block diagram of the processes involved in the proposed fault location algorithm.

Table 5.6: **Simulation parameters for training the fault location ANNs**

Parameter	Main Feeder	Lateral Section (%)
Fault Location (%)	10, 20, 30, 40, 50, 60, 70, 80, 90, 95	5, 25, 50, 75, 95
Fault Resistance (Ω)	0, 20, 100	0, 20, 100
Fault Inception Angle(˚)	0, 30, 60, 90	0, 30, 60, 90

Hence, a total of 1780 cases have been simulated for the training of the fault location ANNs.

5.7.1 Design Process for the Fault Location ANNs

The neural networks for the proposed fault location task are designed as an approximator problem.

This design process followed the same procedure with that used for the FSI task detailed above. Three methods were used in the implementation of the NN fault location algorithm.

These are:

- Split-sample or hold-out validation method
- Early-stopping method
- Regularization method

Detailed explanations of these concepts were provided in the preceding sub-section.

5.7.2 Feature Selection and Data Pre-processing

Neural network training is made more efficient if certain pre-processing steps are further performed on the network inputs and targets. Generally, normalization of the input and target vectors in the dataset is carried out in order to make the ANN training faster and also avoid outputs that will be stuck in the local minima. Thus, the datasets are normalized based on equation (5.13).

Since this is a multi-task problem, the problems are decomposed into individual sub-problems where each sub-problem is dedicated to one task. This is as shown in Figure 5.11. Using an ANN for only one sub-problem makes it more powerful, accurate, efficient, and increases the learnability of the ANN. Thus, the proposed fault location algorithm consists of multiple ANNs.

Four different networks were finally selected to approximate the distance to fault for different fault types. The particular fault location network is activated by the output of the fault type classification module based on the fault type.
The Neural Network Toolbox in MATLAB Version 7.12.0.635 (R2011a) was used in the design of the various ANNs of the fault location algorithms. The initial weights and initial biases were taken as random values.

The four neural network approximators are discussed in subsequent sub-sections. Emphasis was placed on the performance comparison of the neural networks in terms of the size of the neural network, the learning process, error histogram plot(s), regression coefficient, classification accuracy, and its ability to generalize.

The wavelet energy entropies per unit were collected and organized for training using Microsoft Excel. The task of fault location was accomplished by properly designing and developing the appropriate neural networks. Several architectures were investigated and a comparative report of the various architectures was carried out. The procedures followed are similar to that highlighted in Figure 5.10 for the design of ANNs for fault section identification.

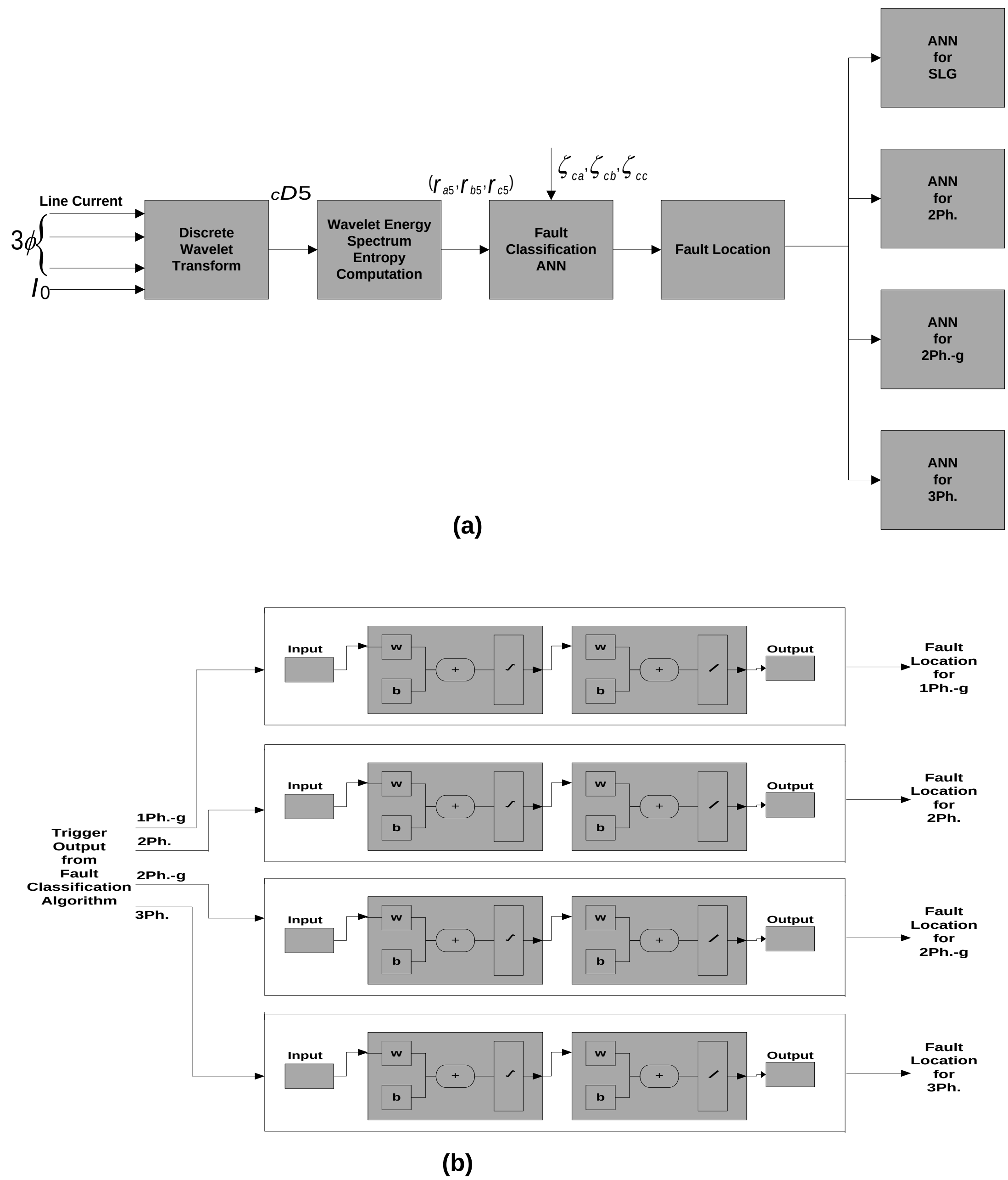


Figure 5.11: (a) Functional blocks of the proposed fault location algorithm; and **(b)** Implementation of the fault location algorithm

5.7.3 Network Architecture

The selection of the right architecture as earlier mentioned in section 5.5.3 is of utmost importance in neural network implementation. Choosing the number of hidden layers and the number of neurons is difficult, because there are no generally acceptable theories.

The solutions normally are found by means of empirical testing. One practical consideration is to try and minimize the size of the network while maintaining good performance. Smaller NNs are less prone to noise in the training data set and generalize better. Thus, the aim here was to find the smallest NNs possible, but with good results. Equations (5.10)-(5.12) were used as a guide in this endeavour.

Different ANN structures are tested for various training strategies. A summary of the various ANN structure considered for 1Ph.-G, 2Ph., 2Ph.-G, and 3Ph. faults are given in Chapter Six.

In MATLAB, the implementation of neural networks is performed by means of matrix manipulation of the inputs, outputs, weights and biases vectors. Vectors from a training set are presented to the networks sequentially. The weights and bias of the network would be updated if the output of the network is different from the target dataset.

Table 5.7 gives a breakdown of the training and test datasets used for 1Ph.-G, 2Ph., 2Ph.-G, and 3Ph. ANNs respectively.

Table 5.7: **Summary of the training and test data set for fault location**

Nr.	Fault Type	Size of Training Dataset	Size of Test Dataset
1	Single Phase-to-Ground	500	100
2	Two Phase	400	100
3	Two Phase-to-Ground	400	100
4	Three Phase	150	30

Thus, the training dataset for single phase-to-ground fault section identification consists of 500 four-element (4x500 matrix) input vectors and four-element target vectors. There are four elements in each target vector because four categories are associated with each input vector.

A coarse search for the hidden layer (H) = 5:5:55 is used to determine the number of hidden layer neurons for the 1Ph.-G, 2Ph., 2Ph.-G, and 3Ph. ANNs, depending on the maximum number of neurons possible based on equations (5.10)-(5.12). A detailed search is carried out after the coarse search in order to determine the optimal size of the ANN.

5.7.4 Neural Network Training

The datasets selected for the neural network training are normalized into the range of [-1, 1] before the training and testing. This is done by using equation (5.13).

Before the commencement of the training, the network was randomized in order to ensure that different weights and bias are used. The initialization of the weights and biases are repeated for 10 trials. The mean of these ten trials are tabulated and presented in Chapter Six. Also, performance plots of the best trial are presented for visualization.

The activation function for the hidden layer was the *tansig*, while *purelin* was used for the output layer. The selected activation functions are in accordance with the recommendations for neural network approximation/regression tasks as given in Demuth *et al.,* 2004. Other activation functions were also experimented upon, but were found to give poor performance.

The results presented and included in this thesis are for Resilient Propagation (RP), Scaled Conjugate Gradient (SCG), One-Step Secant (OSS), Bayesian Regularization (BR), and Levenberg-Marquardt (L-M) algorithms due to space considerations.

It is not possible to consider for training all the cases that an ANN-based fault location algorithm may actually encounter. Therefore, it is necessary to consider some representative fault situations and to train the network with data corresponding to these cases, such that the ANN gives correct output on testing even for those cases which it has never encountered before. During network training, the output of the network is computed and compared with the desired targets. Based on the deviation between these two, the network weights and bias are adjusted and updated.

Also, the number of epochs was determined by experimentation. The number of epoch for the training was set between the range of 500 and 3500. An epoch of 500 means the weights are updated with the learning rule continuously until input dataset has been presented for 500 times. The Mean Square Error (MSE) was e-03. The learning rate considered for SCG algorithm was with a value of 0.6. This was determined by experimentation.

In order to ensure that the NN is able to generalize, it is sometimes necessary to stop the training (learning) before the network converges to a global minimum. The learning or training phase of a NN can be interrupted on three conditions.

These are based on if:

- the maximum number of iteration is used;
- the pre-set time is reached;
- the mean squared error has converged to a minimum; or
- the learning algorithm has reached a minimum gradient

The training can also be terminated during training if it is observed that the error stays constant for a sufficiently long number of training epochs. Both network pruning and network growing presented in Haykin, (1999) were used to determine the final number of neurons/layer in the hidden layer. The number of neurons in the hidden layer was decided by experimentation which involves different network configurations. The process started with the lowest number of neurons in the hidden layer and was increased until a suitable network having satisfactory performance is established. The final architectures and structures selected are presented in Chapter Six. The software subroutines used for the neural network training for fault location are given in Appendices C.11 and C.12 respectively.

5.7.5 Performance Analysis

A number of performance analyses were carried out on the network to determine its viability. First is the performance training curve.

The error histogram serves as a means to assess the performance of the NN approximator. The correlation coefficient is used to measure the relationship between the neural network's targets and the outputs, and how well the targets track the variations in the outputs. The final test method was the use of an independent test vectors. The parameters used in generating the test dataset for the fault location task is the same as in the fault section identification task as given in Table 5.5. The notation used for the trained networks for fault location is in the form [*I-H-O*] as given in sub-section 5.6.5.

The percentage error of the NN accuracy (equation 5.14) is a function of the actual fault location and the total length of the line. Since this is a distribution network unlike a transmission network, it will be erroneous to use the total length of the distribution network or the total length of the main feeder in equation (5.14). In this regard, the length of the lateral where the fault occurred is assumed to be the length of the line in computing the percentage error. The length for the various line segments is calculated from the beginning of the feeder (node 800) to the end of the lateral. Table 5.8 gives the total length of the various segments in the test feeder. Testing of the trained neural network was done using the script file in Appendix C.13.

$$\%Error = \frac{Actual\ Location - Estimated\ Location}{Length\ of\ Line} \ X\,100 \tag{5.14}$$

5.8 Conclusion

The fault detection and diagnosis tasks are divided into five tasks. The tasks performed are for the waveform decomposition using DWT, fault event verification, fault type classification, fault section identification, and fault location. Level-1 and level-5 detail coefficients of discrete wavelet transform decomposition of the three phase and zero sequence current was used as the source of input. Wavelet entropy of the level-1 detail coefficient was then used as the input to the fault detection algorithm. Similarly, the wavelet entropy for level-5 detail coefficients was also calculated.

Table 5.8: **Feeder/lateral lengths**

Nr.	Line section	Feeder/Lateral length (km)
1	Main Feeder	57.415
2	Lateral 808-810	12.907
3	Lateral 816-822	51.112
4	Lateral 824-826	35.762
5	Lateral 854-856	48.594
6	Lateral 832-890	55.930
7	Lateral 858-864	54.699
8	Lateral 834-848	57.750
9	Lateral 836-838	58.982
10	Lateral 836-840	57.677

This formed the basis for the wavelet entropy per unit formulation proposed in this thesis. The wavelet entropy per unit served as the input to the rule-based fault classification algorithm, and also to the neural networks for the fault section identification and fault location algorithms respectively.

Chapter Six presents and discusses the results obtained from the development of the HFDD method. The results of the implementation of the rule-based fault detection and classification are presented and analysed. Similarly, the results and performance analysis for the NN fault section identification and fault location are also presented therein.

CHAPTER SIX

RESULTS AND DISCUSSION

6.1 Introduction

This chapter presents and discusses the results and analysis of the algorithms developed in Chapter Five. These results are presented for the five algorithms developed in thesis. Discrete Wavelet Transform was used to decompose the inputs into 6 levels. Levels-1 entropy values of the detail coefficients served as the inputs to the fault detection algorithm, while level-5 entropy values of the detail coefficients were used as inputs to the fault classification, fault section identification, and fault location algorithms respectively. The decisions for the fault detection and classification tasks were taken by rules developed through extensive simulations. Results of which are presented herein. Similarly, several ANN algorithms were trained for the fault section identification and fault location tasks respectively.

This chapter presents the results of the HFDD method for single phase to ground faults, two phase faults, two phase-to-ground faults, and three phase faults respectively. Also, results for both base case and modified case studies involving the integration of DGs and line extension are presented.

In the following sections and sub-sections, results for signal processing using discrete wavelet transform are presented for different mother wavelets. Also, results and analysis for the fault detection task using wavelet entropy, standard deviation, and mean absolute deviations are given. Furthermore, results and analyses for the implementation of the fault classification, fault section identification, and fault location algorithms are presented herein.

6.2 Discrete Wavelet Transform

The first task implemented in the development of the HFDD method was signal processing using discrete wavelet transform. This involved the use of a suitable mother wavelet to decompose the signal of interest which in this case was the three phase and zero sequence currents obtained from simulations as reported in Chapter Four.

Several experiments were carried out in order to determine which mother wavelet was best for the task. Results for Coiflet, Daubechies, and Symlet mother wavelets are shown in Figures 6.1-6.5, and in Tables 6.1-6.4 for phase A-G fault at line 806-808 (10% of the length of the main feeder). The fault resistance was 20Ω, while the fault inception angle was 90^o.

From these figures, it would be observed that db4 and Coiflet-3 provided the best match when the original waveform is compared to the plots of the approximation and detail components.

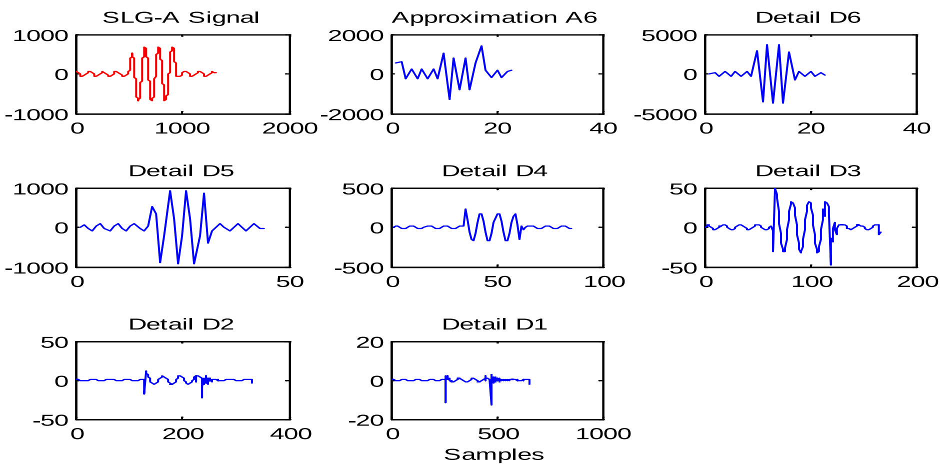


Figure 6.1: **Wavelet decomposition of SLG-A fault at line 806-808 (10% of the length of the main feeder) using Daubechies db2 mother wavelet**

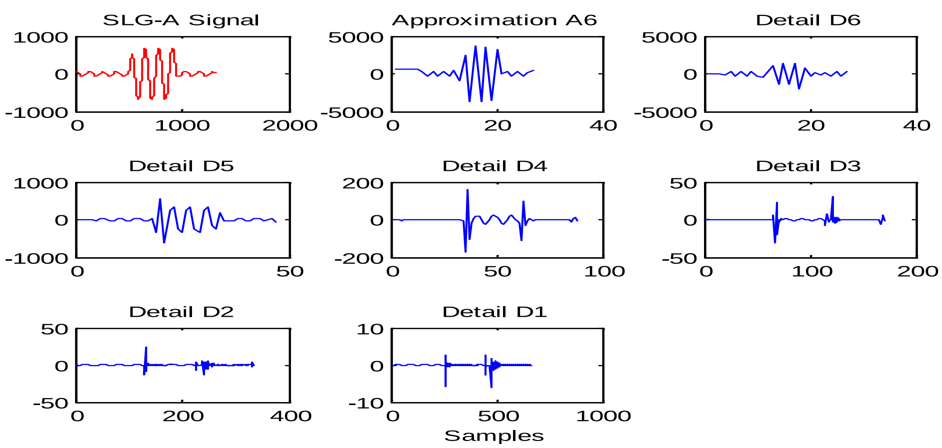


Figure 6.2: **Wavelet decomposition of SLG-A fault at line 806-808 (10% of the length of the main feeder) using Daubechies db4 mother wavelet**

Furthermore, a look at Tables 6.1 and 6.2 showed greater wavelet energies along the decomposition levels for db4 than in Coiflet-3. This implies that db4 demonstrates better characteristics that would be useful in fault detection. Also, the magnitude of wavelet energy in phase A (the faulted phase) is far greater than the other (unfaulted) phases which implies that fault classification can be better done using db4 rather than with

Coiflet-3. This also applies to wavelet entropy as shown in Tables 6.3 and 6.4. Extensive experiments carried out supports the db4 mother wavelet as the best to use, with level-1 detail coefficients providing excellent characteristics for fault detection, while level-5 showed great promise for the other diagnostic tasks.

Based on the above reasons, db4 was used as the mother wavelet in the DWT decomposition carried out in the thesis. The sub-routine (*dwt_exp.m*) used for the DWT experiments is given in Appendix C.1.

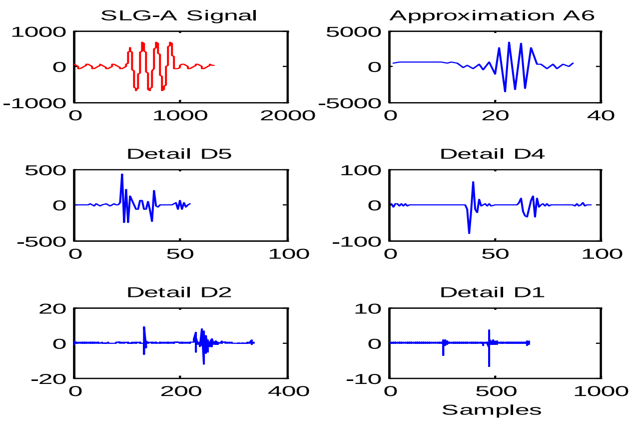


Figure 6.3: **Wavelet decomposition of SLG-A fault at line 806-808 (10% of the length of the main feeder) using Daubechies db8 mother wavelet**

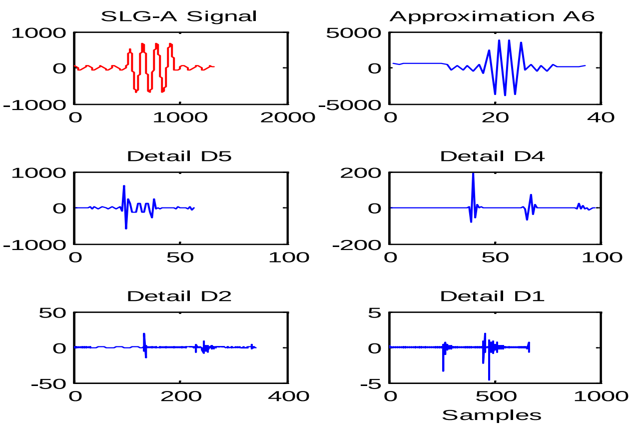


Figure 6.4: **Wavelet decomposition of SLG-A fault at line 806-808 (10% of the length of the main feeder) using Coiflet-3 mother wavelet**

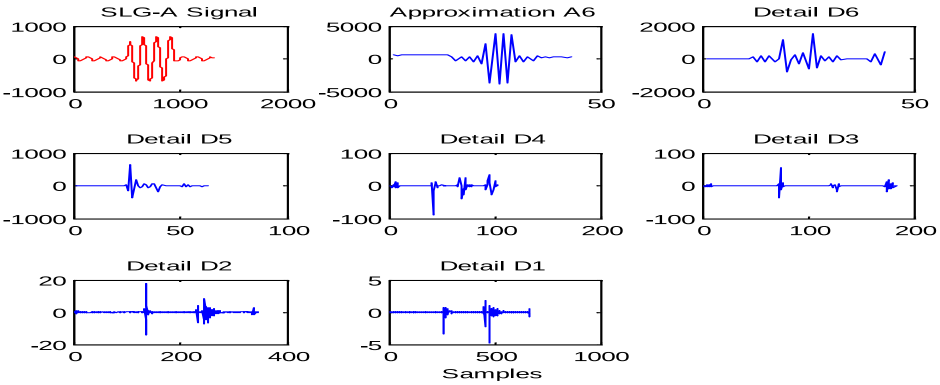


Figure 6.5: **Wavelet decomposition of SLG-A fault at line 806-808 (10% of the length of the main feeder) using Symlet-4 mother wavelet**

Table 6.1: **Wavelet energies for A-G fault at 10% of the length of the main feeder for Daubechies mother wavelets**

Mother wavelet	**db4**				**db5**				**db8**			
Scales	E_{ja}	E_{jb}	E_{jc}	E_{jl0}	E_{ja}	E_{jb}	E_{jc}	E_{jl0}	E_{ja}	E_{jb}	E_{jc}	E_{jl0}
Level-1	128.9	46.5	41.3	37.9	82.1	32.6	29.0	24.9	108.7	31.9	31.7	28.5
Level-2	1.31E+03	358.8	348.3	302.4	633.3	361.3	375.4	255.6	693.2	382.4	372.3	247.7
Level-3	3.19E+03	341.8	320.2	323.5	1.03E+04	527.2	438.9	1.36E+03	5.97E+03	183.1	86.0	671.7
Level-4	9.51E+04	1.86E+03	1.92E+03	1.21E+04	1.40E+04	2.22E+03	2.01E+03	1.76E+03	1.69E+04	1.23E+04	6.90E+03	3.0E+03
Level-5	1.77E+06	3.23E+04	3.77E+04	1.64E+05	8.54E+05	2.14E+04	1.73E+04	8.73E+04	4.98E+05	1.66E+04	9.93E+03	5.02E+04
Level-6	1.43E+07	2.33E+06	9.79E+04	1.13E+06	8.03E+07	7.79E+05	8.78E+05	6.90E+06	2.82E+07	2.24E+06	4.84E+04	2.32E+06

Table 6.2: **Wavelet energies for A-G fault at 10% of the length of the main feeder for Symlet and Coiflet mother wavelets**

Mother wavelet	**Coiflet-3**				**Symlet-4**			
Scales	E_{ja}	E_{jb}	E_{jc}	E_{jl0}	E_{ja}	E_{jb}	E_{jc}	E_{jl0}
Level-1	69.0289	26.4702	25.163	20.3545	78.5282	42.2799	42.0605	27.7958
Level-2	1.08E+03	386.3214	367.1606	267.8046	1.44E+03	370.0263	350.2661	298.4498
Level-3	7.57E+03	859.9102	459.0332	974.4522	1.11E+04	270.7689	223.4783	1.64E+03
Level-4	6.05E+04	8.53E+03	5.30E+03	7.65E+03	8.45E+04	3.61E+03	3.39E+03	1.31E+04
Level-5	1.11E+06	1.29E+04	1.50E+04	1.08E+05	2.41E+06	3.58E+04	3.46E+04	2.28E+05
Level-6	5.62E+06	2.14E+06	9.87E+05	6.38E+05	4.60E+06	2.13E+06	9.24E+05	5.48E+05

Table 6.3: **Wavelet entropies for A-G fault at 10% of the length of the main feeder for Daubechies mother wavelets**

Mother wavelet	db4				db5				db8			
Scales	r_a	r_b	r_c	r_{I05}	r_a	r_b	r_c	r_{I05}	r_a	r_b	r_c	r_{I05}
Level-1	2.5374	3.7492	3.8123	3.5551	2.7692	3.6658	3.6965	3.5996	2.3515	3.6323	3.6152	3.4763
Level-2	2.0987	3.1031	3.1321	2.7209	2.7655	3.1948	3.1571	3.2125	2.8689	3.2767	3.2529	3.1937
Level-3	2.6984	2.5938	3.0079	2.7935	1.9812	2.4746	2.8699	2.0387	2.3655	3.4736	4.1772	2.1614
Level-4	2.7467	4.412	4.4197	2.8112	3.6732	3.774	3.7868	2.927	2.9832	3.0169	3.3404	3.0702
Level-5	3.7374	5.1914	4.5281	3.5526	3.4201	4.6394	4.8192	3.2194	3.0997	4.0444	3.9934	2.8243
Level-6	3.1647	4.5726	3.9347	2.7917	3.0532	4.264	4.3844	2.8472	3.1753	4.6823	4.2715	2.8757

Table 6.4: **Wavelet entropies for A-G fault at 10% of the length of the main feeder for Symlet and Coiflet mother wavelets**

Mother wavelet	Coiflet-3				Symlet-4			
Scales	r_a	r_b	r_c	r_{I05}	r_a	r_b	r_c	r_{I05}
Level-1	2.7514	3.7034	3.7625	3.7041	2.9271	3.6712	3.7035	3.7576
Level-2	2.3085	3.2447	3.2159	2.9415	1.8809	3.1871	3.174	2.6353
Level-3	2.6715	3.5219	3.5581	2.1789	2.1711	4.0226	3.8198	1.9769
Level-4	1.9883	3.444	3.5979	2.3109	2.6058	4.2838	4.1178	2.4218
Level-5	2.7698	4.4614	4.0174	2.6061	3.3516	4.8822	4.5915	3.2439
Level-6	2.8734	4.4627	4.5083	2.5066	2.6486	4.4505	4.2948	2.5245

6.3 Fault Detection

6.3.1 Introduction

The results for fault detection task of the HFDD method are covered in this sub-section. Several results covering faults and ‘no fault’ conditions are presented. ‘No fault’ conditions like load switching and capacitor switching were used to test the capability of the developed algorithm. Wavelet energy and wavelet entropy results are presented for the IEEE test feeder base case. Similarly, wavelet entropy results for the modified cases involving the integration of Distributed Generation (DGs) and line extension are also given. Results obtained from other statistical methods like standard deviation and mean absolute deviation are presented in order to demonstrate the accuracy of the wavelet entropy method in fault detection.

6.3.2 Base Case

In order to show the detection capability of the fault detection algorithm, the procedure given in Figure 5.4 was followed. The results at various locations along the main feeder and at the laterals are presented. The focus of which is the result at the beginning, mid-point, and at the far end of the feeder. These results are illustrated for various fault types, fault resistances, and fault inception angles. The reasoning behind this is that the amplitude of the current signal varies and diminishes further down the feeder away from the voltage source/measurement point at the substation.

Therefore, if the performance of the algorithm is good at the far end, it will be valid at any distance along the entire length of the main feeder. Line 806-808 is located about 4,310 ft. from Node 800 which is the monitoring point at the substation. Line 860-836 is at the farthest point of the main feeder with a distance of 185,630 ft. from Node 800. Similarly, Line 846-848 is a lateral located 188,880 ft. from the upstream substation. This implies that the line current and the fault current will be lower than they are for locations closer to the substation. Wavelet decomposition of level-1 through level-6 for a C-g fault on Line 846-848 is shown in Figure. 6.6. These results are used further for fault detection.
From the results obtained through several simulation cases, the faulted phase is associated with the lowest value of wavelet energy entropy (r_{p1}). Thus, for a 1 Ph. A-G fault, r_{a1} has the lowest value compared to r_{b1} and r_{c1}. Similarly, for BC fault, the value of r_{a1} is greater than both r_{b1} and r_{c1}.

The same rule applies to 2 Ph.-g faults. This is because the faulty phase(s) have the highest energies as shown in Table 6.5. The wavelet energies (calculated using Equation (5.1)) for 10% of the main feeder for different combinations of fault resistance and fault inception angle are shown in Table 6.5. Table 6.6 gives the result for ‘no-fault’ conditions. Summing the energies, calculating the relative energy, and the entropy using Equations (5.2)-(5.4) reverses this. That is to say, the faulted phase(s) would be associated with the highest wavelet energy. But as a result of the relative energy calculation used in the computing the wavelet entropy, the faulted phase(s) would have the lowest wavelet entropy value.

Table 6.7 shows the values of r_{p1} obtained for ‘no fault’ test cases for 0^o load angle and for faults at 10% of the total length of the main lateral. Four classes of faults are given with R_f of 0Ω and θ_{fA} of 0^o respectively. The values of r_{p1} in bold indicates which of the phase is above the pre-set threshold. From Table 6.7, It can be seen that the values of r_{p1} obtained for the fault cases clearly exceed that of 'no fault cases'. Thus, fault can easily be detected.

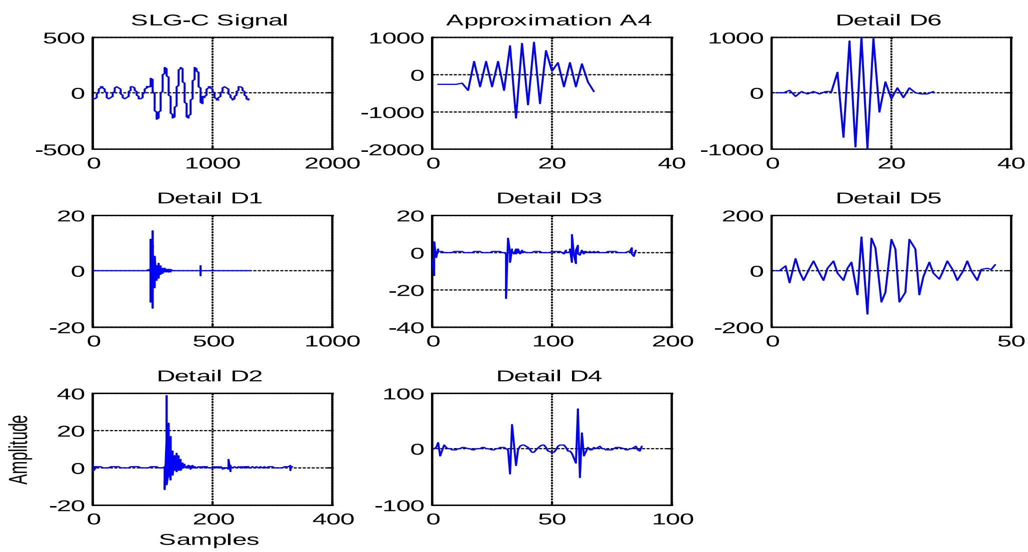


Figure 6.6: **Wavelet decomposition of SLG-C fault at lateral 846-848**

Table 6.5: **Wavelet energies for A-G fault at 10% of the length of the main feeder**

Phase	0Ω/0°	0Ω/45°	0Ω/90°	20Ω/0°	20Ω/45°	20Ω/90°
Phase A	8.13E+06	4.19E+06	5.88E+06	1.03E+06	1.12E+06	1.77E+06
Phase B	3.66E+04	4.13E+04	3.77E+04	3.32E+04	3.48E+04	3.23E+04
Phase C	3.32E+04	3.13E+04	3.85E+04	3.23E+04	3.20E+04	3.77E+04
Zero Seq.	8.38E+05	4.27E+05	6.02E+05	9.39E+04	9.97E+04	1.64E+05

Table 6.6: **Wavelet energies for 'No-Fault' conditions for the test feeder**

Phase	0° load angle	15° load angle	30° load angle	Cap. switching	Load switching
Phase A	3.55E+04	3.41E+04	3.47E+04	4.41E+04	4.11E+04
Phase B	3.37E+04	3.67E+04	3.87E+04	3.84E+04	3.44E+04
Phase C	3.34E+04	3.09E+04	2.81E+04	2.79E+04	2.75E+04
Zero Seq.	199.29	191.37	177.52	157.52	161.65

Similarly, simulation results for 70% and 95% of the total length of the main feeder, and lateral L.834 are shown in Tables 6.8, 6.9, and 6.10 respectively.

Table 6.7: **Base case: Phase entropies for fault detection at 10% of the length of the main feeder (R_f of 0Ω and θ_{fA} of 0°)**

Phase entropy	No fault at 0° Load angle	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
r_{a1}	1.0784	**2.7212**	**3.0878**	**2.9462**	**3.0572**
r_{b1}	1.2919	3.4114	**3.0511**	**2.9330**	**3.4025**
r_{c1}	1.4095	3.4107	3.9099	3.6307	**3.1949**
r_{I01}	1.6068	3.3150	3.8154	3.6251	3.5792

Table 6.8: **Base case: Fault detection at 70% of the length of the main feeder (R_f of 20Ω and θ_{fA} of 90°)**

Phase entropy	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
r_{a1}	3.8206	3.4699	4.0225	2.7248
r_{b1}	3.2173	2.6767	3.3226	3.0926
r_{c1}	3.9665	2.8812	3.6255	3.0992
r_{I01}	3.4574	1.9863	3.5613	3.5892

Table 6.9: **Base case: Fault detection at 95% of the length of the main feeder (R_f of 100Ω and θ_{fA} of 0°)**

Phase entropy	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
r_{a1}	3.3507	3.0387	3.1459	2.1061
r_{b1}	3.4015	1.86	3.4209	3.1723
r_{c1}	2.4288	3.0401	3.1525	3.0082
r_{I01}	3.4225	3.5149	3.391	1.7031

Table 6.10: **Base case: Fault detection at a lateral L.834 (R_f of 20Ω and θ_{fA} of 0°)**

Phase entropy	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
r_{a1}	**2.5654**	**3.0367**	**3.0067**	**2.9767**
r_{b1}	2.8155	**2.8694**	**2.311**	**2.5621**
r_{c1}	3.2109	3.5178	3.1504	**2.5057**
r_{I01}	3.2438	3.6753	**2.7349**	**2.383**

6.3.3 Modified Case Studies

The results of the integration of DGs at specific locations as shown in Figure 4.13 and the extension of Line 840 are presented in Table 6.11. Table 6.11 also gives the wavelet entropies for A-G, AB, AB-G, and ABC faults for DG1 case study. This table shows the values of r_{p1}, $p = a,b,c,I_0$ obtained for 'no fault' test cases and for A-G faults at 10% of the length of the main lateral for DG1, DG2, DG3, and line extension case studies.

Table 6.11: **Fault detection-DG and line extension case studies: fault detection for 'no fault' and fault cases on DG1 case study**

Phase entropy	No fault DG1	No fault DG2	No fault DG3	No fault Line ext.	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
r_{a1}	0.8352	0.8279	0.8358	1.0656	**2.5316**	**2.698**	**2.4178**	**2.6336**
r_{b1}	0.982	1.0278	0.9736	1.5289	3.2908	**3.7966**	**3.4185**	**3.4749**
r_{c1}	1.2614	1.1472	1.2815	1.3469	3.2583	3.679	3.378	**3.463**
r_{I01}	1.2123	1.2146	1.1984	0.9119	2.0576	3.3537	3.2797	3.2942

6.3.4 Fault Detection using Standard Deviation and Mean Absolute Deviation

For comparison with wavelet entropy, values of Standard Deviation (STD) and Mean Absolute Deviation (MAD) obtained from level-1 DWT decomposition of the three phase and zero sequence waveform are given in Table 6.12 for 'no-fault', load and capacitor switching, and for various fault types.

Table 6.12: **Fault detection using Standard Deviation and Mean Absolute Deviation**

Method	No fault 0^o Load angle	No fault 90^o Load angle	Load switching	Capacitor 844 switching	1Ph. A-G	2Ph. AB	2Ph. AB-G	3Ph.
STD(a)	3.68	3.16	3.67	4.34	25.86	33.23	27.21	26.33
STD(b)	3.00	3.11	2.99	3.45	6.54	32.69	51.15	34.50
STD(c)	2.75	3.03	2.75	2.81	5.87	3.27	11.57	42.89
STD(I0)	0.20	1.83	0.20	0.21	6.83	0.13	20.69	17.84
MAD (a)	2.12	1.94	2.12	2.53	13.55	13.30	13.54	13.48
MAD (b)	1.82	1.90	1.82	2.11	3.52	12.95	19.51	14.29
MAD (c)	1.80	1.83	2.57	1.88	3.17	2.25	5.44	18.33
MAD (I0)	0.12	0.24	0.13	0.13	3.62	0.091	8.91	4.28

6.3.5 Discussion of the Results

Results obtained for the base case and modified case studies showed excellent accuracy for fault detection using wavelet entropy. The algorithm was able to distinguish between fault and 'no fault' conditions like load and capacitor switching, and also load angle variations. These are known to cause transients which are often very similar to fault transients.

The performance of the algorithm was excellent for various fault types irrespective of the fault location, fault resistance, and fault inception angle. Standard deviation and mean absolute deviation also did provide good results.

6.4 Fault Classification Algorithm

6.4.1 Introduction

In order to show the classification capability of this algorithm, the results at various locations along the main feeder and at the laterals are also presented for various fault resistances and fault inception angles. The procedure given in Figure 5.5 was followed. The results presented below are for Line 806-808, Line 860-836, and Line 846-848. It should be noted that the faulted phase is associated with the lowest value of λ_{p5}, $p = a,b,c$.

6.4.2 Base Case

Tables 6.13-6.17 present the classification results obtained by the algorithm. The values of λ_{p5} obtained for ten different faults types with R_f of 0Ω and θ_{fA} of 0^o are shown in the tables. The values of λ_{p5} in bold are the faulted phase(s). From the results, it is seen that the various types of faults can easily be classified.

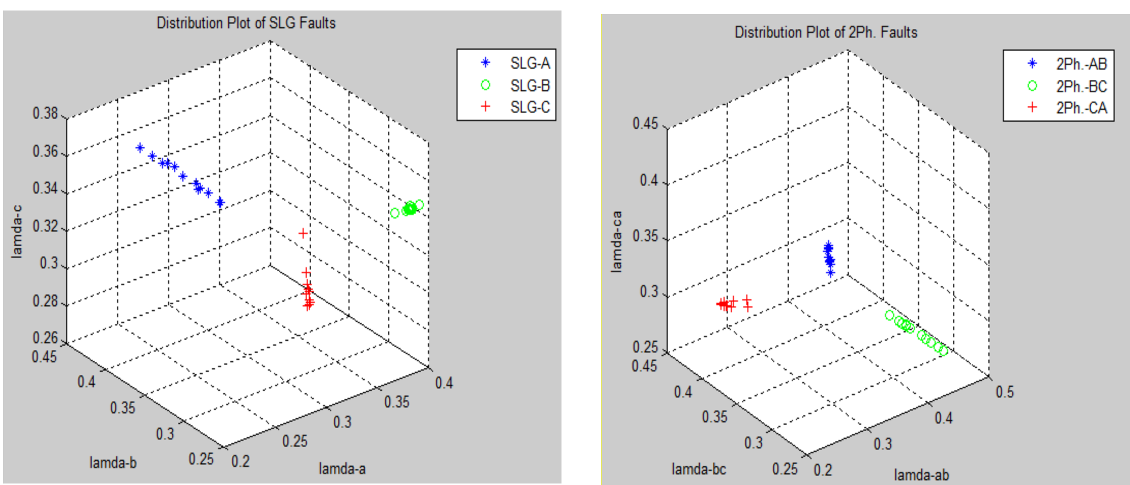
Thus, for a A-G fault, λ_{a5} has the lowest value compared to λ_{b5} and λ_{c5}. Similarly, for BC fault, the value of λ_{a5} is greater than both λ_{b5} and λ_{c5}. Figure 6.7 shows the distribution plot for the base case using wavelet entropy per unit from level-5 DWT decomposition for 1Ph.-G and 2Ph. Faults. Similarly, Figure 6.8 shows the distribution plots for 2Ph.-G fault using wavelet entropy per unit for level-5 DWT decomposition and that obtained for 1Ph.-G faults using wavelet entropy per unit for level-6 DWT decomposition respectively.

Table 6.13: Base Case: Fault Indices at Line 806-808 (R_f of 0Ω and θ_{fA} of 0^o)

Phase entropy	A-G	B-G	C-G	AB	BC	CA	AB-G	BC-G	CA-G	3Ph.
λ_{a5}	0.2308	0.3935	0.3684	0.3051	0.4529	0.2925	0.2701	0.4261	0.2462	0.2877
λ_{b5}	0.3960	0.2633	0.3631	0.3091	0.2732	0.4152	0.3191	0.2743	0.4156	0.3406
λ_{c5}	0.3732	0.3442	0.2684	0.3858	0.2739	0.2923	0.4109	0.2996	0.3382	0.3717
r_{I05}	2.8017	3.4457	3.6767	4.7597	3.5885	3.7559	2.9088	3.2036	3.3103	1.2639

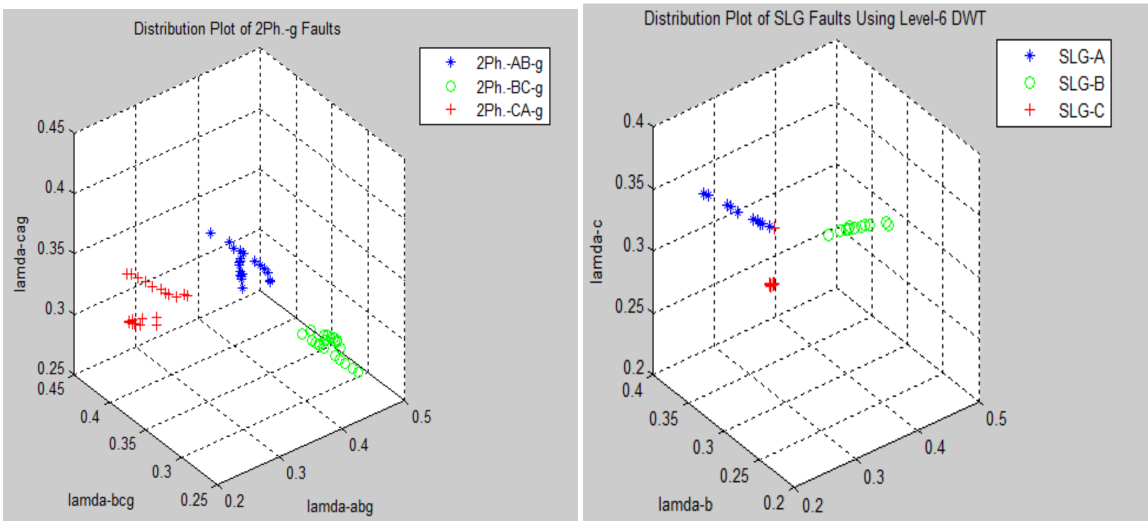
Table 6.14: Base Case: Fault Indices at Line 860-836 (Rf of 0Ω and θ_{fA} of 0o)

Phase entropy	A-G	B-G	C-G	AB	BC	CA	AB-G	BC-G	CA-G	3Ph.
λ_{a5}	0.2899	0.3868	0.3577	0.3258	0.3939	0.2982	0.3384	0.3956	0.2979	0.3365
λ_{b5}	0.3717	0.2744	0.3500	0.3243	0.2991	0.4194	0.2956	0.2894	0.3757	0.3161
λ_{c5}	0.3384	0.3387	0.2923	0.3499	0.3069	0.2823	0.3659	0.3149	0.3263	0.3473
r_{I05}	2.8632	3.2011	3.7994	4.7133	4.3531	4.3444	3.8369	2.9392	3.509	1.2407



(a) **(b)**

Figure 6.7: Distribution plot for the base case using entropy per unit **(a)** 1 Ph.-G faults; and **(b)** 2Ph. faults



(a) **(b)**

Figure 6.8: **Distribution plot for the base case using entropy per unit weight (a) 2Ph.-G faults; and (b) 1Ph.-G faults**

Table 6.15: Base Case: Fault Indices at Lateral L. 834 (R_f of 0Ω and θ_{fA} of 0°)

Phase entropy	A-G	B-G	C-G	AB	BC	CA	AB-G	BC-G	CA-G	3Ph.
λ_{a5}	0.2901	0.3868	0.3577	0.3258	0.3939	0.2983	0.3382	0.3955	0.2951	0.3362
λ_{b5}	0.3715	0.2744	0.3499	0.3244	0.2992	0.4193	0.2957	0.2895	0.3781	0.3162
λ_{c5}	0.3384	0.3388	0.2923	0.3498	0.3069	0.2824	0.3661	0.3149	0.3268	0.3475
r_{I05}	2.8633	3.1985	3.7981	4.7133	4.3602	4.346	3.5905	2.9385	3.4007	1.2446

Table 6.16 gives the indices obtained for Lateral 846-848 at R_f of 20Ω and θ_{fA} of 0°. Similarly, in Table 6.17, the λ_{p5} values for A-G, B-G, and C-G are given for R_f of 0Ω and 20Ω and θ_{fA} of 0°, 45°, and 90°.

Table 6.16: Base Case: Fault Indices at Line 846-848 (R_f of 20Ω and θ_{fA} of 0°)

Phase entropy	A-G	B-G	C-G	AB	BC	CA	AB-G	BC-G	CA-G	3Ph.
λ_{a5}	0.3069	0.3817	0.3563	0.3260	0.3885	0.3010	0.3470	0.3869	0.3190	0.3467
λ_{b5}	0.3649	0.2799	0.3509	0.3265	0.3036	0.4175	0.2954	0.2977	0.3613	0.3202
λ_{c5}	0.3282	0.3384	0.2927	0.3474	0.308	0.2815	0.3576	0.3154	0.3196	0.3331
r_{I05}	3.1512	3.2751	3.7770	4.6549	4.3623	4.2438	3.743	2.8895	3.5036	1.8387

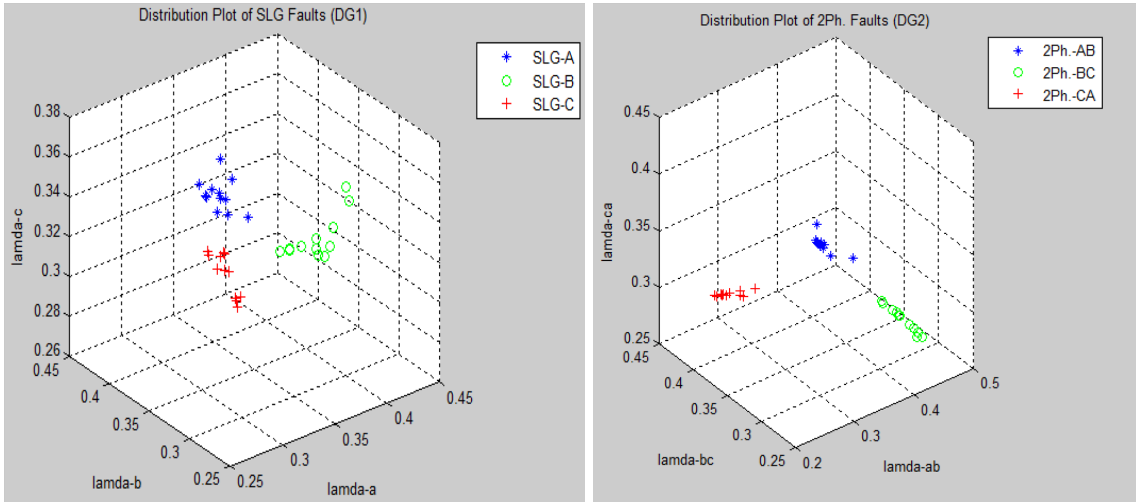
6.4.3 Modified Case Studies

Figure 6.9 is a visualization of the distribution or spread of 1 Ph.-G faults for DG1 and 2Ph.-G for DG2 case studies respectively. It shows the distribution of 1 Ph.-G and 2Ph. faults for 10% to 95% of the length of the main feeder, at laterals 820-822, and at 846-848 respectively. The distribution of these features indicates that WEE is efficient and can reflect high order statistical information. Table 6.18 shows typical values obtained for the various modified case studies. The λ_{p5} indices in bold are the phase(s) involved in a fault.

Table 6.19 presents some misclassified faults. These were recorded for A-g faults at high fault resistance and for AB-G and CA-G faults located in close proximity to the generators. Although, these faults were misclassified, they were nevertheless correctly detected by the fault detection module.

Table 6.17: **Base Case: Fault Indices for various operating conditions**

Fault type	**Fault location/ % of the feeder length**	**Fault inception angle/deg.**	**Fault resistance/Ω**	λ_{a5}	λ_{b5}	λ_{c5}	r_{I05}
A-g	10	0	0	**0.2308**	0.3960	0.3732	2.8017
		0	20	**0.2821**	0.3840	0.3339	3.5949
		45	0	**0.2375**	0.3891	0.3735	2.8145
		45	20	**0.2891**	0.3806	0.3303	3.7199
		90	0	**0.2534**	0.3827	0.3639	3.1550
		90	20	**0.2777**	0.3858	0.3365	3.5526
B-g	10	0	0	0.3925	**0.2633**	0.3442	3.4457
		0	20	0.4011	**0.2568**	0.3421	3.2068
		45	0	0.3903	**0.2833**	0.3264	3.6803
		45	20	0.4083	**0.2535**	0.3383	3.1162
		90	0	0.3823	**0.2742**	0.3435	3.6003
		90	20	0.4081	**0.2391**	0.3527	2.9221
C-g	10	0	0	0.3683	0.3631	**0.2684**	3.6767
		0	20	0.3686	0.3658	**0.2628**	3.5727
		45	0	0.3644	0.3632	**0.2724**	3.6811
		45	20	0.3686	0.3713	**0.2601**	3.5123
		90	0	0.3667	0.3639	**0.2694**	3.6458
		90	20	0.3731	0.3751	**0.2518**	3.4032



(a) **(b)**

Figure 6.9: **Distribution Plot of wavelet energy entropy per unit weight: (a) 1 Ph.-G faults for DG1; and (b) 2 Ph. faults for DG2**

Table 6.18: **DG case studies: fault indices for various operating conditions at location 10% of the length of the main feeder**

Fault type	**Phase entropy**	**DG1**		**DG2**		**DG3**		**Line ext.**	
		$0\Omega/0^o$	**$20\Omega/45^o$**	**$0\Omega/0^o$**	**$20\Omega/45^o$**	**$0\Omega/0^o$**	**$20\Omega/45^o$**	**$0\Omega/0^o$**	**$20\Omega/45^o$**
C-g	λ_{a5}	0.3935	0.3722	0.3800	0.3683	0.3716	0.3670	0.3687	0.3714
	λ_{b5}	0.3389	0.3807	0.3378	0.3854	0.3467	0.3866	0.3618	0.3717
	λ_{c5}	**0.2677**	**0.2471**	**0.2822**	**0.2463**	**0.2817**	**0.2464**	**0.2695**	**0.2569**
	r_{I05}	3.4945	3.3177	3.7045	3.3298	3.6886	3.3331	3.7115	3.4528
2Ph. A- B	λ_{a5}	**0.2969**	**0.3048**	**0.2989**	**0.2995**	**0.3057**	**0.2985**	**0.3071**	**0.3038**
	λ_{b5}	**0.3028**	**0.3091**	**0.3063**	**0.3045**	**0.3117**	**0.3038**	**0.3109**	**0.3099**
	λ_{c5}	0.4003	0.3861	0.3948	0.3959	0.3826	0.3977	0.3819	0.3863
	r_{I05}	4.6020	4.6162	4.5545	4.5911	4.5930	4.6017	4.8441	4.7565
2Ph. C-A-G	λ_{a5}	**0.2726**	**0.3186**	**0.2767**	**0.3161**	**0.2710**	**0.3156**	**0.2456**	**0.3105**
	λ_{b5}	0.4248	0.4043	0.3791	0.4096	0.3927	0.4108	0.4131	0.4008
	λ_{c5}	**0.3026**	**0.2771**	**0.3442**	**0.2743**	**0.3363**	**0.2736**	**0.3413**	**0.2889**
	r_{I05}	**3.6647**	**3.3265**	**3.5935**	**3.3226**	**3.6062**	**3.3167**	**3.3990**	**3.3649**

Table 6.19: Misclassified faults

Location	Fault resistance/ Ω	Fault inception angle /°	λ_{a5}	λ_{b5}	λ_{c5}	r_{I05}	Fault type	Misclassified as
Base Case/lateral 818	100	0	0.3209	0.3596	0.3195	3.7224	A-g	C-g
Base Case/lateral 864	100	0	0.3340	0.3542	0.3118	3.7553	A-g	C-g
Base Case/Line 830-854	100	45	0.3578	0.3041	0.3381	3.6540	AB-g	BC-g
DG1/Line 836-840	0	0	0.3174	0.3783	0.3042	3.7380	A-g	C-g
DG1/Line 836-840	0	0	0.3398	0.3143	0.3459	3.1996	BC-g	AB-g
DG2/Line 844-846	0	0	0.3174	0.3783	0.3042	3.7380	A-g	C-g
DG2/Line 844-846	0	0	0.3531	0.3042	0.3427	3.1224	AB-g	BC-g
DG3/Line 836-840	0	0	0.3231	0.3773	0.2996	3.8725	A-g	C-g
DG3/Line 842-844	0	0	0.3222	0.3775	0.3003	3.8432	A-g	C-g
DG3/Line 842-844	0	0	0.3412	0.3068	0.3519	3.2347	BC-g	AB-g
DG3/Line 842-844	0	0	0.3392	0.3271	0.3337	3.3283	CA-g	BC-g

6.4.4 Fault Classification using Standard Deviation and Mean Absolute Deviation

The values of STD and MAD obtained from level-5 DWT decomposition of the three phase and zero sequence waveforms are given in Table 6.20 for the various case studies considered. The table illustrates a B-C fault along Line 846-848. Also, the values obtained for Lines 834-842 and 828-830 are presented. The faulted phase(s) are associated with the highest values of STD and MAD.

Table 6.20: Fault Classification using Standard Deviation and Mean Absolute Deviation

Case study	Location	Fault type	Fault parameters	STD(a)	STD(b)	STD(c)	STD(I0)	MAD(a)	MAD(b)	MAD(c)	MAD(I0)
Base Case	Line 846-848	B-C	$R_f = 2.5\Omega$, $\theta_f = 30^o$	1.79	**6.11**	**6.56**	0.45	1.54	**3.95**	**4.10**	0.25
DG1	Line 846-848	B-C	$R_f = 2.5\Omega$, $\theta_f = 30^o$	3.52	**4.85**	**5.16**	0.13	2.17	**4.88**	**5.48**	1.93
DG2	Line 846-848	B-C	$R_f = 2.5\Omega$, $\theta_f = 30^o$	3.52	**4.63**	**4.72**	0.11	2.51	**3.105**	**2.99**	0.07
DG3	Line 846-848	B-C	$R_f = 2.5\Omega$, $\theta_f = 30^o$	3.84	**4.66**	**4.56**	0.09	2.73	**3.09**	**2.87**	0.06
DG1	Line 834-842	A-g	$R_f = 0\Omega$, $\theta_f = 0^o$	**5.12**	3.85	3.55	1.99	**3.43**	2.57	2.30	1.16
DG1	Line 834-842	B-g	$R_f = 0\Omega$, $\theta_f = 0^o$	9.25	**10.25**	4.05	4.39	5.31	**4.53**	2.30	1.97
Base Case	Line 828-830	A-g	$R_f = 100\Omega$, $\theta_f = 0^o$	**5.43**	2.98	2.88	2.08	**3.46**	1.93	2.00	0.86
Base Case	Line 834-842	A-g	$R_f = 100\Omega$, $\theta_f = 0^o$	**5.35**	3.39	3.35	2.67	**3.34**	2.17	2.21	1.02

6.4.5 Discussion of the Results

Results obtained for the base case and modified case studies showed excellent accuracy for fault classification using wavelet entropy. The fault classification task was the identification of the type of fault and also the faulted phase(s). The fault classification algorithm was tested with several fault types at different locations, fault resistance, fault inception angles, external disturbances, etc. Apart from cases with mutual coupling amongst the phases, the algorithm performed well. However, standard deviation and mean absolute deviation did not give good results for fault classification in general. This is as shown in Table 6.20.

6.5 Fault Section Identification

Multi-Layer Perceptron Neural Network and various back-propagation learning algorithms with different combinations of hidden layers (and the number of neurons per hidden layer) have been experimented on. Of these, some of the networks trained and the final network that achieved satisfactory performance are shown. An exhaustive survey on various neural networks has been performed by varying the number of hidden layers, number of neurons per hidden layer, the activation functions, and the learning algorithm. Of these ANNs, the most appropriate ANN is chosen based on the Receiver Operating Characteristics (ROC), confusion matrix, MSE performance, regression coefficient, and testing using untrained dataset. After considering the aforementioned criteria, the final decider is the number of weights of the network. The network with the lowest number of weights and good performance is chosen.

The ROC curves are plots between the true positive rates (rate of positive classification) and the false positive/incorrect classification rates of the neural network classifier. An ideal ROC curve shows points only in the upper-left corner because that is an indication of 100 percent true positivity and 0 percent false positivity in the classification. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test, and vice versa. Given a classifier and an instance, there are four possible outcomes. If it is positive and it is classified as positive, it is regarded as a true positive. If incorrectly classified as negative, it is regarded as a false negative. For a negative instance, if classified correctly as negative, it is regarded as a true negative, and false positive for vice versa.

The confusion matrix examines the actual number of cases that have been classified positive by the neural network. Ideally, this percentage is 100. Hence, if the confusion matrix indicates low positive classification rates, it indicates that the neural network might not perform well. A confusion matrix contains information about known target and predicted outputs. That is, the *(i,j)* element in the confusion matrix refers to the number of samples with known class label *i* and predicted class *j*. The diagonal elements represent the correctly classified observations.

Regression coefficient (*R*) is used to describe the relationship between the target and output vectors. The analysis of this curve gives an idea on the network performance of the ANN. Ideally the slope should be 1. Also, the error goal (MSE) indicates the closeness of the desired and the actual output. A low MSE during training suggests that the network construction is of a good accuracy and quality. MSE goal of 0.001, 0.01, and 0.1 were experimented on.

The last method of testing the neural network is to present it with a whole new set of data (untrained) with known inputs and targets and calculate the percentage error in the neural networks output.

Since the training trials per NN training is ten, the best plot for some of the networks are included for illustration in this thesis.

6.5.1 Network Simulation for Single Phase-to-Ground Faults

6.5.1.1 Network 1 [4-55-4]

The first network had a single hidden layer with 55 neurons. The number of input and output neurons was 4 respectively. The target MSE was 0.001, while the maximum number of epochs was 1000. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* respectively.

The structure of the Multi-Layer Perceptron (MLP) neural network is as given in Figure 6.10.

The performance plot in Figure 6.11(a) shows that the training converged slowly and the network did not show any sign of improvement at the end of 1000 epochs. The number of epochs was increased to 2500. Despite this, the network still did not converge. Also, the network was not able to generalize to new data.

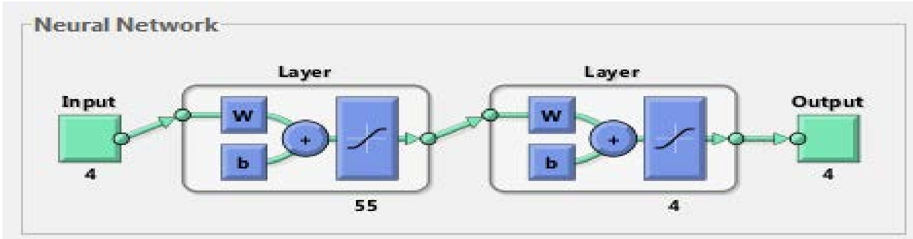
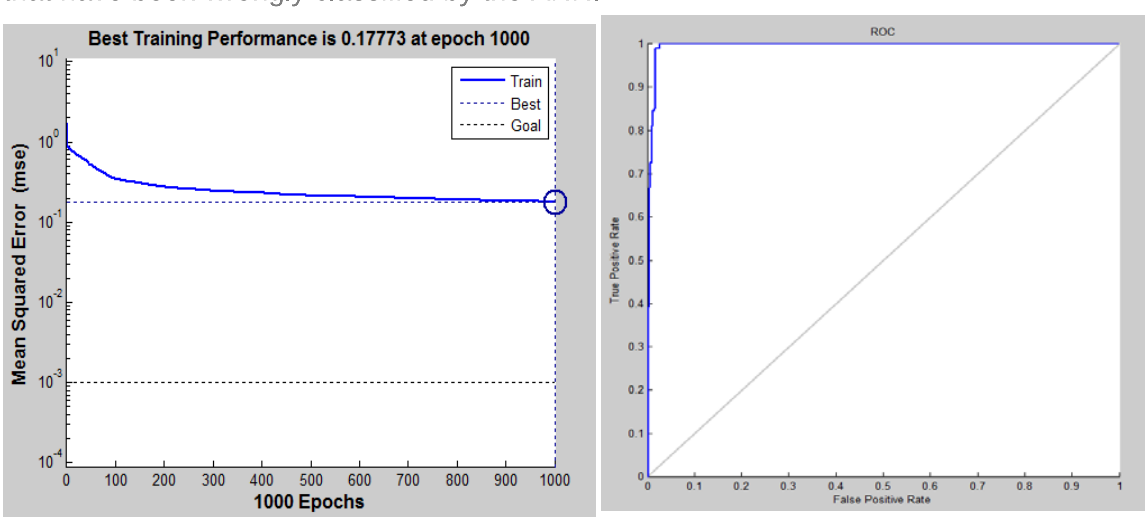


Figure 6.10: **Neural network architecture for 4-55-4 network**

A number of performance analyses were carried out on the network. The first of these was the ROC plot. This plot was used to check the quality of the NN classifier. For each class, threshold values are applied across the interval [0, 1] to the outputs. For each threshold, two values are calculated, the True Positive Ratio (the number of outputs greater or equal to the threshold, divided by the number of true (1s) targets), and the False Positive Ratio (the number of outputs less than the threshold, divided by the number of untrue (0s) targets). The ROC plot in Figure 6.11(b) shows the curve tending to the left and top edges of the plot but not to the edge. This implies that the best accuracy is not yet reached.

The second means of testing the performance of the neural network was to plot the confusion matrices for the various types of errors that occurred for the trained neural network. Figure 6.12(a) shows the confusion matrix for the NN training. The diagonal cells in green indicate the number of cases that have been classified correctly by the neural network and the off diagonal cells which are in red indicate the number of cases that have been wrongly classified by the ANN.



(a) (b)

Figure 6.11: **(a)** **Training performance curve for 4-55-4; and** **(b)** **ROC plot showing the training state for 4-55-4**

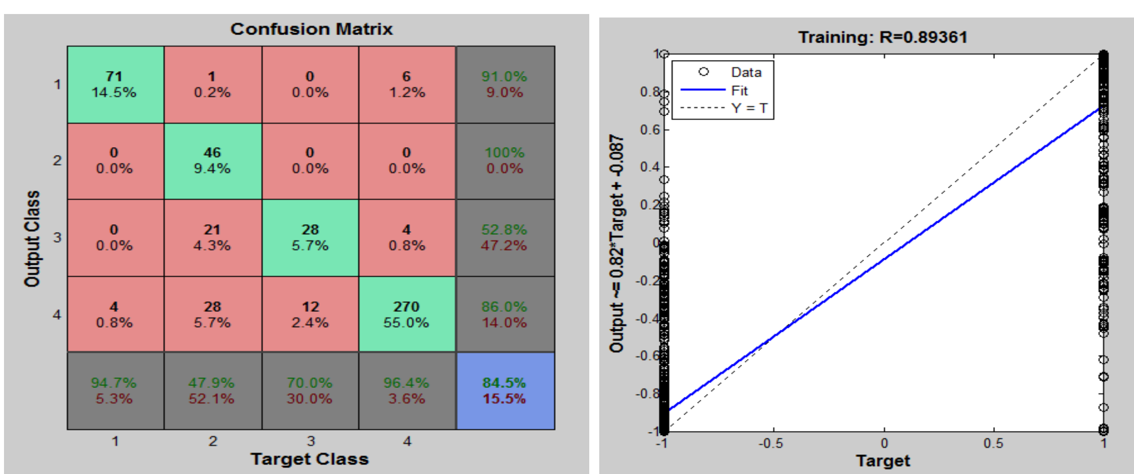
The last cell in blue in each of the matrices indicates the total percentage of cases that have been classified correctly in green while the red signifies incorrectly classified cases. It can be seen that the neural network has 84.5% accuracy in fault section identification.

Plotting the best linear regression that relates the targets to the outputs as shown in Figure 6.12(b). The correlation coefficient is a measure of the relationship between the neural network's targets and the outputs, and how well the targets can track the variations in the outputs. A correlation of 0 signifies no correlation, while 1 means complete correlation. The correlation coefficient in this case was 0.89361.

The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different location at different fault resistance and fault inception angles. These parameters are different from that used in the training dataset. The result obtained from the test dataset showed an accuracy of 89 correctly classified samples out of 100 test data.

Reducing the target goal to 0.01 showed a faster convergence of the learning curve. The learning curve descended at a fast rate for the first 400 epochs and then decreased at a slow rate. The goal of getting an MSE of 0.01 was not achieved; however the MSE of 0.168 was achieved at the end of 1000 epochs.

The correlation coefficient (*R*) in this case was 0.89994. The overall confusion matrix was 84.3%. Testing the network using untrained data set resulted in 18 errors out of 100 test data (82%).



(a) (b)

Figure 6.12: (a) Confusion matrix for 4-55-4; and **(b)** Regression plot for 4-55-4

The performance of this network measured by error rate was reasonably good. However, in term of training time it was quite slow. To get an MSE of 0.168 in 1000 epochs took 29 seconds.

Further reducing the target MSE to 0.1 gave a performance goal of 0.187 at the end of 1000 epochs. The correlation coefficient in this case was 0.88823, while the overall confusion matrix was 82.3%. Testing the network using untrained data set resulted to an accuracy of 79%.

6.5.1.2 Network 2 (4-55-4)

The training algorithm for the above network was changed to resilient back propagation (RProp). The target MSE was 0.01. The activation functions in the hidden and output layers were *tansig*.

The best training performance obtained using resilient back propagation at the end of 1000 epoch was 0.23674. The overall confusion matrix was 84.1%. The correlation coefficient in this case was 0.85535. Testing the network using untrained data set resulted to an accuracy of 81%.

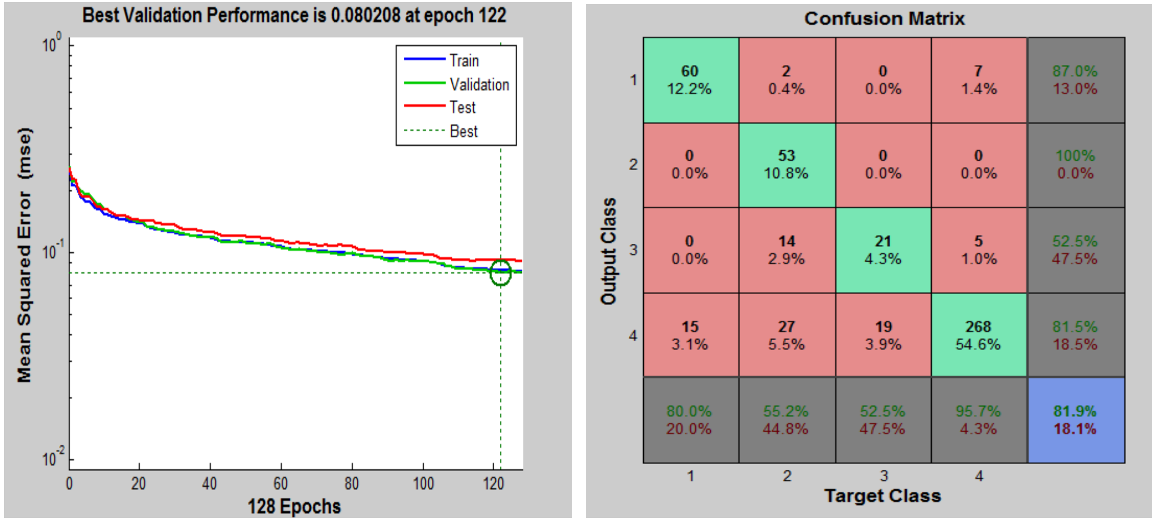
6.5.1.3 Network 3 (4-55-4)

The Levenberg-Marquardt (L-M) algorithm was also used in training this network. The activation functions in the hidden and output layers were *tansig* respectively.

The goal achieved was 0.099997, while an overall confusion matrix was 91%. The correlation coefficient in this case was 0.9538. Testing the network using untrained data set resulted to an accuracy of 86%.

6.5.1.4 Network 4 (4-55-4)

The early stopping method with SCG algorithm was used to simulate the 4-55-4 MLP network architecture. Using a cross validation set during the early stopping training ensured the network did not over-train.The performance plots is as shown in Figure 6.13(a). An overall confusion matrix of 81.9% (Figure 6.13(b)) with a correlation coefficient of 0.87824 was obtained. Testing the network using untrained data set resulted to an accuracy of 76%.



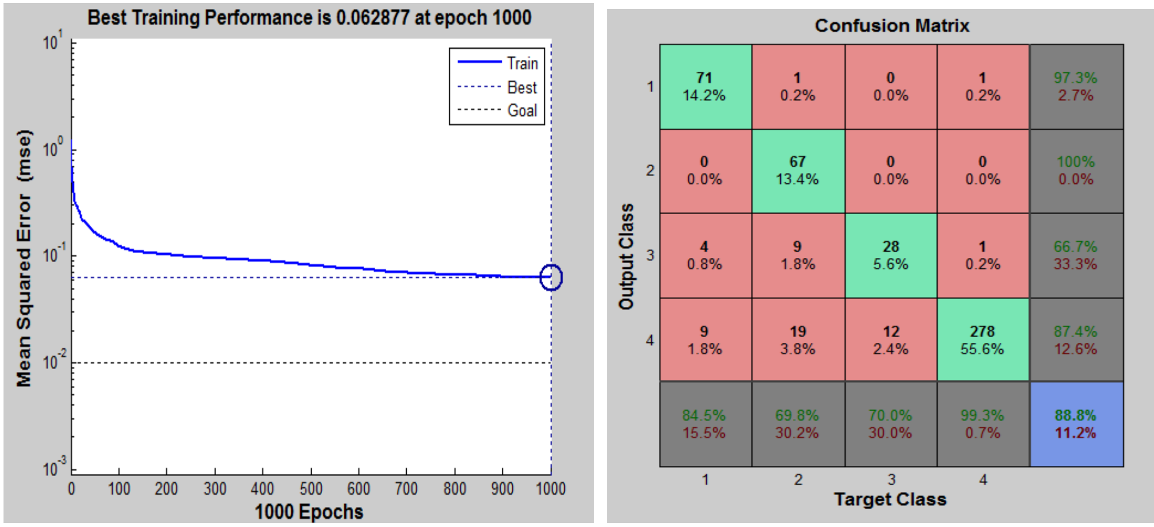
(a) (b)

Figure 6.13: (a) Performance plot for 4-55-4; and **(b)** Confusion matrix for 4-55-4

6.5.1.5 Network 5 (4-55-4)

The regularization method using the Bayesian regularization (BR) algorithm was also used to simulate the 4-55-4 MLP architecture. The confusion matrix obtained was 88.8%, while the correlation coefficient was 0.93729. Testing the network using untrained data set resulted to an accuracy of 87%.

From the foregoing, L-M algorithm was found to be the fastest learning algorithm which was able to converge the problem. The next stage now is to investigate the optimal size of the network. Thus, the network size was further reduced by reducing the network size in the hidden layer. The performance of the networks with hidden layer neurons with range H=55:5:5 were investigated. The learning method used was L-M algorithm. The networks were trained to reach performance goals of 0.001, 0.01, and 0.1. Some results of these NN training are given below in Figure 6.14.



(a) (b)

Figure 6.14: (a) Performance plot for 4-55-4; and **(b)** Confusion matrix for 4-55-4

6.5.1.6 Network 6 (4-10-4)

1. This network had a single hidden layer with 10 neurons. The target MSE was 0.001. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig*.

   The goal achieved was 0.15177, while the overall confusion matrix of 86.2% was obtained with a correlation coefficient of 0.9109. Testing the network using untrained data set resulted to an accuracy of 79%. Figures 6.15 and 6.16 show the performance plots obtained.

2. Early stopping method was used in order to validate the above neural network training by ensuring that the trained networks are not over trained and have the ability to generalize. This network had a single hidden layer with 10 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm.

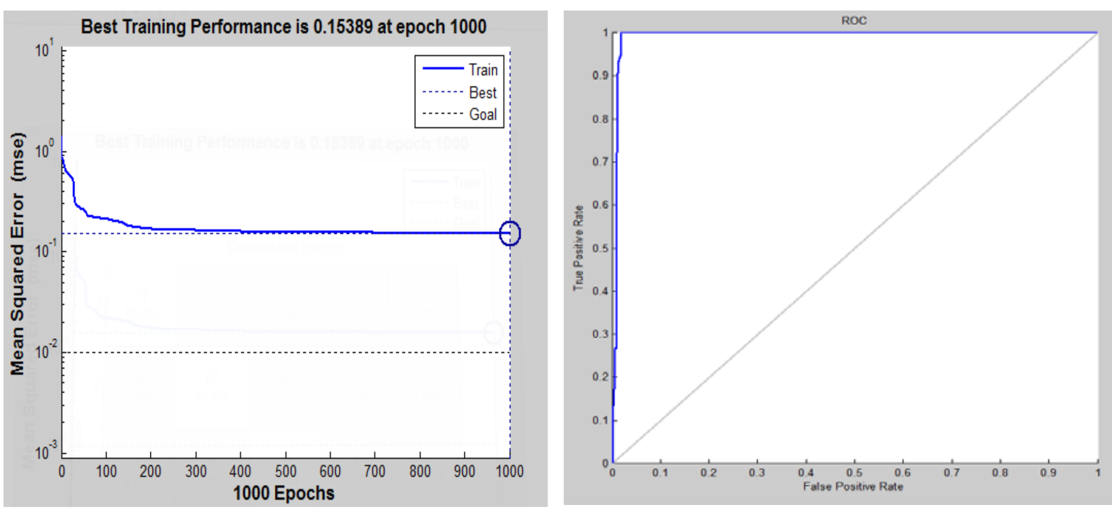
   The activation functions in the hidden and output layers were *tansig* respectively. The training stopped after 47 epochs. A MSE of 0.0468 was obtained in 8 seconds. The correlation coefficient in this case was 0.86445 with an overall confusion matrix of 85.9%. Testing the network using untrained data set resulted to an accuracy of 76%.

3. The regularization method was also used in order to validate the neural network training by ensuring that the trained networks are not over trained. The network had a single hidden layer with 10 neurons. The target MSE was 0.001. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and

output layers were *tansig*. The training stopped after 1000 epochs. A SSE of 71.22 was obtained. The correlation coefficient in this case was 0.91456 with an overall confusion matrix of 87.4%. Testing the network using untrained data set resulted to an accuracy of 81%. Similarly, Resilient Back-Propagation (RP) algorithm was investigated, but did not give any impressive results. Also, the One Step Secant (OSS) algorithm did not give good results.

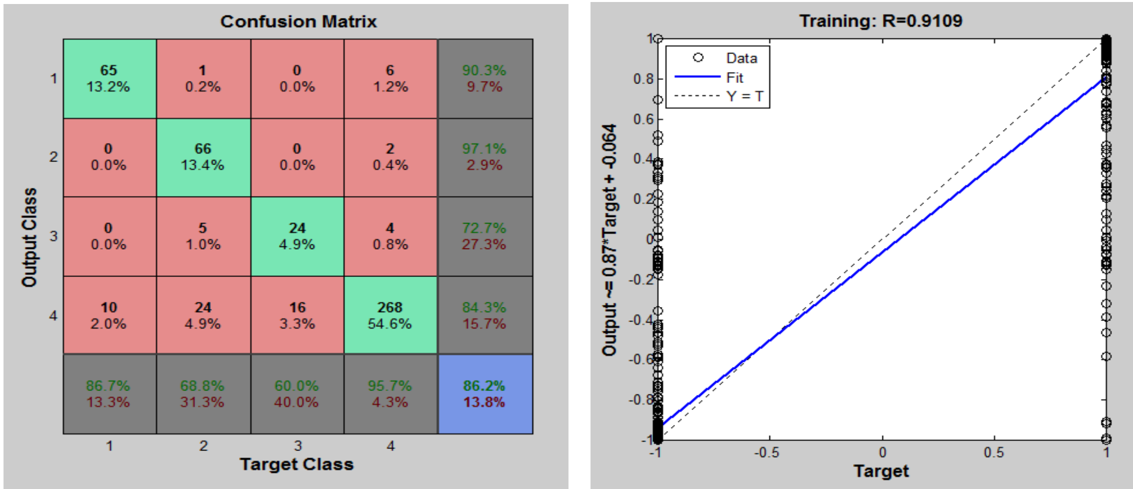
4. The Resilient Back-Propagation (RP) algorithm method was used in order to compare the various training algorithm and select the best in terms of speed and performance. The training stopped after 140/1000 epochs. A MSE of 0.10191 was obtained in 5 seconds. The correlation coefficient in this case was 0.67666 with an overall confusion matrix of 60.5%. Testing the network using untrained data set resulted to an accuracy of 69%.

5. Training with the One Step Secant (OSS) algorithm stopped after 4 seconds and took 63 epochs. A MSE of 0.11737 was obtained; the correlation coefficient obtained for this network was 0.63244 with an overall confusion matrix of 58.7%. Testing the network using untrained data set resulted to an accuracy of 53%.



(a) (b)

Figure 6.15: (a) Performance plot for 4-10-4; and **(b)** ROC plot for 4-10-4

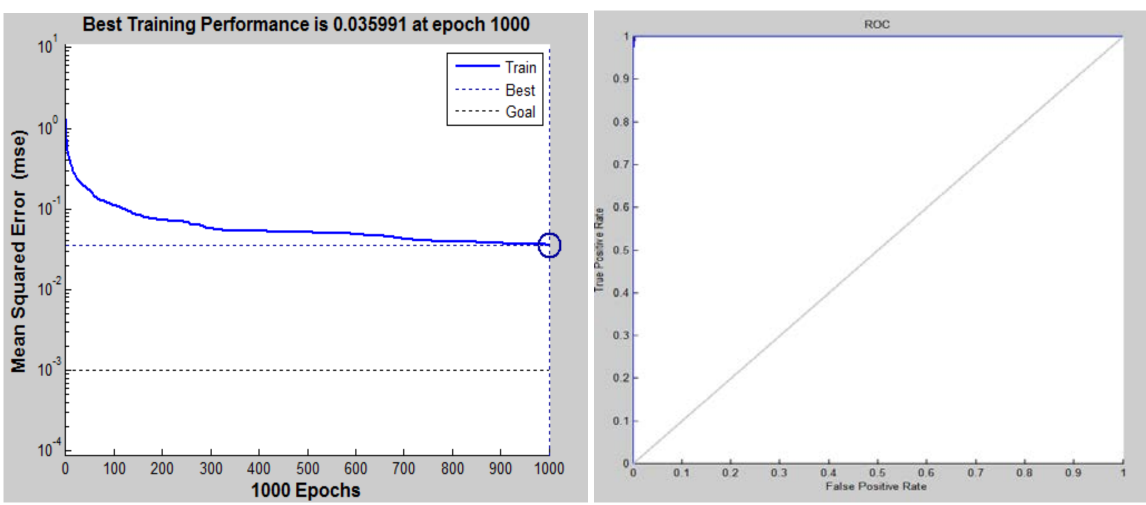


(a) (b)

Figure 6.16: (a) Confusion matrix for 4-10-4; and **(b)** Regression plot for 4-10-4

6.5.1.7 Network 7 (4-18-18-4)

Network 7 was implemented with two hidden layers with 18 neurons in each respectively. This network was trained using the L-M algorithm for 1000 iterations. A MSE of 0.036 was obtained; the overall confusion matrix was 89.8% with a correlation coefficient of 0.9793. Figures 6.17 and 6.18 show the performance plots obtained.

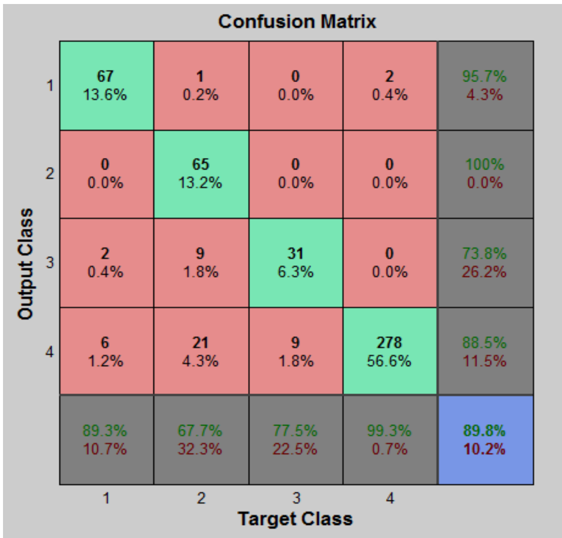


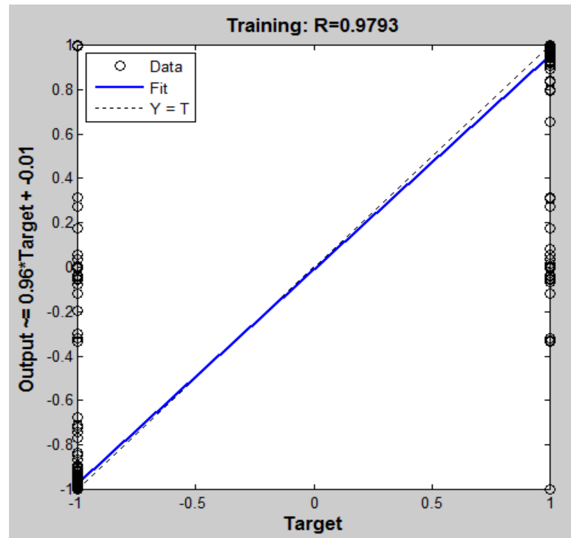
(a) (b)

Figure 6.17: (a) Performance plot for 4-18-18-4; (b) ROC plot for 4-18-18-4

This network was selected as the best based on the performance indices hitherto mentioned. Also, the early-stopping and regularization methods were also used to validate the results.

Furthermore, tests have been carried out on the trained network using a hold-out (untrained) test sample. This untrained test set was used to evaluate the network performance. Results show that the generalization was good with a classification accuracy of 89%.





(a) (b)

Figure 6.18: (a) Confusion matrix for 4-18-18-4; and **(b)** Regression plot for 4-18-18-4

6.5.1.8 Discussion of the Results

From the foregoing, it was observed that the use of two hidden layers with 18 neurons was a good choice for the problem. The final structure is shown in Figure 6.19.

The following tables (Tables 6.21-6.24) summarize the results of training the network using five different training algorithms. Each entry in the table represents 10 different trials, where different random initial weights are used in each trial. In each case, the network is trained using a MSE of 0.001. The best algorithm for this problem was the Levenberg-Marquardt algorithm.

From these tables, it can be seen that the best confusion matrix was obtained using Levenberg-Marquardt (L-M) algorithm. Results also show that the Mean Square Error (MSE) and confusion matrix improved with increase in the number of hidden layers and hidden layer neurons.

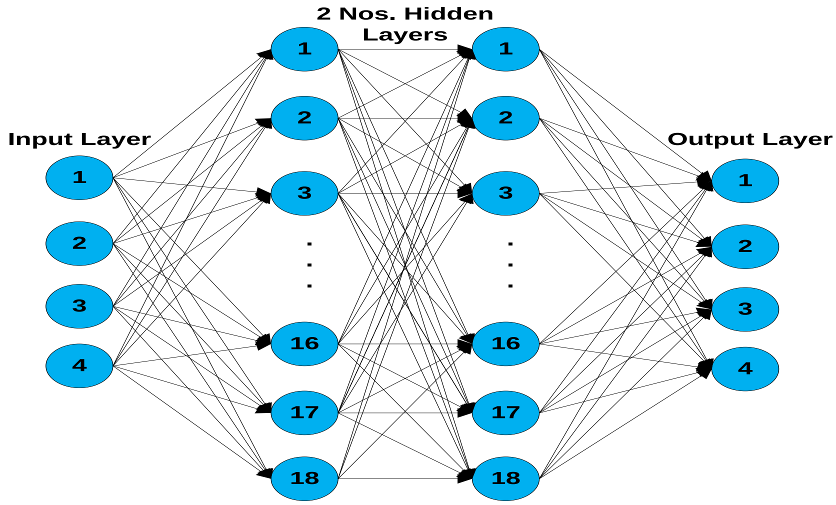


Figure 6.19: **Final ANN for single phase fault section identification (4-18-18-4)**

However, the guideline used in the thesis was to select the smallest neural network with good performance. This is because bigger networks easily over-learn the training dataset and would fail to generalise when presented with untrained dataset.

Table 6.21: **Analysis of the Confusion Matrix Results for the Chosen Architecture**

Algorithm	Mean confusion matrix (%)	Min. confusion matrix (%)	Max. confusion matrix (%)
BR	87.32	85.80	88.70
LM	87.57	86.80	89.00
SCG	83.52	81.90	85.70
RP	86.64	83.70	88.60
OSS	84.41	79.80	87.00

Table 6.22: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.9284	0.9141	0.9556
LM	0.9676	0.9541	0.9827
SCG	0.8876	0.7596	0.9142
RP	0.8808	0.8637	0.8913
OSS	0.8639	0.7428	0.8897

A summary of the learning algorithm, activation function, number of hidden layer neurons, and confusion matrix are shown in Table 6.23. Similarly, Table 6.24 presents the effect of increasing the number of hidden layers.

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.23: **Effect of varying the number of hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Tansig*	55	0.005	0.1	10	0.0743	89.6
MLP	LM	*Tansig/Tansig*	50	0.005	0.1	10	0.0755	89.0
MLP	LM	*Tansig/Tansig*	45	0.005	0.1	10	0.0611	89.4
MLP	LM	*Tansig/Tansig*	40	0.005	0.1	10	0.0713	89.8
MLP	LM	*Tansig/Tansig*	35	0.005	0.1	10	0.0905	89.0
MLP	LM	*Tansig/Tansig*	30	0.005	0.1	10	0.0744	89.0
MLP	LM	*Tansig/Tansig*	25	0.005	0.1	10	0.0735	90.0
MLP	LM	*Tansig/Tansig*	20	0.005	0.1	10	0.0952	89.6
MLP	LM	*Tansig/Tansig*	15	0.005	0.1	10	0.114	89.8
MLP	LM	*Tansig/Tansig*	10	0.005	0.1	10	0.199	87.6
MLP	LM	*Tansig/Tansig*	5	0.005	0.1	10	0.127	87.8

Table 6.24: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Confusion matrix (%)	Regression
1	5	LM	*Tansig*	0.279	87.6	0.82628
1	10	LM	*Tansig*	0.164	89.6	0.90370
1	15	LM	*Tansig*	0.164	87.4	0.93264
2	5	LM	*Tansig/Tansig*	0.115	88.2	0.90314
2	10	LM	*Tansig/Tansig*	0.087	86.6	0.94914
2	15	LM	*Tansig/Tansig*	0.076	89.8	0.95620

6.5.2 Network Simulation for Two Phase Faults

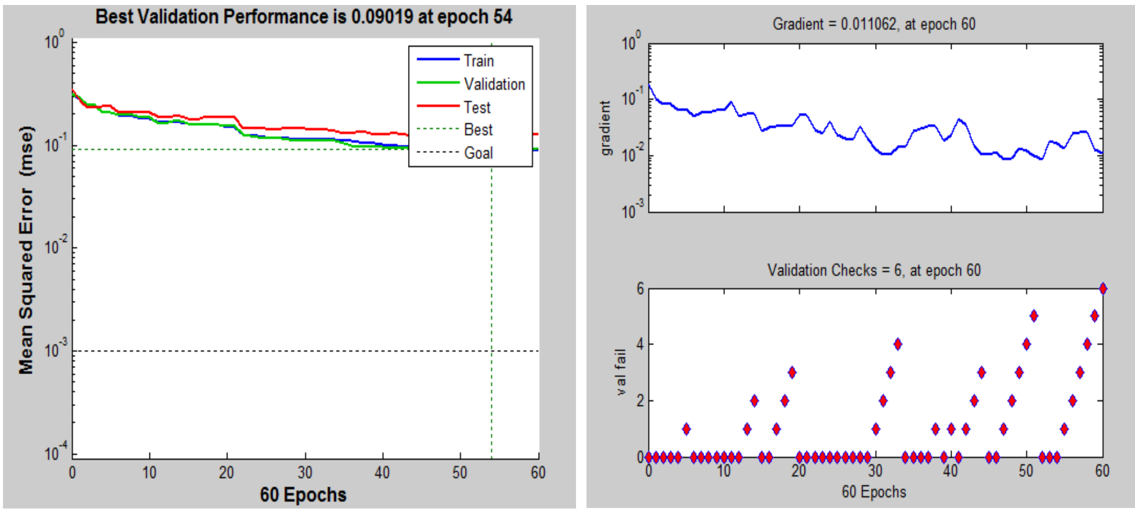
6.5.2.1 Network 1 [4-5-4]

The first network trained had a single hidden layer with 5 neurons. The target MSE was 0.001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* respectively.

The performance plot in Figure 6.20(a) shows that the training converged slowly. Figure 6.20(b) presents the training states showing the gradient and validation vs. Epoch plots.

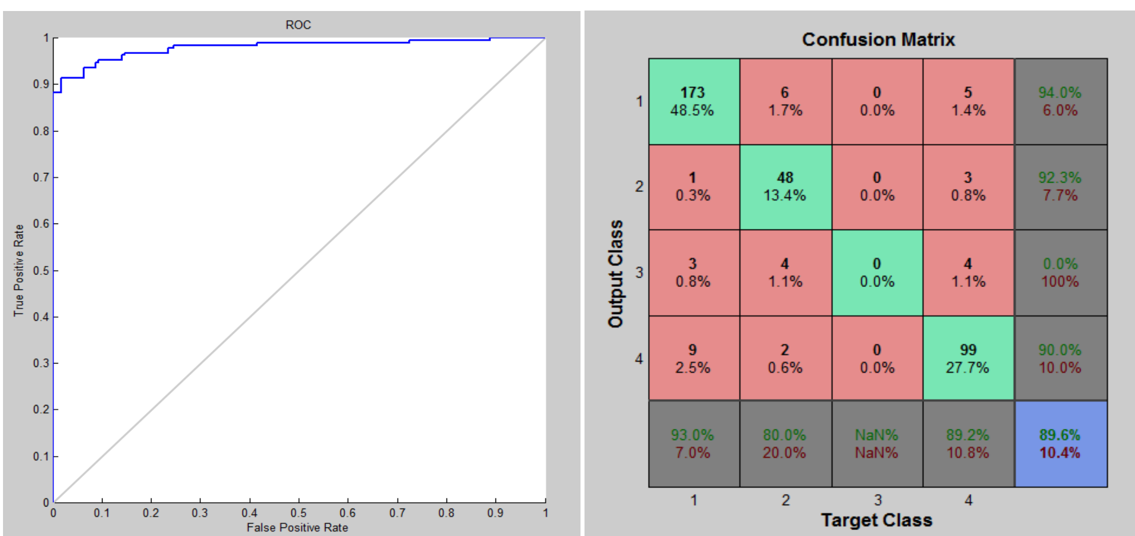
A number of performance analyses were carried out on the network. The ROC was used to check the quality of classification. The ROC plot in Figure 6.21(a) shows the curve tending to the left, but away from the top edge of the plot. Better accuracy would be obtained if the curve moves towards the top left corner.

The confusion matrices for the various types of errors that occurred for the trained neural network was obtained as shown in Figure 6.21(b).



(a) (b)

Figure 6.20: (a) Performance plot for 4-5-4; and **(b)** Training states for 4-5-4



(a) (b)

Figure 6.21: (a) ROC plot for 4-5-4; and **(b)** Confusion matrix for 4-5-4

The last cell in blue indicates the total percentage of cases that have been classified correctly in green while the red signifies incorrectly classified cases. It can be seen that the neural network has 89.6% accuracy in fault section identification.

Plots of the best linear regression that relates the targets to the outputs are as shown in Figure 6.22. The correlation coefficient was 0.76868.

The forth validation method in the testing process was to create a separate test (untrained) dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different location at different fault resistance and fault inception angles. These parameters are different from that used in the training dataset. The result obtained from the test dataset showed an accuracy of 79%.

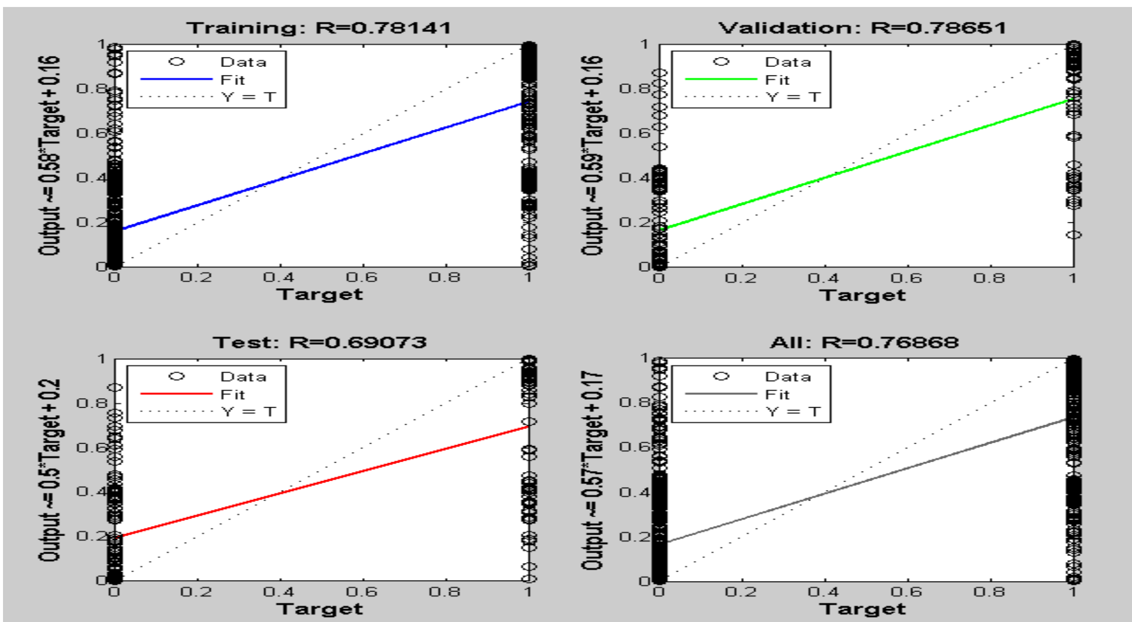


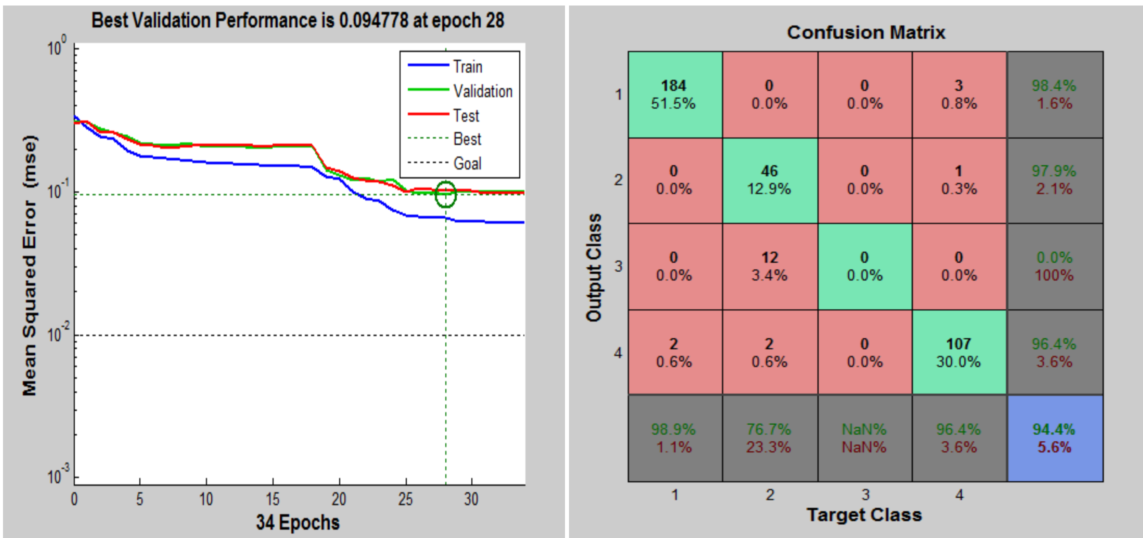
Figure 6.22: **Regression plots for 4-5-4**

Bayesian Regularization (BR) algorithm was also implemented for this network. Compared with the result obtained for the network above using SCG algorithm, the network using BR algorithm had better performance in terms of goal, regression, and confusion matrix. The goal achieved was 0.079447 in 31 iterations, while the correlation coefficient was 0.82897, with an overall confusion matrix of 95.8%. Testing the network using untrained data set resulted to an accuracy of 83%.

6.5.2.2 Network 2 [4-10-4]

This network had a single hidden layer with 10 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.23, 6.24, and 6.25 respectively. The performance goal achieved was 0.094778 in 2 iterations while the correlation coefficient in this case was 0.82943 with an overall confusion matrix of 94.4%. Testing the network using untrained data set resulted to an accuracy of 79%.



(a) **(b)**

Figure 6.23: (a) **Performance plot for 4-10-4; and (b) Confusion matrix for 4-10-4**

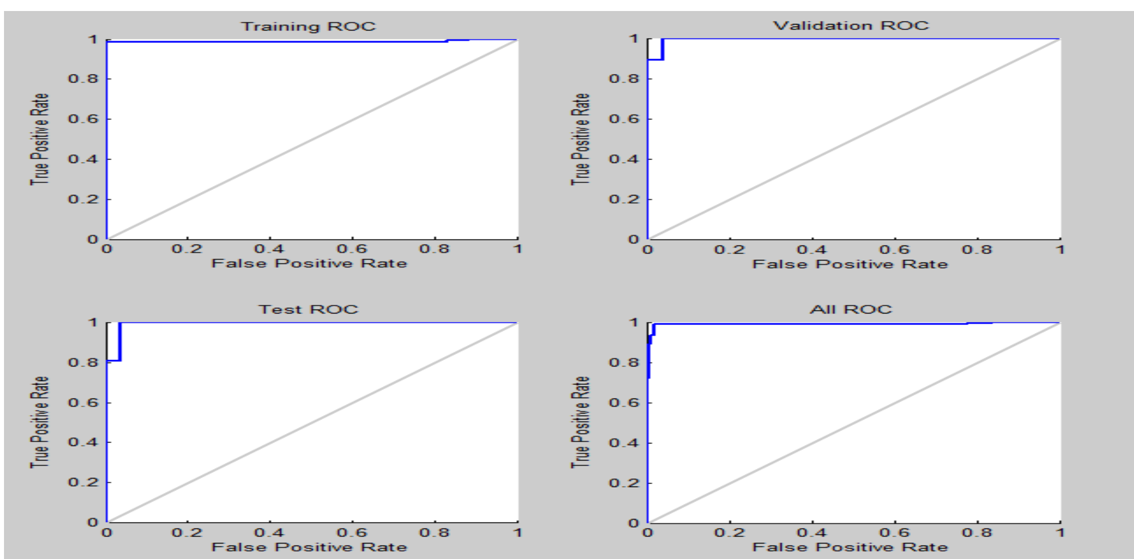


Figure 6.24: ROC plots for 4-10-4

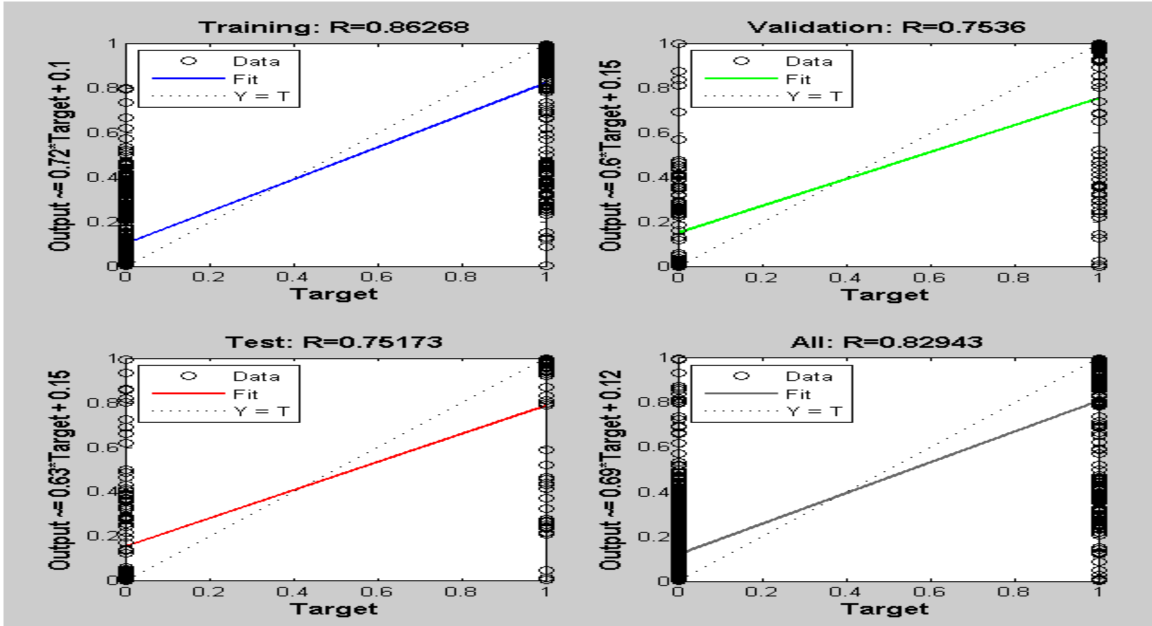
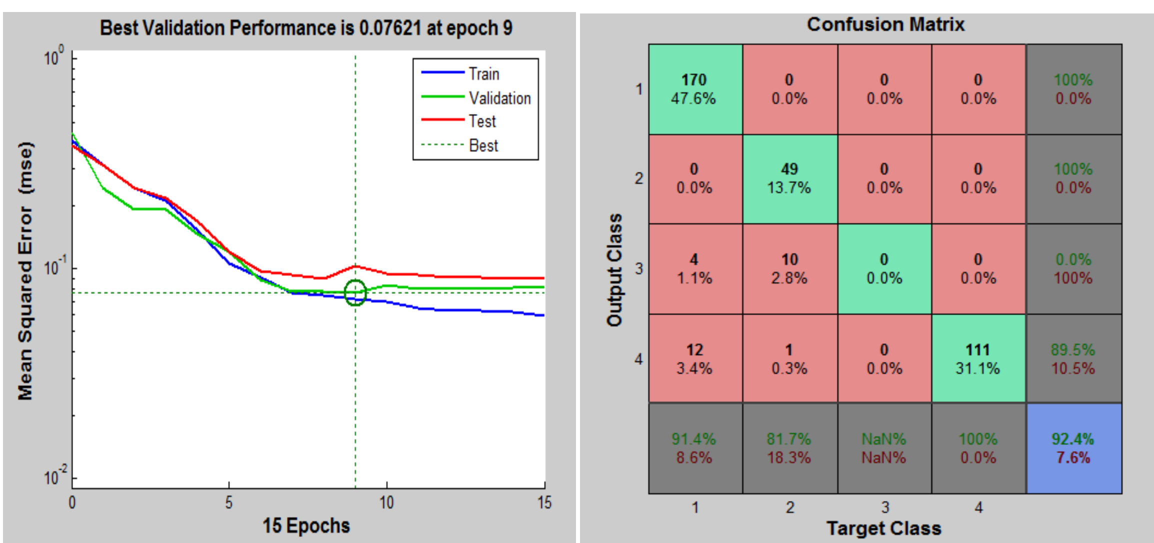


Figure 6.25: **Regression plots for 4-10-4**

6.5.2.3 Network 3 [4-25-4]

This network had a single hidden layer with 25 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.26-6.28 respectively. The goal achieved was 0.07621 in 26 iterations while the overall correlation coefficient was 0.84697 with an overall confusion matrix of 92.4%. Testing the network using untrained data set resulted to an accuracy of 81%.



(a) (b)

Figure 6.26: (a) **Performance plot for 4-25-4; and (b) Confusion matrix for 4-25-4**

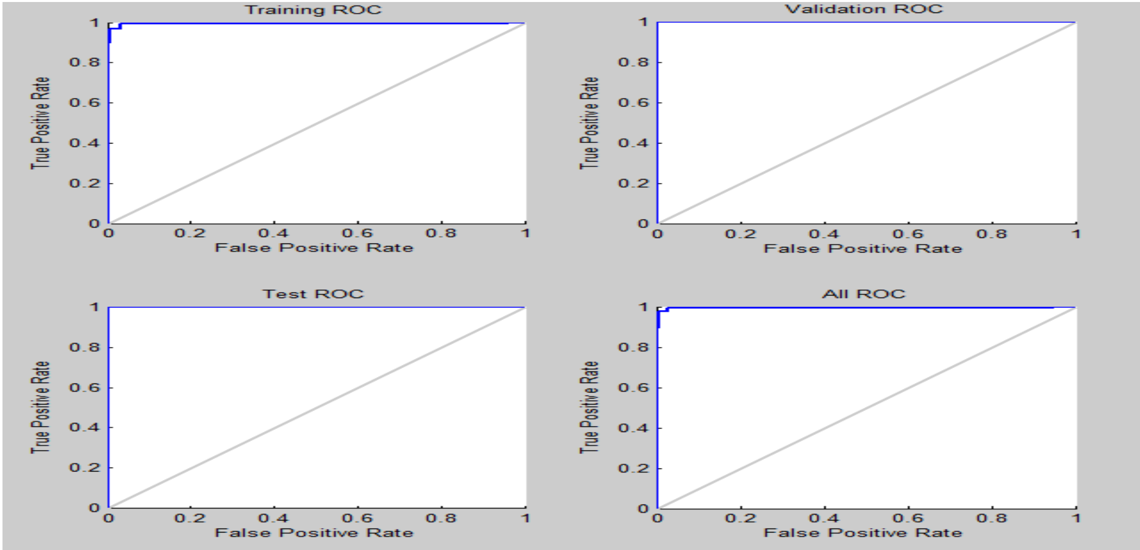


Figure 6.27: ROC plot for 4-25-4

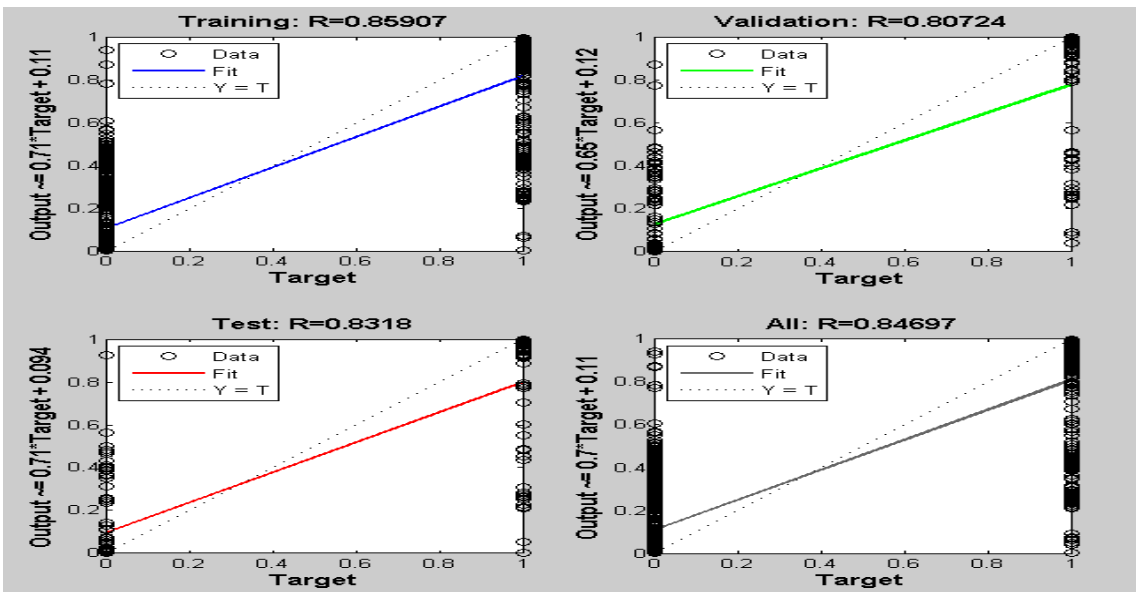
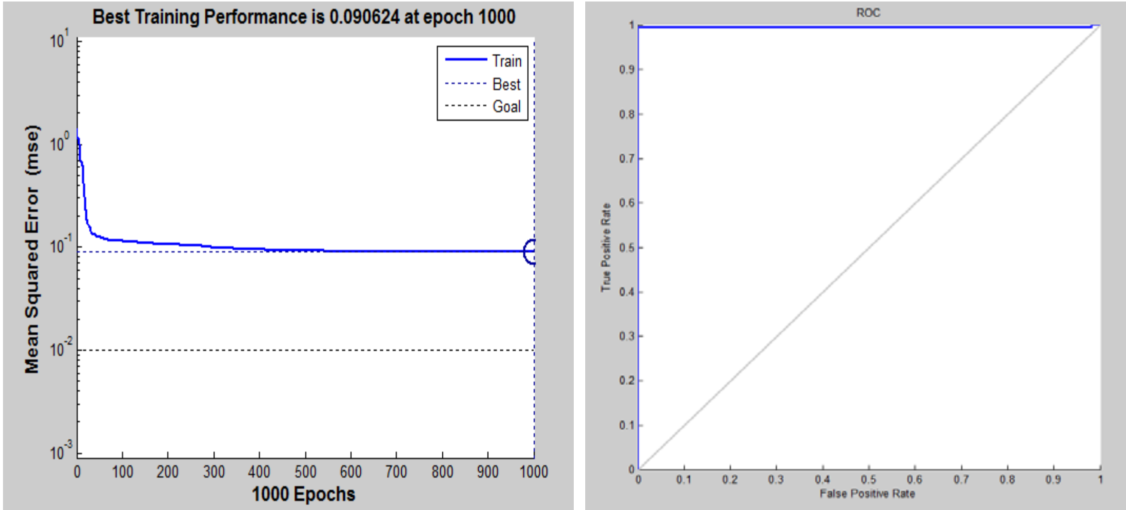


Figure 6.28: Regression plots for 4-25-4

6.5.2.4 Network 4 [4-5-5-4]

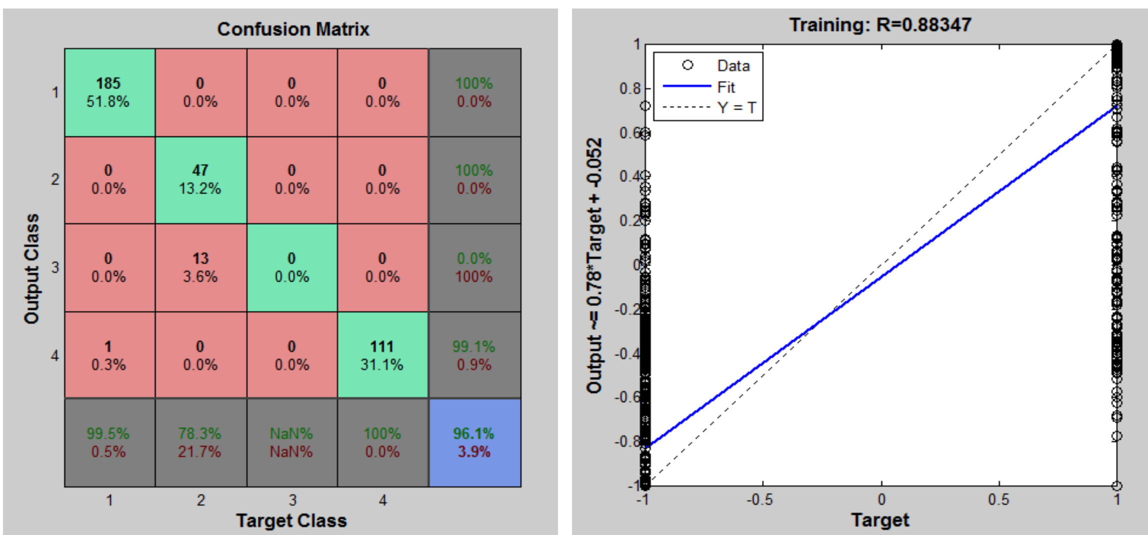
This network had two hidden layers with 5 neurons respectively. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.29-6.30 respectively. The goal achieved was 0.90624 at the end of 1000 iterations, while the correlation coefficient in this case was 0.88347 with an overall confusion matrix of 96.1%. Testing the network using untrained data set resulted to an accuracy of 84%.



(a) **(b)**

Figure 6.29: (a) Performance plot for 4-5-5-4; and **(b)** ROC plot for 4-5-5-4



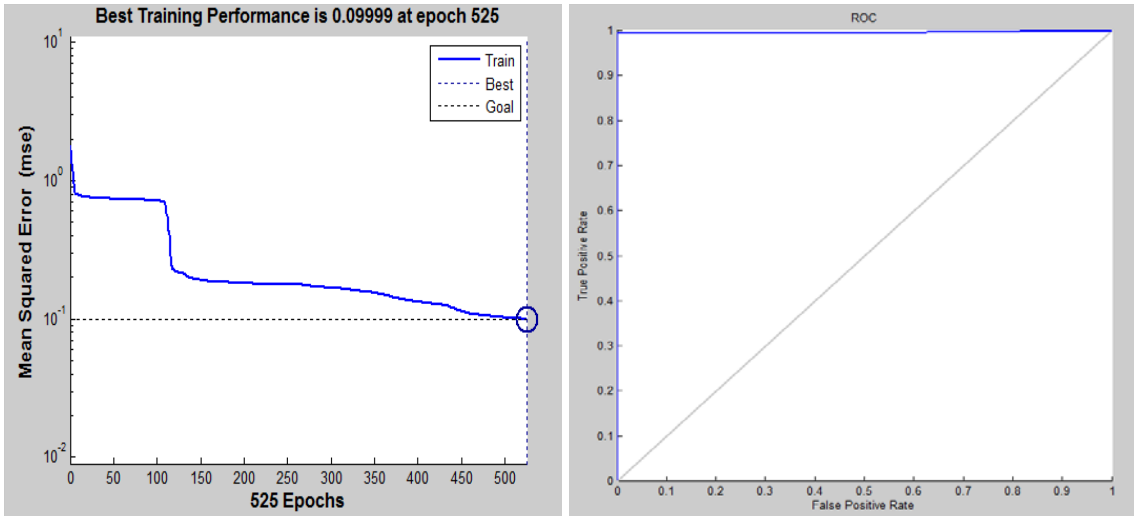
(a) **(b)**

Figure 6.30 (a) Confusion matrix for 4-5-5-4; **(b)** Regression plot for 4-5-5-3

6.5.2.5 Network 5 [4-10-10-4]

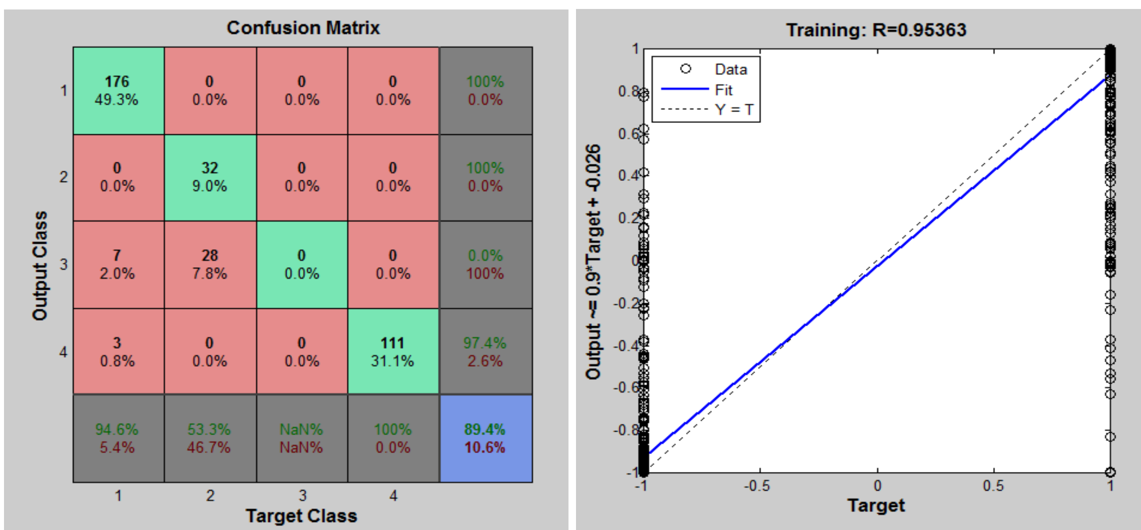
This network had two hidden layers with 10 neurons respectively. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.31 and 6.32 respectively. The goal achieved was 0.099997 at 525 iterations, while the correlation coefficient in this case was 0.95363 with an overall confusion matrix of 89.4%. Testing the network using untrained data set resulted to an accuracy of 86%.



(a) (b)

Figure 6.31: (a) Performance plot for 4-10-10-4; and **(b)** ROC plot for 4-10-10-4



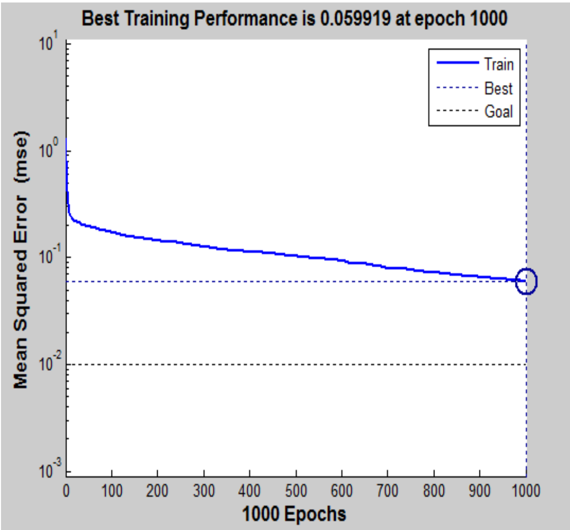
(a) (b)

Figure 6.32: (a) Confusion matrix for 4-10-10-4; and **(b)** Regression plot for 4-10-10-4

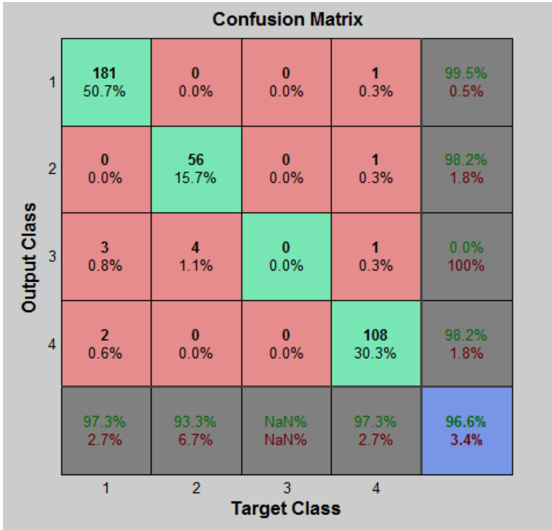
6.5.2.6 Network 6 [4-21-4]

A network having a single hidden layer with 21 neurons was also investigated. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figure 6.33. The correlation coefficient in this case was 0.93291 with an overall confusion matrix of 96.6%. Testing the network using untrained data set resulted to an accuracy of 91%.



(a)



(b)

Figure 6.33: (a) Performance plot for 4-21-4; and **(b)** Confusion matrix for 4-21-4

6.5.2.7 Discussion of the Results

Result obtained indicates that a single hidden layer with 21 neurons was a good choice for the two phase fault section identification task. The final structure is shown in Figure 6.34.

Tables 6.25-6.28 summarize the results of training the network using five different training algorithms. Each entry in the table represents 10 different trials, where different random initial weights are used in each trial. In each case, the network is trained using a MSE of 0.001. A summary of the learning algorithm, activation function, number of hidden layer neurons, and confusion matrix are shown in Table 6.27. Table 6.28 present the effect of increasing the number of hidden layers.

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for mu respectively.

From these tables, it can be seen that the best confusion matrix was obtained using Levenberg-Marquardt (L-M) algorithm. Generally, the Mean Square Error (MSE) improved with increase in the number of hidden layers and hidden layer neurons. However, the confusion matrix was not dependent on the number of hidden layers or hidden layer neurons.

Table 6.25: Analysis of the Confusion Matrix Results for the Chosen Architecture

Algorithm	Mean confusion matrix (%)	Min. confusion matrix (%)	Max. confusion matrix (%)
BR	84.9	83.8	87.1
LM	92.4	82.4	96.6
SCG	88.8	86.0	93.3
RP	96.8	95.5	98.0
OSS	90.1	81.2	97.2

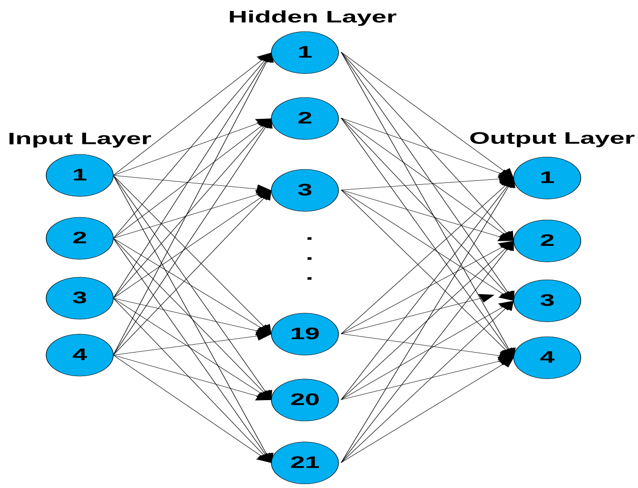


Figure 6.34: Final ANN for 2 Ph. fault section identification (4-21-4)

Table 6.26: Analysis of the Regression Results for the Chosen Architecture

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.8877	0.8854	0.8898
LM	0.9374	0.9043	0.9506
SCG	0.8741	0.8674	0.8783
RP	0.8589	0.8522	0.8652
OSS	0.8429	0.7661	0.8562

6.5.3 Network Simulation for Two Phase-Ground Faults

6.5.3.1 Network 1 [4-5-4]

The first network trained had a single hidden layer with 5 neurons. The target MSE was 0.001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* respectively.

Performance goal of 0.0756 was achieved in 83 iterations. A number of performance analyses were carried out on the network. The first of these was by using the ROC plot.

This plot as given in Figure 6.35(a) shows the plot tending away from the left edge. Thus, the accuracy would not be great.

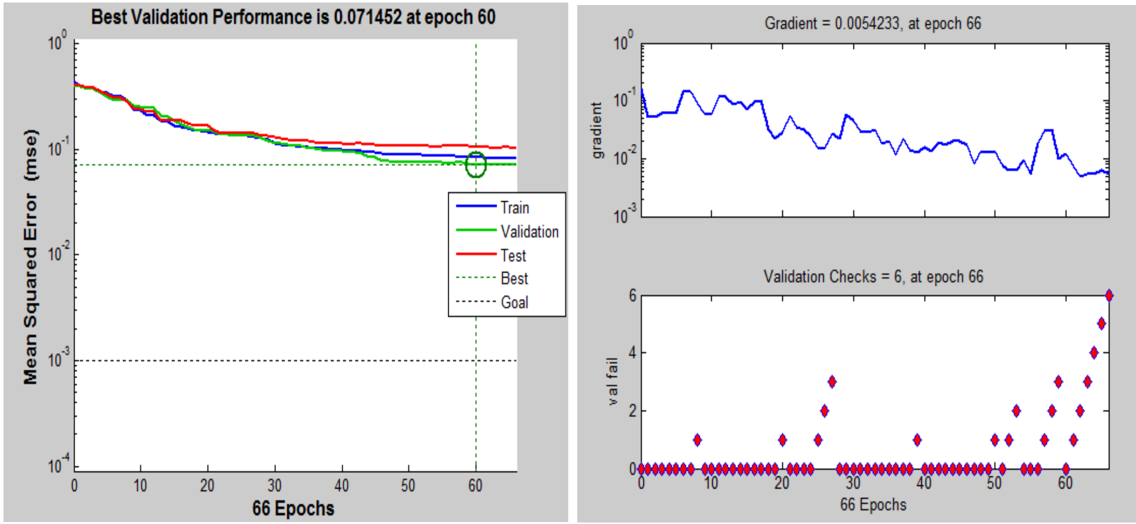
Table 6.27: **Effect of varying the number of hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Tansig*	5	0.005	0.1	10	0.2510	95.8
MLP	LM	*Tansig/Tansig*	10	0.005	0.1	10	0.1750	93.8
MLP	LM	*Tansig/Tansig*	15	0.005	0.1	10	0.1350	93.6
MLP	LM	*Tansig/Tansig*	20	0.005	0.1	10	0.1180	95.0
MLP	LM	*Tansig/Tansig*	25	0.005	0.1	10	0.0106	90.5
MLP	LM	*Tansig/Tansig*	30	0.005	0.1	10	0.0681	94.7
MLP	LM	*Tansig/Tansig*	35	0.005	0.1	10	0.0181	92.4
MLP	LM	*Tansig/Tansig*	40	0.005	0.1	10	0.0959	84.3
MLP	LM	*Tansig/Tansig*	45	0.005	0.1	10	0.0606	85.4
MLP	LM	*Tansig/Tansig*	50	0.005	0.1	10	0.0576	87.5
MLP	LM	*Tansig/Tansig*	55	0.005	0.1	10	0.0826	90.2

Table 6.28: **Effect of increasing the number of hidden layers**

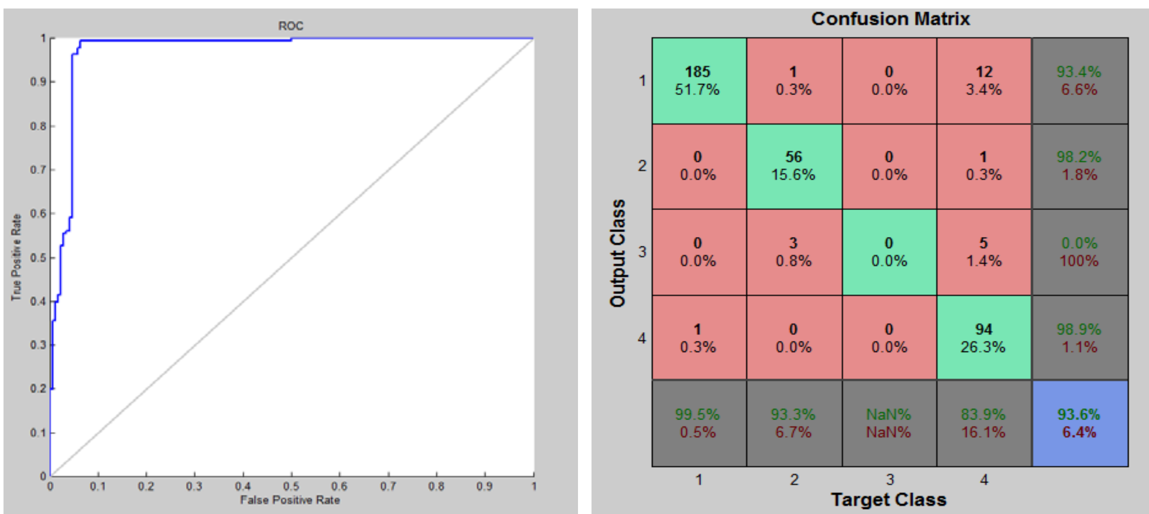
No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Confusion matrix (%)	Regression
1	5	LM	*TANSIG*	0.279	96.1	0.84300
1	10	LM	*TANSIG*	0.215	94.4	0.82722
1	15	LM	*TANSIG*	0.163	95.4	0.91061
2	5	LM	*TANSIG/TANSIG*	0.196	98.6	0.89135
2	10	LM	*TANSIG/TANSIG*	0.090	92.4	0.95142
2	15	LM	*TANSIG/TANSIG*	0.067	89.4	0.96424

The second means of testing the performance of the neural network was the confusion matrices for the various types of errors that occurred for the trained neural network. Figure 6.36(b) shows the confusion matrix for the NN training. It can be seen that the neural network has 93.6 percent accuracy in fault section identification. Plotting the best linear regression that relates the targets to the outputs as shown in Figure 6.37. The correlation coefficient in this case has been found to be 0.8206. The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different location at different fault resistance and fault inception angles. These parameters are different from that used in the training dataset. The result obtained from the test dataset showed an accuracy of 72%.



(a) (b)

Figure 6.35: (a) Performance plot for 4-5-4; and **(b)** Training states for 4-5-4



(a) (b)

Figure 6.36: (a) ROC plot for 4-5-4; and **(b)** Confusion matrix for 4-5-4

6.5.3.2 Network 2 [4-5-4]

The training algorithm used of the previous network was replaced with the Levenberg-Marquardt (L-M) algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plot and confusion matrix are given in Figures 6.37(a) and 6.37(b) respectively. The goal achieved was 0.010217, while the correlation coefficient in this case has been found to be 0.83592 with an overall confusion matrix of 95.3%. The result obtained from the test dataset showed an accuracy of 79%.

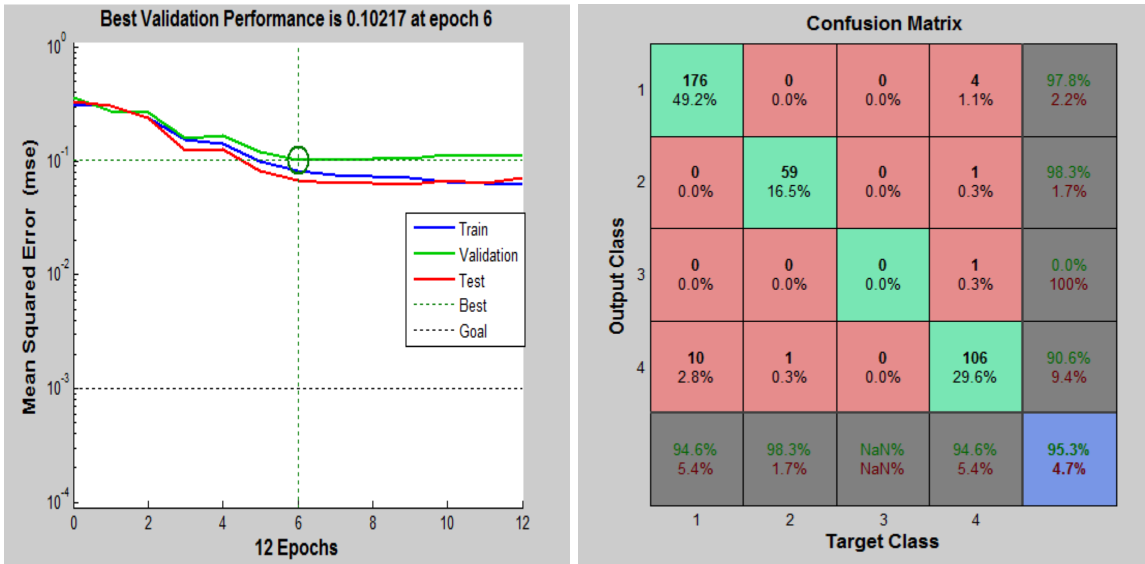


Figure 6.37: **(a) Performance plot for 4-5-4; and** **(b) Confusion matrix for 4-5-4**

6.5.3.3 Network 3 [4-5-4]

The above network was also trained with Bayesian Regularization (BR) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

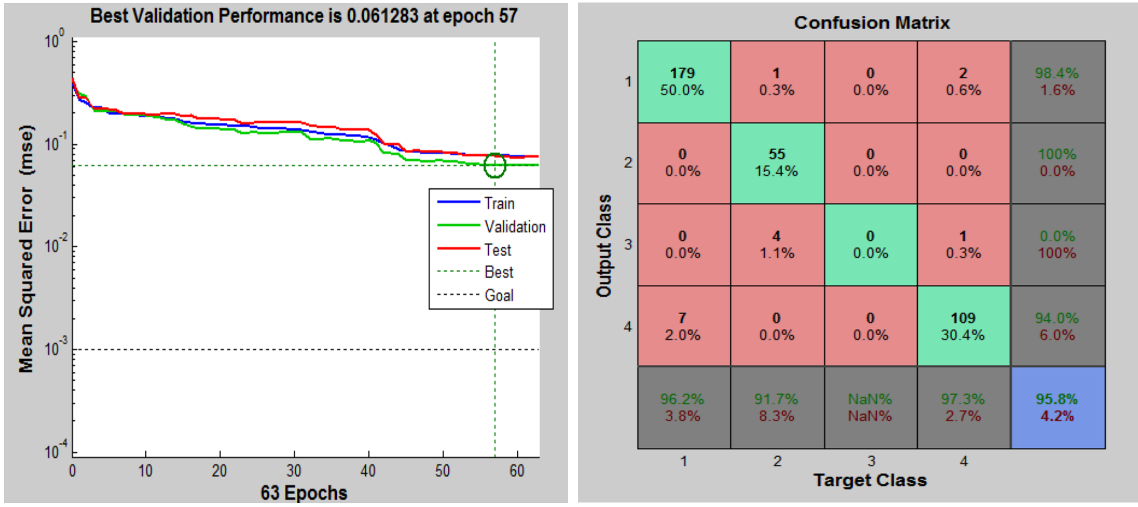
The SSE goal achieved was 90.2 in 265 iterations, while the correlation coefficient in this case has been found to be 0.85745 with an overall confusion matrix of 98.0%. The result obtained from the test dataset showed an accuracy of 85%

Reducing the MSE to 0.01 with training algorithm as BR algorithm, gave a MSE goal of 97.7 SSE in 210 iterations, while the correlation coefficient in this case has been found to be 0.86365 with an overall confusion matrix of 99.4%. The result obtained from the test dataset showed an accuracy of 85%

6.5.3.4 Network 4 [4-10-4]

This network had a single hidden layer with 10 neurons. The target MSE was 0.001. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig*.

The performance plots are shown in Figures 6.38 and 6.39 respectively. The performance goal achieved was 0.061283 in 57 iterations.



(a) **(b)**

Figure 6.38: (a) Performance plot for 4-10-4; and **(b)** Confusion matrix for 4-10-4

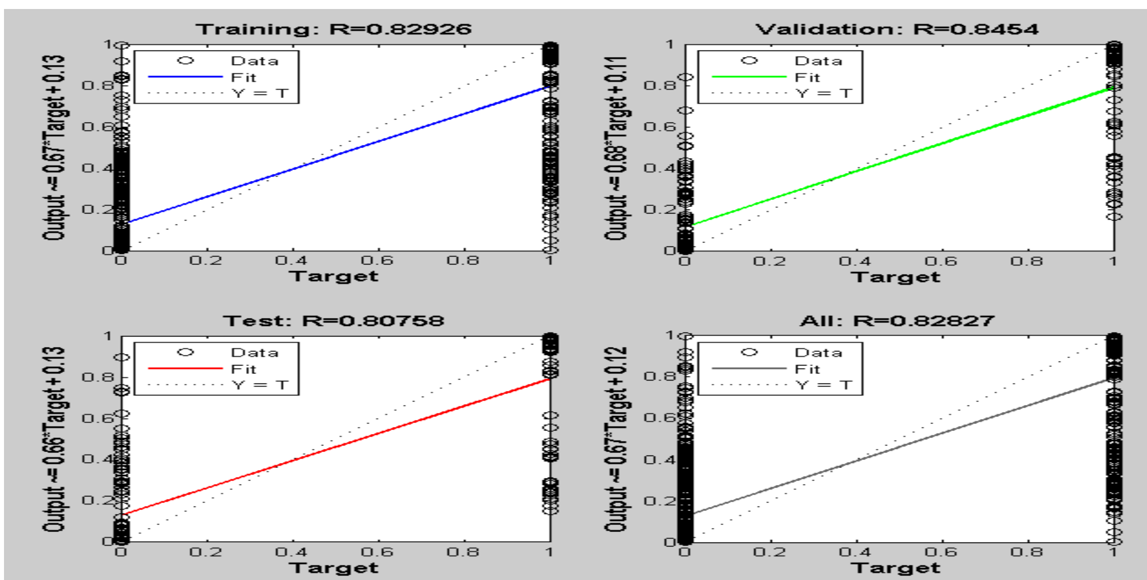


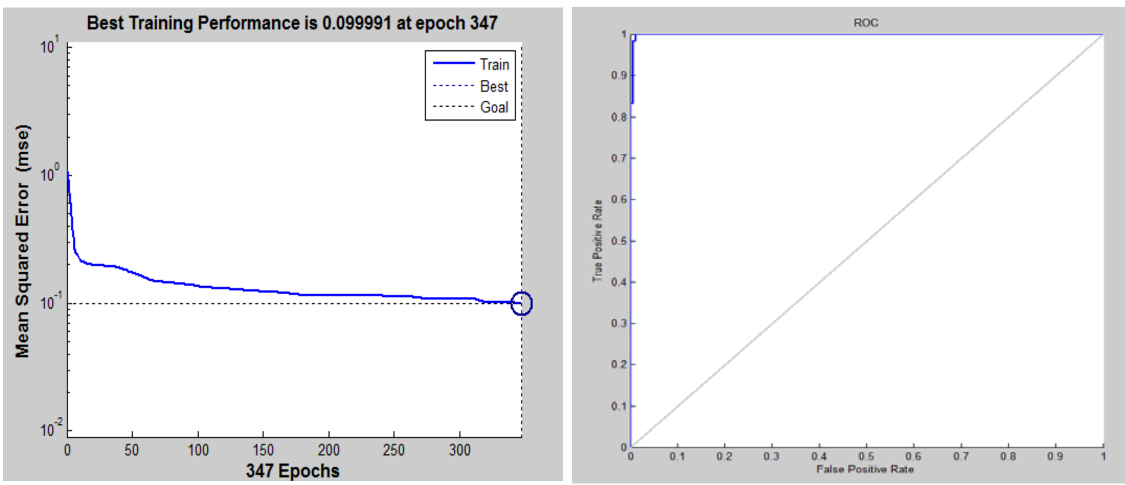
Figure 6.39: Regression plots for 4-10-4

The correlation coefficient in this case has been found to be 0.82827. The second means of testing the performance of the neural network was the confusion matrices for the various types of errors that occurred for the trained neural network. Figure 6.38(b) shows the confusion matrix for the NN training. It can be seen that the neural network has 95.8 percent accuracy in fault section identification. The result obtained from the test dataset showed an accuracy of 91%.

Reducing the target MSE to 0.01 and retraining gave a goal of 0.087885 in 15 iterations, while the correlation coefficient in this case has been found to be 0.82948 with an overall confusion matrix of 97.8%. The result obtained from the test dataset showed an accuracy of 89%

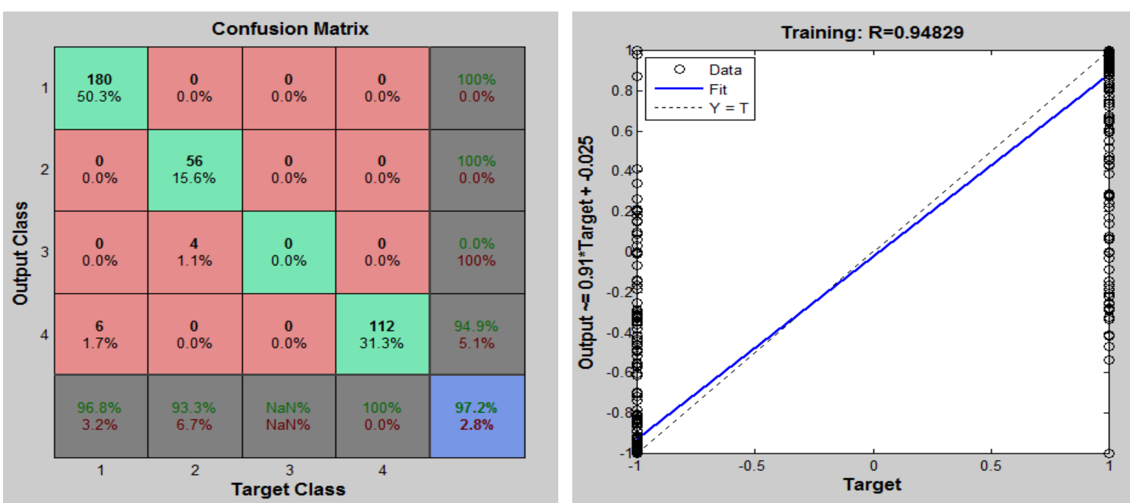
6.5.3.5 Network 5 [4-20-4]

This network had a single hidden layer with 20 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively. The performance plots are as shown in Figures 6.40-6.41. The goal achieved was 0.099991 at the end of 1000 iterations, while the correlation coefficient in this case has been found to be 0.94829 with an overall confusion matrix of 97.2%. The result obtained from the test dataset showed an accuracy of 89%.



(a) (b)

Figure 6.40: (a) Performance plot for 4-20-4; and **(b)** ROC plot for 4-20-4



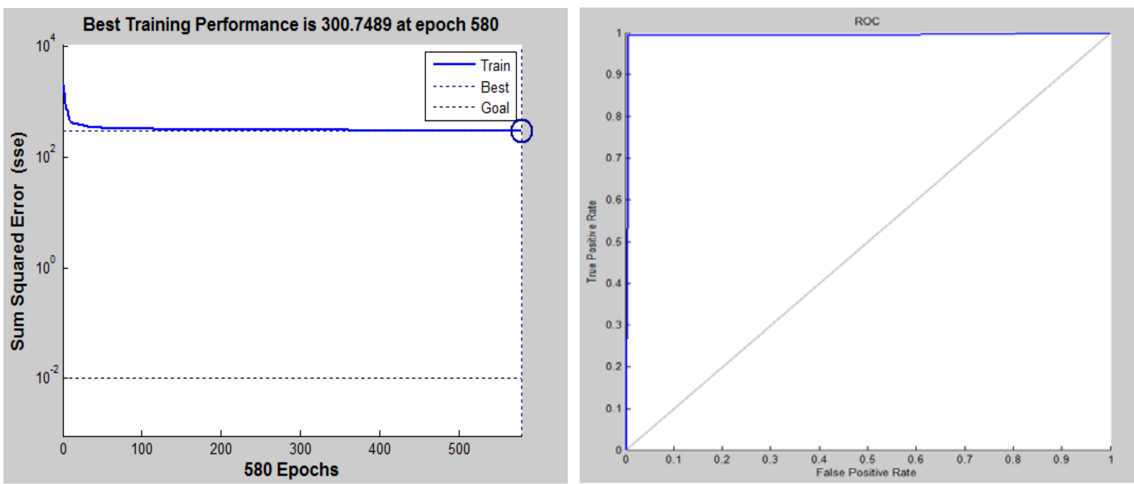
(a) (b)

Figure 6.41: (a) Confusion matrix for 4-20-4; and **(b)** Regression plot for 4-20-4

6.5.3.6 Network 6 [4-5-5-4]

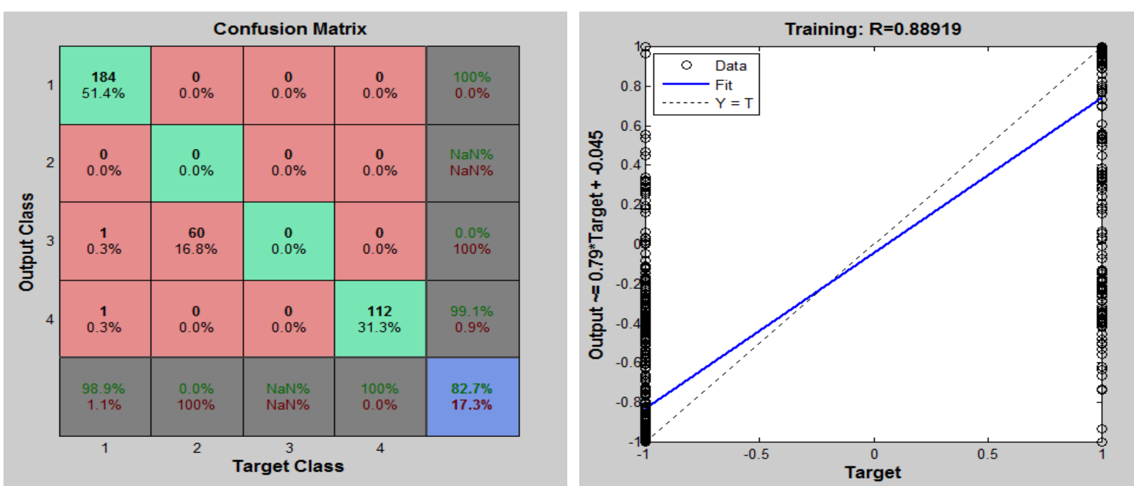
This network had two hidden layers with 5 neurons each. The target MSE was 0.01. The training algorithm used was Bayesian Regularization (BR) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.42 and 6.43 respectively. A maximum ***mu*** (Marquardt adjustment parameter) of 300.75 SSE was reached after 580 iterations, while the correlation coefficient in this case has been found to be 0.88919 with an overall confusion matrix of 82.7%. The result obtained from the test dataset showed an accuracy of 87%.



(a) (b)

Figure 6.42: (a) Performance plot for 4-5-5-4; and **(b)** ROC plot for 4-5-5-4



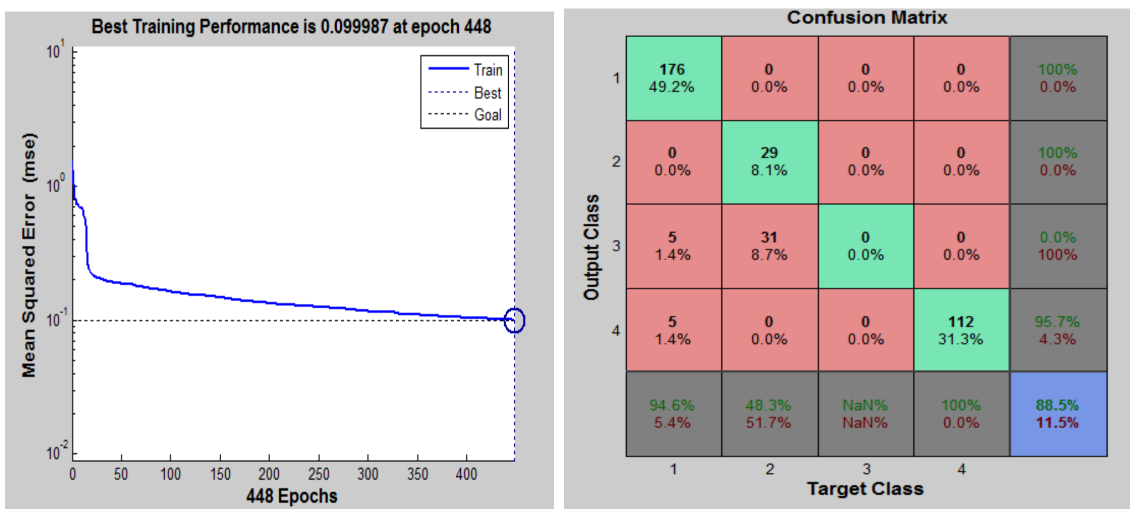
(a) (b)

Figure 6.43: (a) Confusion matrix for 4-5-5-4; and (b) Regression plot for 4-5-5-4

The 4-5-5-4 network was retrained using L-M algorithm. A performance goal of 0.198 was achieved, while the correlation coefficient in this case was 0.88966 with an overall confusion matrix of 93.6%. The result obtained from the test dataset showed an accuracy of 84%.

6.5.3.7 Network 7 [4-10-10-4]

This network had two hidden layers with 10 neurons each. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively. The performance plots are shown in Figure 6.44. The goal achieved was 0.099987, while the correlation coefficient in this case has been found to be 0.95272 with an overall confusion matrix of 88.5%. The result obtained from the test dataset showed an accuracy of 89%.



(a) **(b)**

Figure 6.44: (a) Performance plot for 4-10-10-4; and (b) Confusion matrix for 4-10-10-4

Retraining this network using BR algorithm gave a SSE of 123 after 1000 iterations, while the correlation coefficient was 0.95363 with an overall confusion matrix of 80.4%. The result obtained from the test dataset showed an accuracy of 90%. Figure 6.45 illustrates the final structure of the selected NN.

6.5.3.8 Discussion of the Results

From the foregoing, Network 4 [4-10-4] was chosen as the network of interest for the 2Ph.-G fault section identification task based on its size and the performance obtained when compared to other networks.

Tables 6.29-6.32 summarize the results of training the network 4 [4-10-4] using five different training algorithms. Each entry in the table represents 10 different trials, where different random initial weights are used in each trial. In each case, the network is trained using a MSE of 0.001.

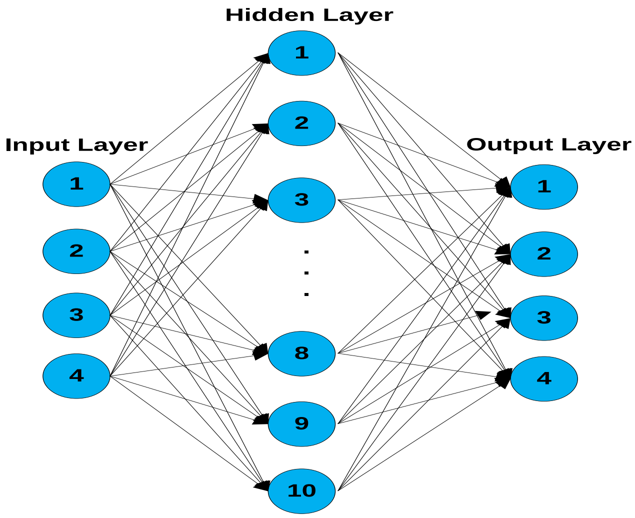


Figure 6.45: **Final ANN for 2Ph.-G fault section identification (4-10-4)**

A summary of the learning algorithm, activation function, number of hidden layer neurons, and confusion matrix are shown in Table 6.31. Table 6.32 presents the effect of increasing the number of hidden layers.

From these tables, the best confusion matrix was obtained using Resilient Propagation (RP) algorithm, while Levenberg-Marquardt (L-M) algorithm was next in line. However, the best coefficient of regression was obtained with L-M algorithm. Also, network training with RP algorithm took a longer time. Generally, the Mean Square Error (MSE) improved with increase in the number of hidden layers and hidden layer neurons. However, the confusion matrix was not dependent on the number of hidden layers or hidden layer neurons.

Table 6.29: **Analysis of the Confusion Matrix Results for the Chosen Architecture**

Algorithm	Mean confusion matrix (%)	Min. confusion matrix (%)	Max. confusion matrix (%)
BR	88.30	86.0	90.2
LM	96.36	94.1	97.8
SCG	89.30	83.0	98.0
RP	98.70	97.2	99.7
OSS	92.60	86.6	97.8

Table 6.30: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.8792	0.8738	0.8814
LM	0.8979	0.8926	0.9016
SCG	0.8452	0.7316	0.8790
RP	0.8640	0.8625	0.8653
OSS	0.8696	0.8675	0.8741

A summary of the learning algorithm, activation function, number of hidden layer neurons, and confusion matrix are shown in Table 6.27.

Table 6.31: **Effect of increasing the number of hidden layers neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Tansig*	5	0.005	0.1	10	0.2500	97.5
MLP	LM	*Tansig/Tansig*	10	0.005	0.1	10	0.1730	98.6
MLP	LM	*Tansig/Tansig*	15	0.005	0.1	10	0.1170	96.6
MLP	LM	*Tansig/Tansig*	20	0.005	0.1	10	0.2730	93.3
MLP	LM	*Tansig/Tansig*	25	0.005	0.1	10	0.0969	91.1
MLP	LM	*Tansig/Tansig*	30	0.005	0.1	10	0.0676	96.4
MLP	LM	*Tansig/Tansig*	35	0.005	0.1	10	0.0639	98.9
MLP	LM	*Tansig/Tansig*	40	0.005	0.1	10	0.0610	81.0

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.32: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Confusion matrix (%)	Regression
1	5	LM	*TANSIG*	0.227	98.9	0.87236
1	10	LM	*TANSIG*	0.177	95.5	0.90246
1	15	LM	*TANSIG*	0.151	97.2	0.91752
2	5	LM	*TANSIG/TANSIG*	0.185	82.7	0.89282
2	10	LM	*TANSIG/TANSIG*	0.112	89.7	0.93945
2	15	LM	*TANSIG/TANSIG*	0.069	88.5	0.96271

6.5.4 Network Simulation for Three Phase Faults

6.5.4.1 Network 1 [4-5-4]

The first network trained had a single hidden layer with 5 neurons. The target MSE was 0.001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.46-6.48 respectively. The goal achieved was 0.23592 in 35 iterations, while the correlation coefficient in this case has been found to be 0.80172 with an overall confusion matrix of 89.1%. The result obtained from the test dataset showed an accuracy of 85%.

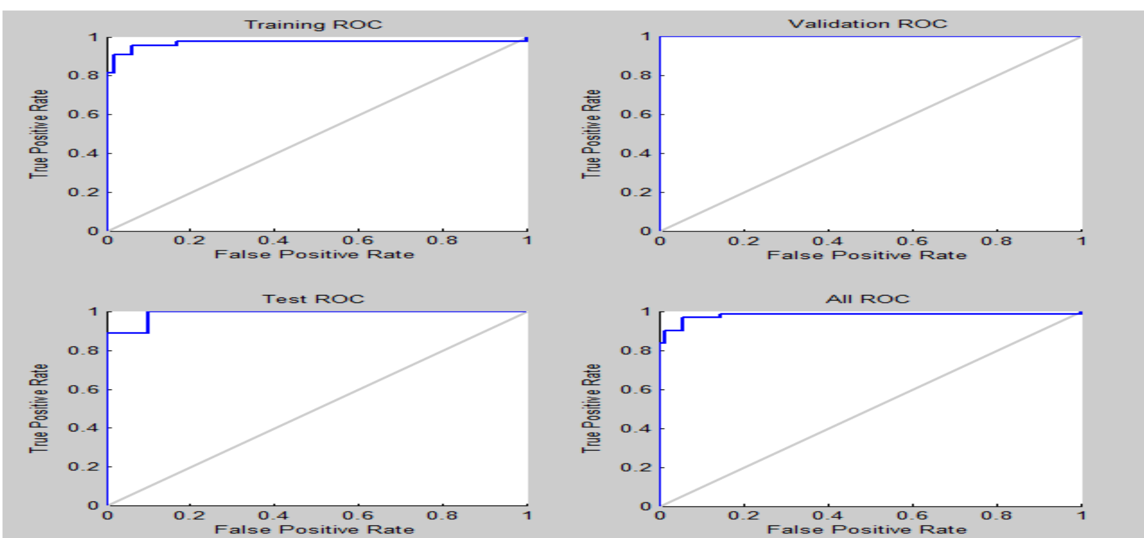
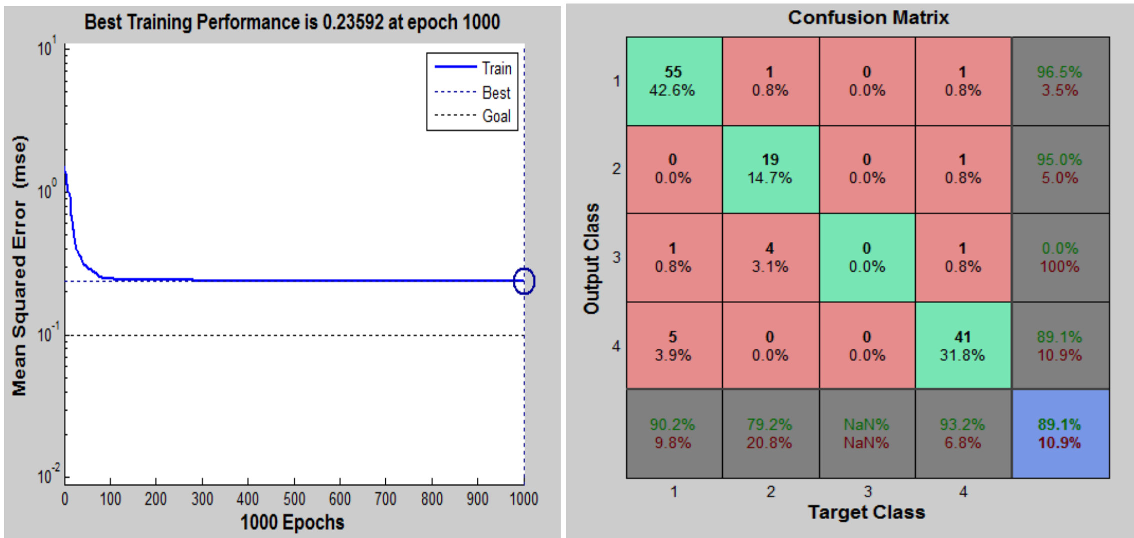


Figure 6.46: **ROC plots for 4-5-4**



(a) **(b)**

Figure 6.47: (a) Performance plot for 4-5-4; and **(b)** Confusion matrix for 4-5-4

Retraining this network with L-M algorithm gave a MSE of 0.0556 in 28 iterations, while the correlation coefficient in this case has been found to be 0.85864 with an overall confusion matrix of 90.7%. The result obtained from the test dataset showed an accuracy of 89%.

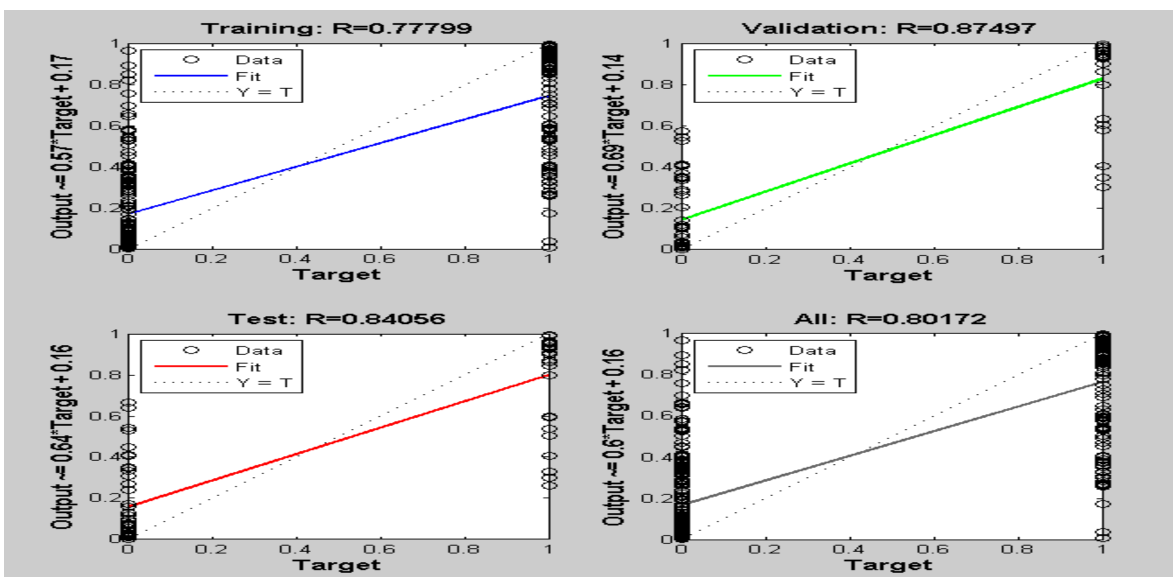
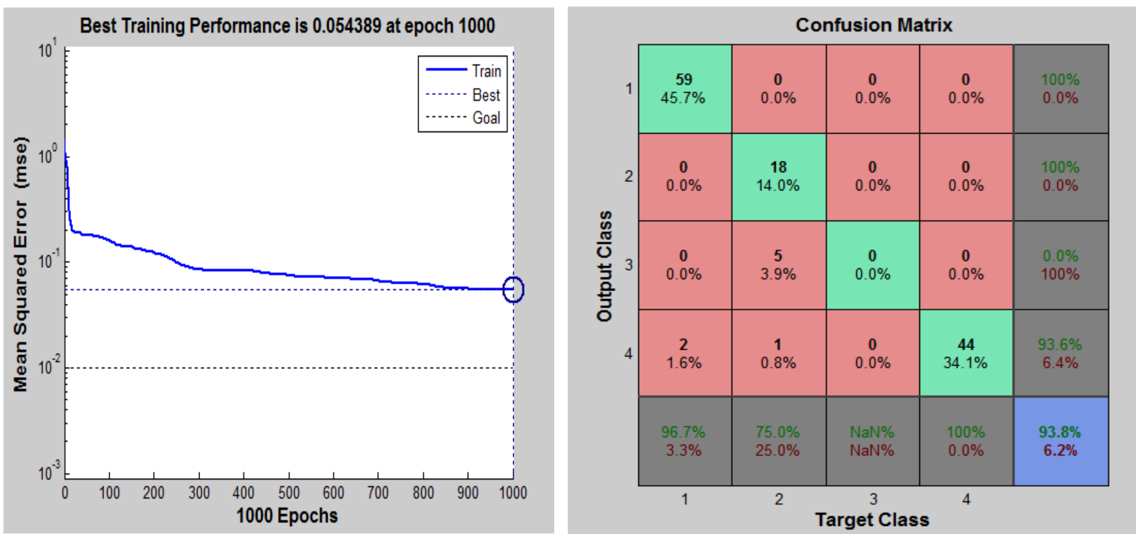


Figure 6.48: Regression plots for 4-5-4

6.5.4.2 Network 2 [4-10-4]

This network had a single hidden layer with 10 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.49-6.51 respectively. The goal achieved was 0.054389, while the correlation coefficient in this case has been found to be 0.85452 with an overall confusion matrix of 93.8%. The result obtained from the test dataset showed an accuracy of 96%.



(a) **(b)**

Figure 6.49: (a) Performance plot for 4-10-4; and **(b)** Confusion matrix for 4-10-4

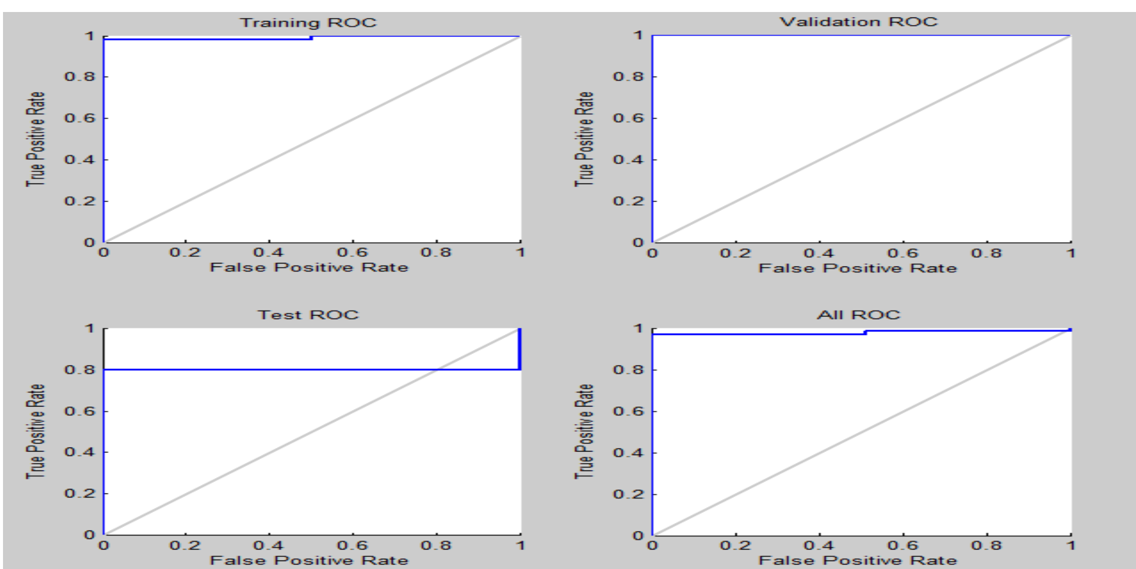


Figure 6.50: ROC plot for 4-10-4

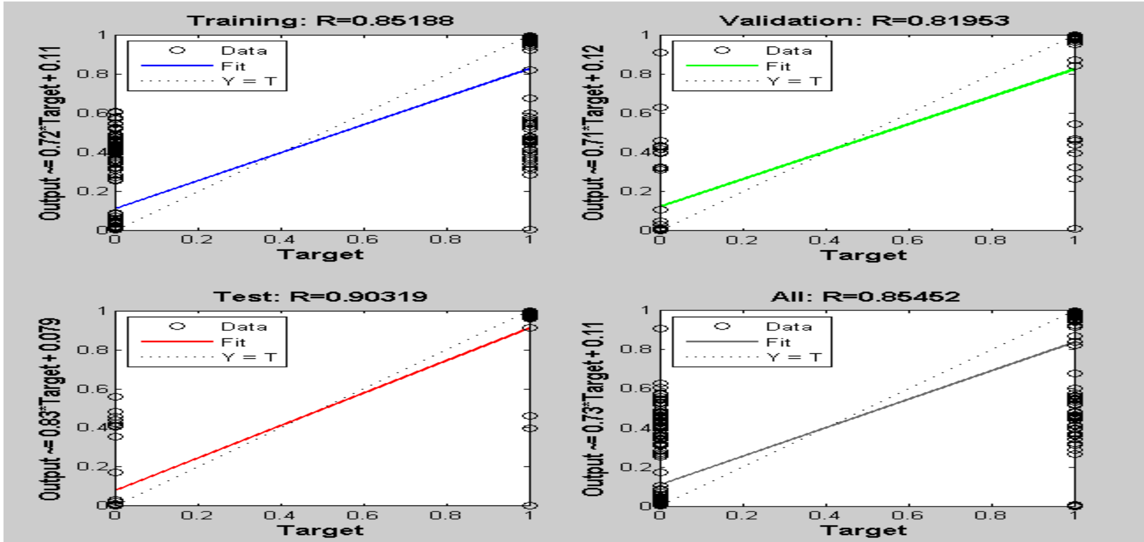
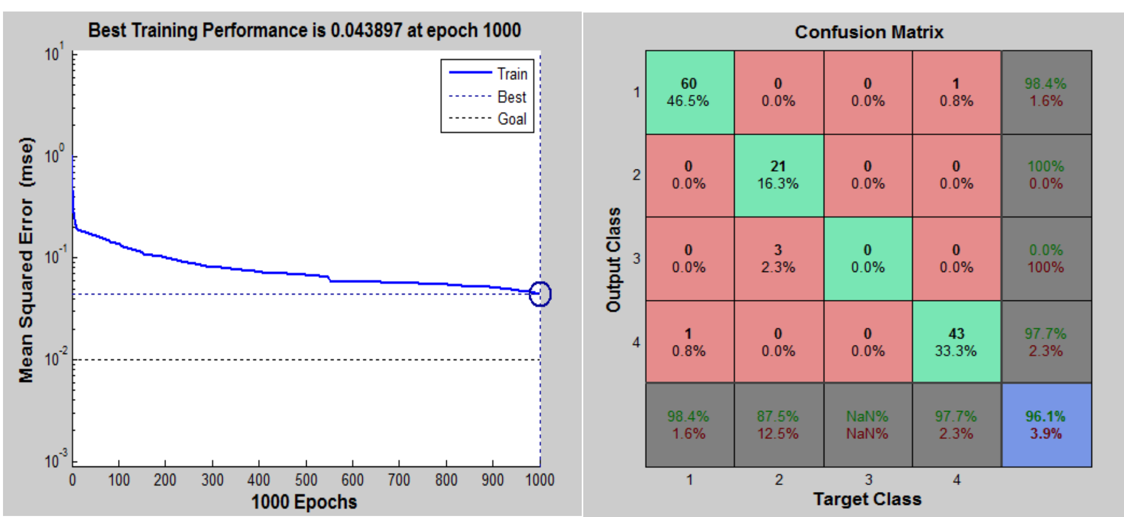


Figure 6.51: Regression plots for 4-10-4

6.5.4.3 Network 3 [4-12-4]

This network had a single hidden layer with 12 neurons. The target MSE was 0.01. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plots are shown in Figures 6.52-6.54 respectively. The goal achieved was 0.043897 at the end of 1000 iterations, while the correlation coefficient in this case has been found to be 0.87468 with an overall confusion matrix of 96.1%. The result obtained from the test dataset showed an accuracy of 97%.



(a) (b)

Figure 6.52: (a) Performance plot for 4-12-4; and **(b)** Confusion matrix for 4-12-4

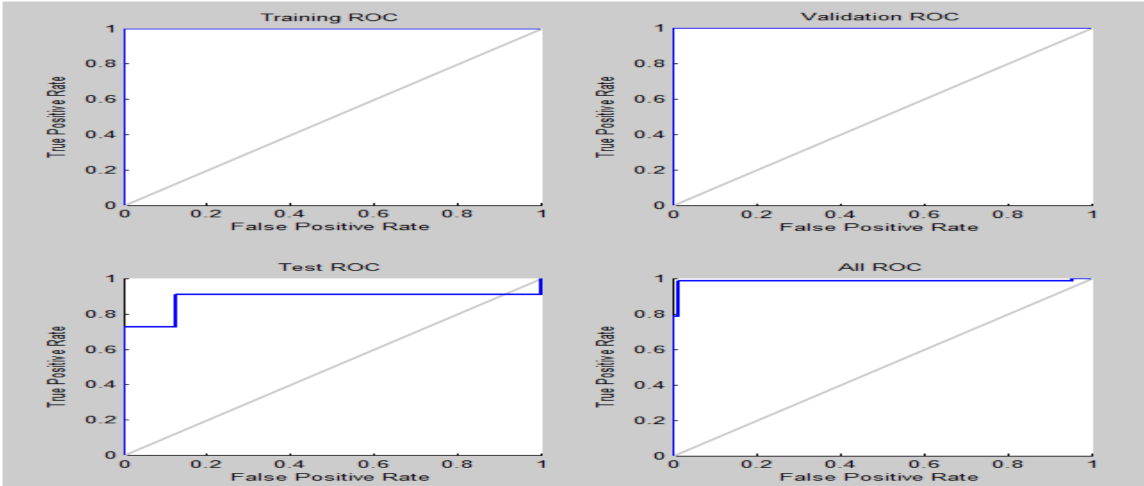


Figure 6.53: ROC plots for 4-12-4

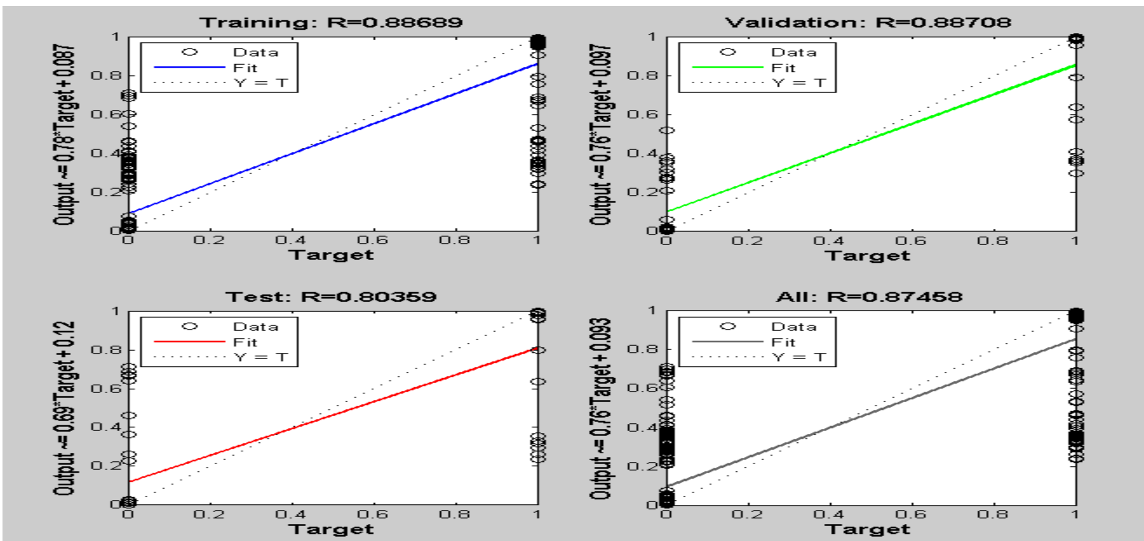


Figure 6.54: Regression plots for 4-12-4

6.5.4.4 Discussion of the Results

The result obtained shows that the best structure to use is 4-12-4. The final structure is shown in Figure 6.55. Tables 6.33-6.36 give a summary of the investigations carried out for the fault section identification for three phase faults.

A summary of the learning algorithm, activation function, number of hidden layer neurons, and confusion matrix are shown in Table 6.35. Table 6.36 gives the effect of increasing the number of hidden layers.

From these tables, it can be seen that the best confusion matrix was obtained using Resilient Propagation (RP) algorithm, while Levenberg-Marquardt (L-M) algorithm was second. However, the best coefficient of regression was obtained using L-M algorithm. Besides, Resilient Propagation (RP) took a longer training time. Hence, L-M algorithm was chosen for this task. Generally, the Mean Square Error (MSE) and confusion matrix did not improve with increase in the number of hidden layers and hidden layer neurons.

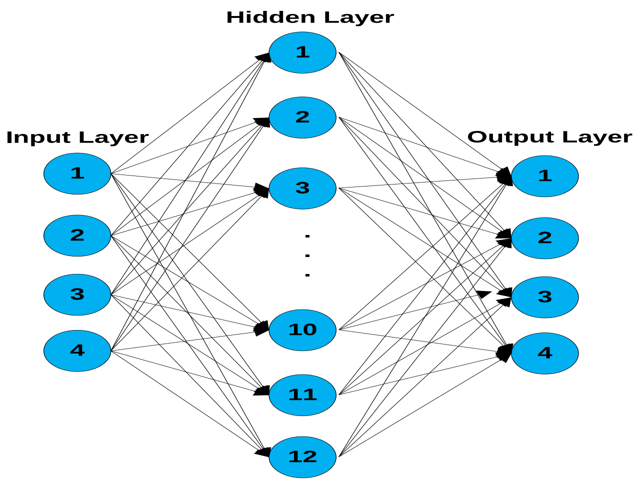


Figure 6.55: **Final ANN for 3Ph. fault section identification (4-12-4)**

Table 6.33: **Analysis of the Confusion Matrix Results for the Chosen Architecture**

Algorithm	Mean confusion matrix (%)	Min. confusion matrix (%)	Max. confusion matrix (%)
BR	89.9	88.4	92.2
LM	93.6	87.6	96.1
SCG	85.2	81.4	93.0
RP	98.6	96.9	99.2
OSS	87.8	81.4	93.8

Table 6.34: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.8790	0.8759	0.8811
LM	0.9671	0.9553	0.9766
SCG	0.8775	0.8608	0.8914
RP	0.8752	0.8632	0.8809
OSS	0.8688	0.8430	0.8851

Table 6.35: **Network Simulation Summary for Three Phase Fault Section Identification**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Tansig*	5	0.005	0.1	10	0.1510	85.3
MLP	LM	*Tansig/Tansig*	10	0.005	0.1	10	0.0768	86.0
MLP	LM	*Tansig/Tansig*	15	0.005	0.1	10	0.0685	92.2
MLP	LM	*Tansig/Tansig*	20	0.005	0.1	10	0.0736	79.8

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.36: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Confusion matrix (%)	Regression
1	5	LM	*TANSIG*	0.198	95.3	0.89040
1	10	LM	*TANSIG*	0.059	95.3	0.96878
2	5	LM	*TANSIG/TANSIG*	0.144	91.5	0.92158
2	10	LM	*TANSIG/TANSIG*	0.049	82.9	0.97362

6.6 Fault Location

This section reports on the results of the neural network based fault location algorithm for each of the various types of faults. This is presented in the following subsections.

Several factors have been considered while choosing the optimum architecture for this task.

Results are shown firstly for small neural networks, then for large ones with varying combination of hidden layer(s) and number of neurons per hidden layer. The performance of the trained neural network was based on their regression result, MSE, error histogram, and independent testing using a hold-out test dataset. The gradient and validation performance plots were also monitored during training.

6.6.1 Fault Location for Single Phase-Ground Faults

6.6.1.1 Network 1 [4-5-1]

The first network trained had a single hidden layer with 5 neurons. The target MSE was 0.001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. The structure is as shown in Figure 6.56.

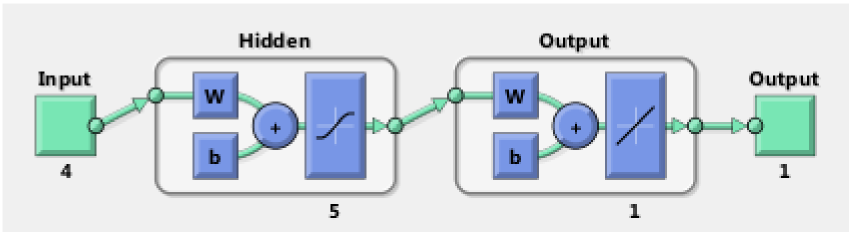
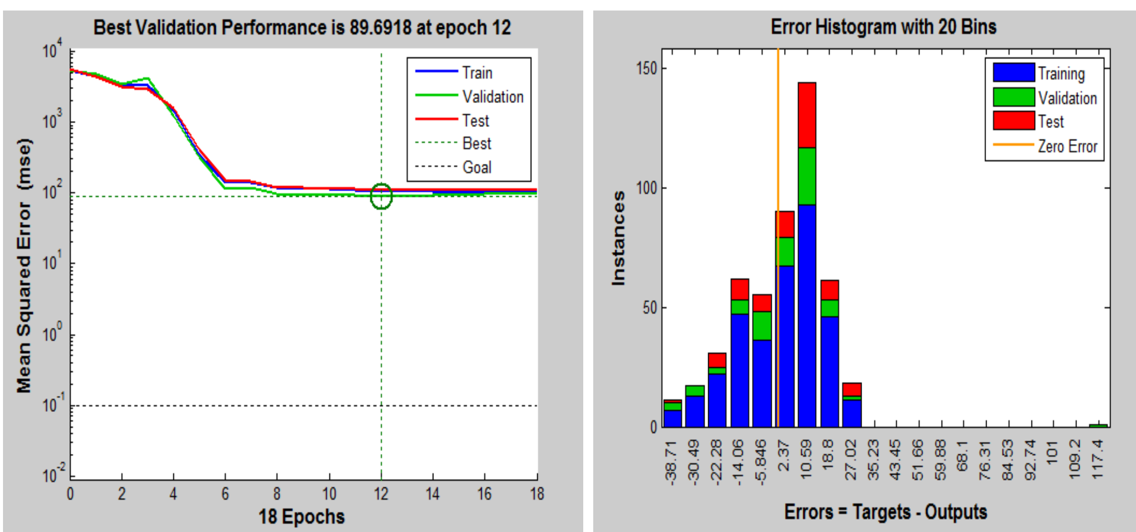


Figure 6.56: **MLP structure for 4-5-1**

A number of performance analyses were carried out on the network. The first of these was the performance plot. The performance plot in Figure 6.57(a) shows that the training went into local minima after about 30 epochs and stopped after 84 epochs.

The error histogram is as shown in Figure 6.57(b). The blue bars represent training data; the green bars represent the validation data, while the test data is represented by the red bars. Most errors fell between -30.49 and 27.02. Outliers were also found in the dataset. There was a validation point with error of 117.4. These outliers were also visible on the training, validation, and testing regression plot at -38.71. Figure 6.58 shows the training states.



(a) **(b)**

Figure 6.57: (a) **Performance plot for 4-5-1; and** **(b)** **Error histogram for 4-5-1**

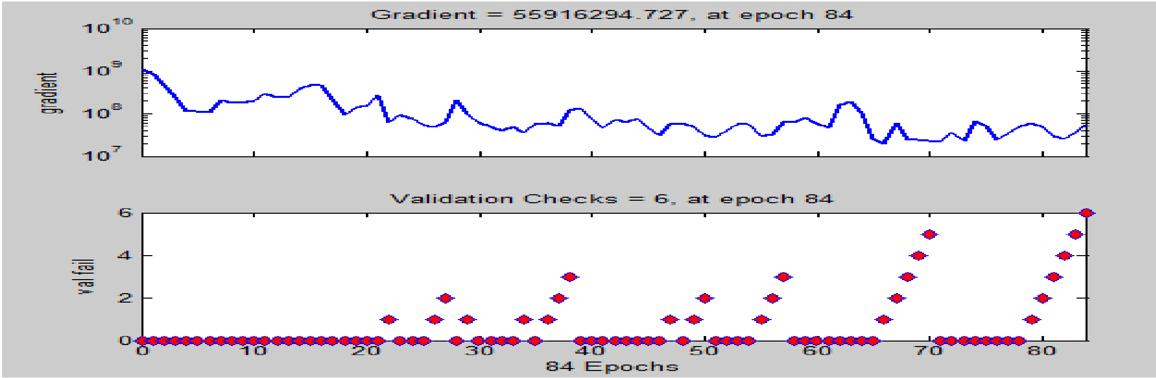


Figure 6.58: **Training state plots for 4-5-1**

Another performance analysis was by using regression plot. These plots are as shown in Figure 6.59. The dotted line in the figure indicates the ideal regression fit, while the coloured solid lines indicate the actual fit of the neural network. A plot of the best regression shows a correlation coefficient of 0.83914.

The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. It should be noted that this test dataset are untrained and are generated with parameters that are different from that used for the training dataset.

Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 19.19% and 8.85% respectively. The maximum error recorded was for fault at the beginning of the main feeder at 11.23 km, while the minimum was at L.832 at 54.32 km.

The gradient and validation performance plots were also monitored during training. It can be seen that there is a steady decrease in the gradient and also that the number of validation checks at the 84th epoch was 6. This indicates smooth and efficient training.

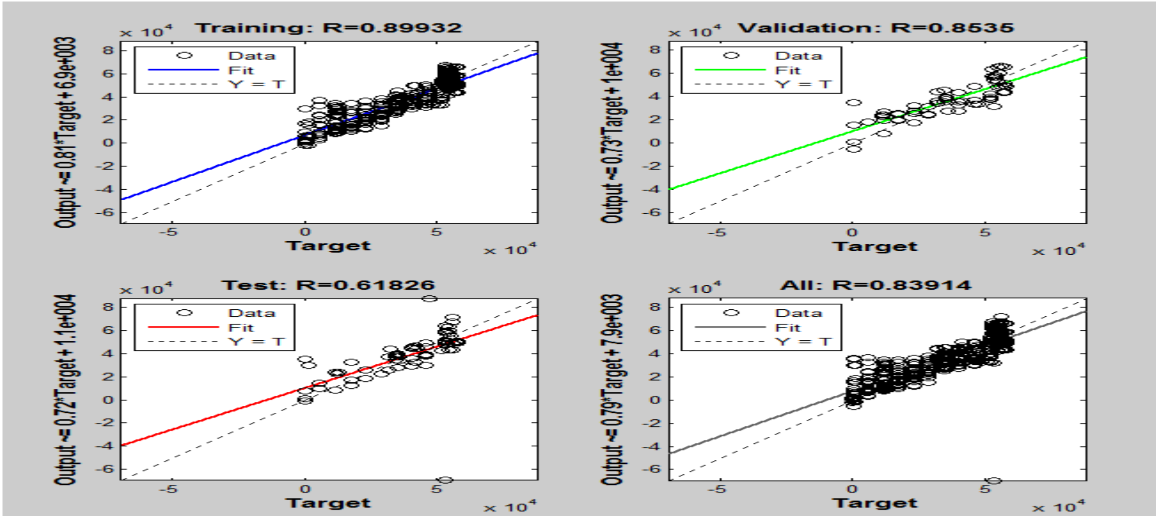


Figure 6.59: **Regression plots for 4-5-1**

6.6.1.2 Network 2 [4-5-1]

Retraining the 4-5-1 network with L-M algorithm stopped after 36 epochs. A number of performance analyses were carried out on the network. The first of these was the performance plot. The performance plot showed a steady decrease in MSE.

The second means of testing was the training regression plot. A plot of the best regression shows a correlation coefficient of 0.96042. Thirdly, the error histogram showed that most errors fell between -20 and 18. Outliers were also found in the dataset. There was a training point with an error of -40 and validation and testing points with errors of -30.

The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. It should be noted that this test dataset are untrained and are generated with parameters that are different from that used for the training dataset.

Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 10.87% and 6.36% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder at 11.23 km.

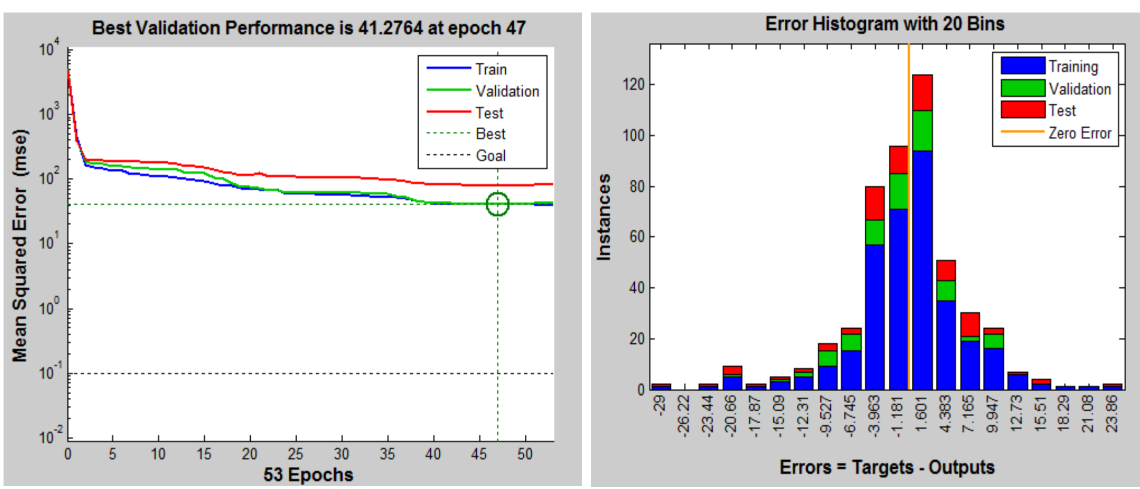
The gradient and validation performance plots were also monitored during training. It can be seen that there is a steady decrease in the gradient and also that the number of validation checks at the 34th epoch was 6. This indicates smooth and efficient training.

6.6.1.3 Network 3 [4-10-1]

Next was a network trained with 10 neurons in a single hidden layer. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 42 epochs

The performance plots are shown in Figures 6.60 and 6.61 respectively. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows a correlation coefficient of 0.96658. The error histogram showed most errors falling between -6.745 and 9.947. Outliers existed at training points with an error of -20.66 and 15.51. The errors in the validation points were at -12.31 and -15.09, while that of the testing points were at -29.0, -23.44, -20.66, 12.73, 15.51, 18.29, 21.08, and 23.86. Another validation method in the testing process was to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. It should be noted that this test dataset is untrained and is generated with parameters that are different from that used for the training dataset.

Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 9.43% and 5.84% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder at 11.23 km.



(a) **(b)**

Figure 6.60: (a) Performance plot for 4-10-1; and **(b)** Error histogram for 4-10-1

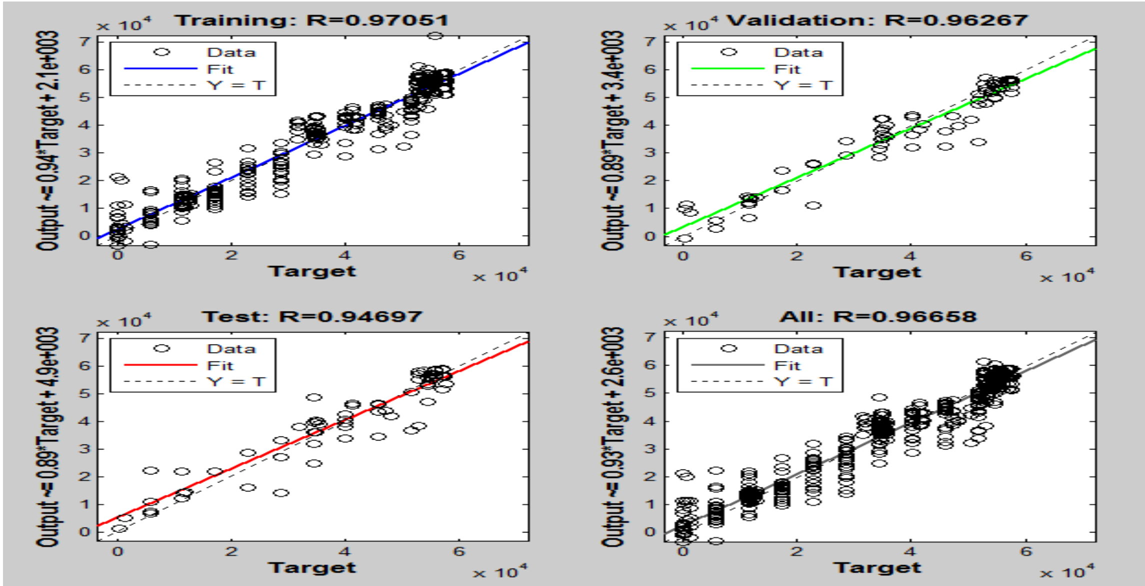


Figure 6.61: Regression plots for 4-10-1

6.6.1.4 Network 4 [4-20-1]

This network was trained with 20 neurons in a single hidden layer. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 18 epochs. The performance plots are shown in Figure 6.62. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.97446. The error histogram shows that most errors fell between -5576 and 3264. Outliers existed at training points with an error of -30, -20, -10 and 12. The errors in the validation points were at -30, -20, -10, 12 and 21, while that of the testing points was at -20, -10, 12, and 30.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 5.04% and 3.19% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

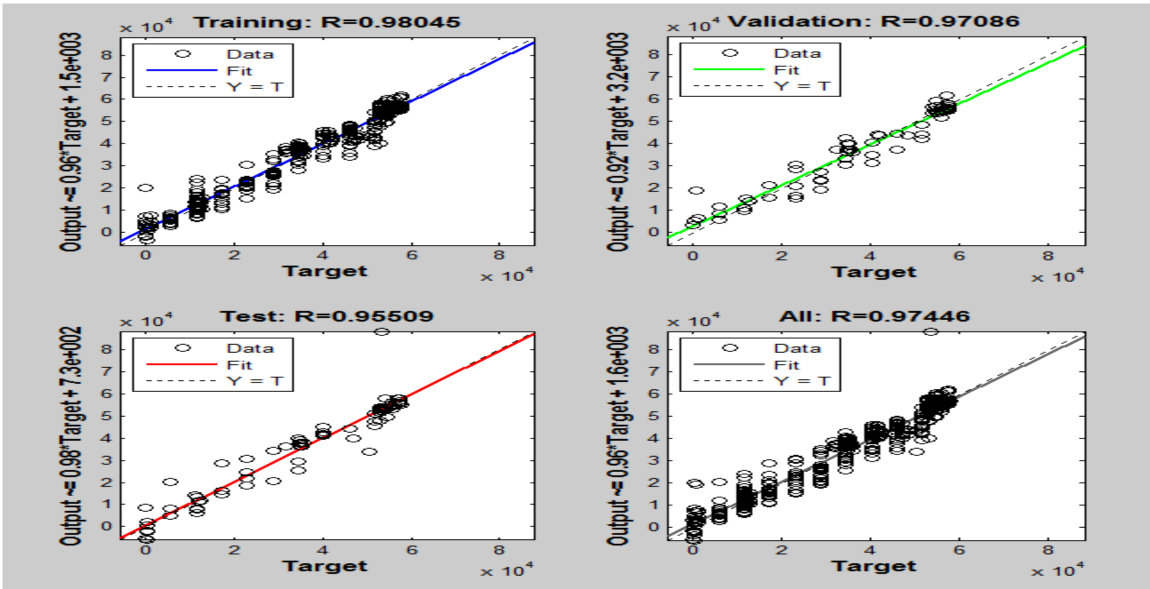


Figure 6.62: Regression plots for 4-20-1

6.6.1.5 Network 5 [4-30-1]

This network was trained with 30 neurons in a single hidden layer. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 48 epochs. The performance plots are shown in Figure 6.63. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.97885. The error histogram shows that most errors fell between -6615 and 3789. Outliers existed at training points with an error of -20 and 14. The errors in the validation points were at -20 and 14, while that of the testing points was at 14.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 9.43% and 5.84% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder at 11.23 km. The result obtained from the test dataset showed maximum and minimum errors of 0.97% and 0.22% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

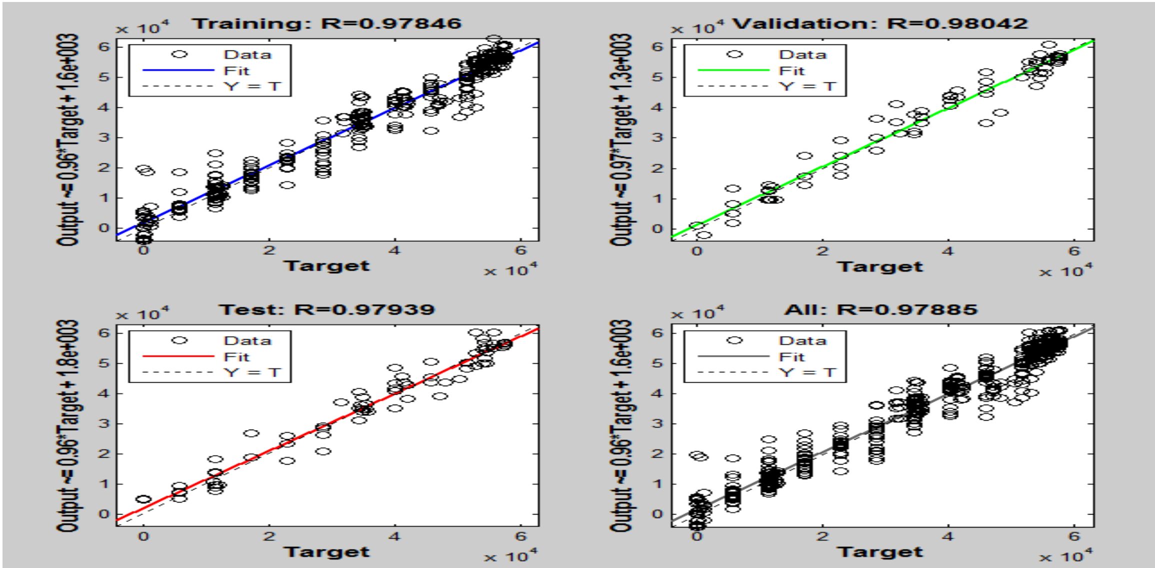


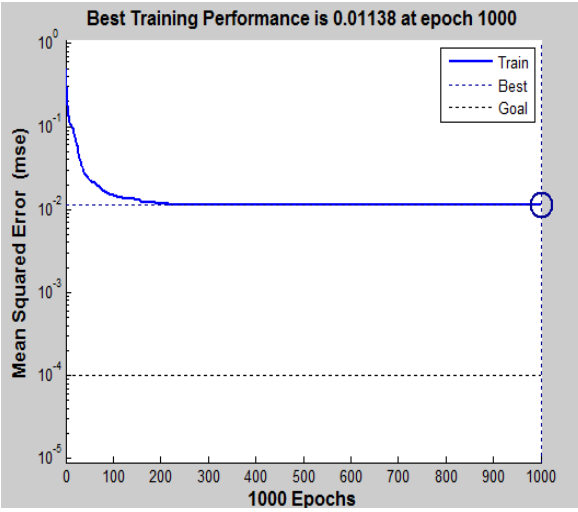
Figure 6.63: **Regression plots for 4-30-1**

6.6.1.6 Network 6 [4-5-5-1]

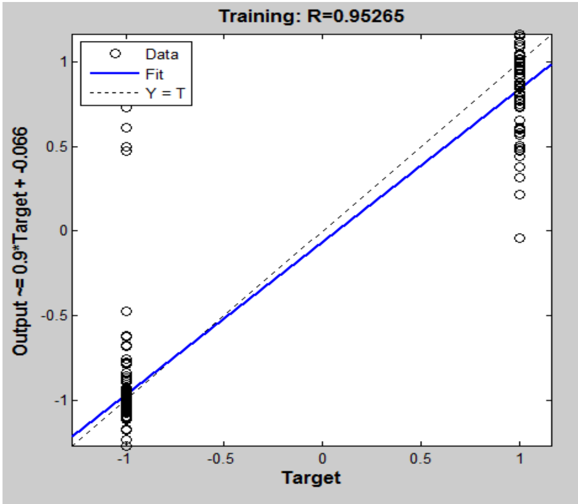
This network was trained with 5 neurons each in two hidden layers. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 598 epochs.

The performance plots are shown in Figure 6.64. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.95265. The error histogram is as showed with most errors falling between -0.3 and 0.001. Outliers existed at training points with an error of -10 and 13. The errors in the validation points were at -0.09, and 0.01.1, while that of the testing points was at -20 and 13.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 11.97% and 3.13% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.



(a)



(b)

Figure 6.64: (a) Performance plot for 4-5-5-1; and **(b)** Regression plot for 4-5-5-1

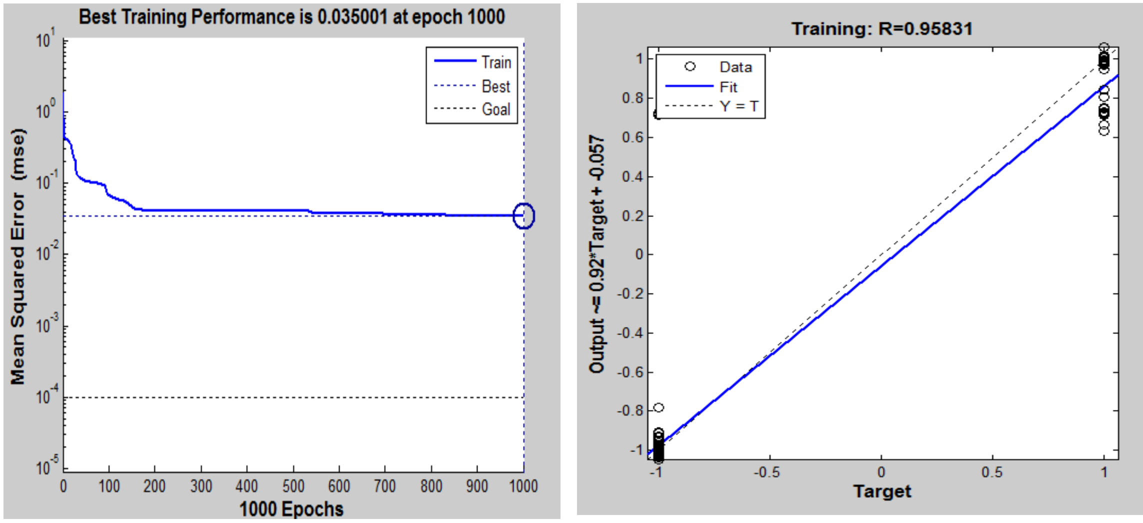
6.6.1.7 Network 7 [4-10-10-1]

This network was trained with 10 neurons each in two hidden layers. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

The performance plots are shown in Figure 6.65. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.95831. The error histogram showed majority of the errors were at -0.04687.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing.

The result obtained from the test dataset showed maximum and minimum errors of 4.91% and 1.13% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

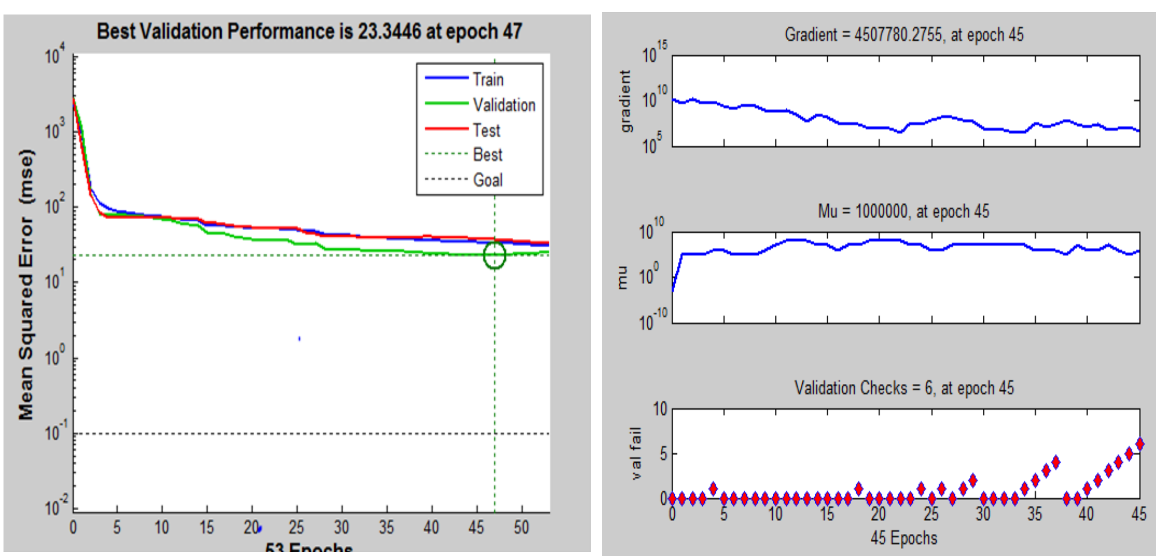


(a) (b)

Figure 6.65: (a) Performance plot for 4-10-10-4; and **(b)** Regression plot for 4-10-10-1

6.6.1.8 Network 8 [4-21-1]

This network was trained with 21 neurons in a single hidden layer. The training algorithm was L-M algorithm with early-stopping. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 45 epochs. The performance plots are shown in Figures 6.66 and 6.67 respectively. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.97665. The error histogram showed that most errors fell at 1.249.



(a) (b)

Figure 6.66: (a) Performance plot for 4-21-4, and **(b)** Training states for 4-21-4

Training, validation, and test errors exist at -20. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. The result obtained from the test dataset showed maximum and minimum errors of 0.71% and 0.26% respectively.

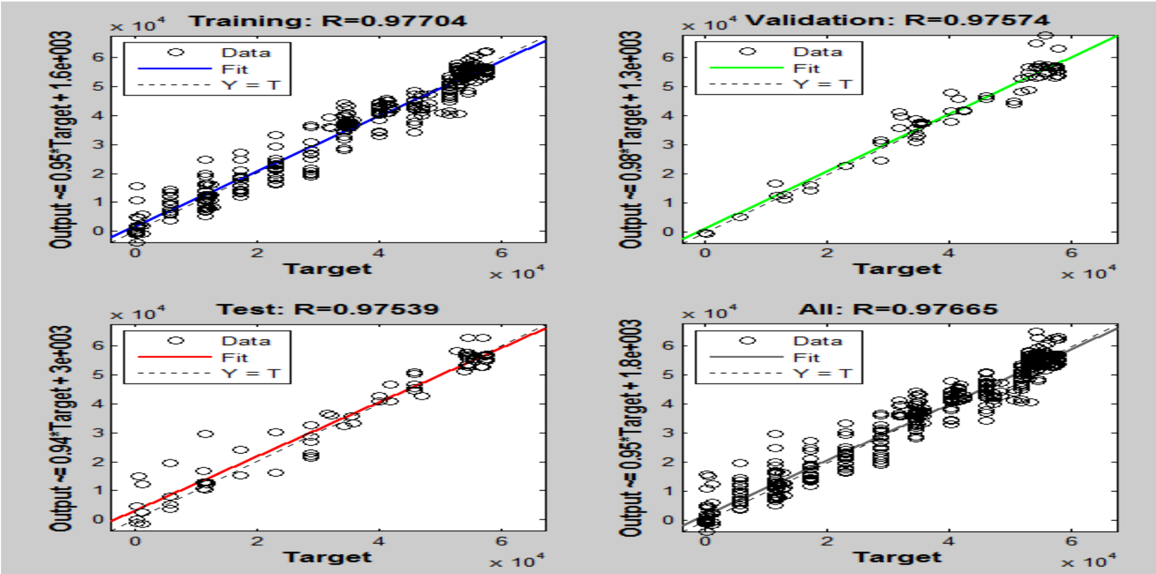


Figure 6.67: **Regression plots for 4-21-1**

6.6.1.9 Network 9 [4-21-1]

Retraining Network 8 with BR algorithm showed further decrease in the percentage error. Also, a plot of the best regression shows an overall correlation coefficient of 0.99062. The error histogram showed that most errors were between -3.613 and 3.967.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 1.07% and 0.13% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.1.10 Discussion of the Results

The result obtained shows that the best structure to use is 4-21-1. The final structure is shown in Figure 6.68.

Table 6.37 presents the analysis of the chosen architecture for different learning algorithms. Tables 6.38-6.39 give a summary of the effect of increasing the number of hidden layer neurons and number of hidden layer respectively.

From these tables, it can be seen that best coefficient of regression was obtained using Bayesian Regularization (BR) algorithm; next to it was the Levenberg-Marquardt (L-M) algorithm. However, BR algorithm took a longer time in training. Hence, L-M algorithm was used for chosen for this reason and also because of its good performance when tested. Results also show that the Mean Square Error (MSE) and confusion matrix improved with increase in the number of hidden layers and hidden layer neurons. The guideline used in the thesis was to select the smallest neural network with good performance. This is because bigger networks easily over-learn the training dataset and would fail to generalise when presented with untrained dataset.

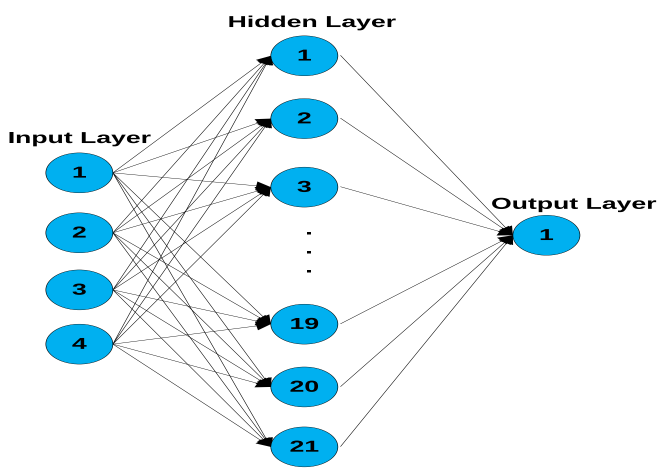


Figure 6.68: **Final ANN for 1Ph. fault location (4-21-1)**

Table 6.37: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.9905	0.9901	0.9913
LM	0.9686	0.9447	0.9759
SCG	0.9018	0.8838	0.9257
RP	0.8997	0.8871	0.9278
OSS	0.8951	0.8764	0.9114

Table 6.38: **Effect of increasing the number of hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Purelin*	5	0.005	0.1	10	0.0397	0.95228
MLP	LM	*Tansig/Purelin*	10	0.005	0.1	10	0.0209	0.97521
MLP	LM	*Tansig/Purelin*	15	0.005	0.1	10	0.0118	0.98610
MLP	LM	*Tansig/Purelin*	20	0.005	0.1	10	0.0068	0.99195
MLP	LM	*Tansig/Purelin*	25	0.005	0.1	10	0.0054	0.99371
MLP	LM	*Tansig/Purelin*	30	0.005	0.1	10	0.0039	0.99533
MLP	LM	*Tansig/Purelin*	35	0.005	0.1	10	0.0026	0.99696
MLP	LM	*Tansig/Purelin*	40	0.005	0.1	10	0.0024	0.99713
MLP	LM	*Tansig/Purelin*	45	0.005	0.1	10	0.0027	0.99679

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.39: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Regression
1	5	LM	*TANSIG*	0.03970	0.95228
1	10	LM	*TANSIG*	0.02090	0.97521
1	15	LM	*TANSIG*	0.01180	0.98610
2	5	LM	*TANSIG/TANSIG*	0.01620	0.98077
2	10	LM	*TANSIG/TANSIG*	0.00323	0.99621
2	15	LM	*TANSIG/TANSIG*	0.00046	0.99946

6.6.2 Fault Location for Two Phase Faults

This section reports on the results of the neural network based fault location algorithm for two phase faults. Several factors have been considered while choosing the optimum architecture for this task. Results are shown firstly for small neural networks, then for large ones with varying combination of hidden layer(s) and number of neurons per hidden layer. The performance of the trained neural network was based on their regression result, MSE, error histogram, and independent testing using a hold-out test dataset. The gradient and validation performance plots were also monitored during training.

6.6.2.1 Network 1 [4-5-1]

The first network trained had a single hidden layer with 5 neurons. The learning rate and momentum were 0.7 and 0.8 respectively. The target MSE was 0.001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

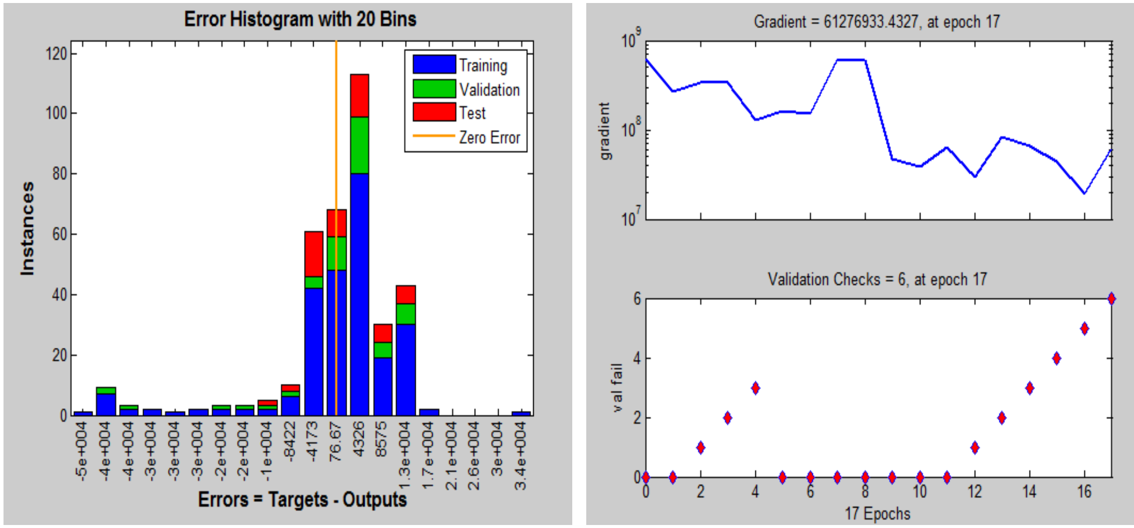
A number of performance analyses were carried out on the network. The first of these was the performance plot. The performance plot shows that the training went into local minima after about 30 epochs. Training stopped after 84 epochs.

The second means of testing was the error histogram is as shown in Figure 6.69(a). The blue bars represent training data; the green bars represent the validation data, while the test data is represented by the red bars. Most errors fell between -4.173 and 1.3e+001. Outliers were also found in the dataset. There were training points with error between -5e+001 and 8.422, and validation points with errors of -4e+001 and -8.422. These outliers were also visible on the testing regression plot at -1e+001, -8.422, 1.7e+001, and 3.7e+001.

The third means of validation was the training regression plot. These plots are as shown in Figure 6.70. The dotted line in the figure indicates the ideal regression fit, while the coloured solid lines indicate the actual fit of the neural network. A plot of the best regression shows a correlation coefficient of 0.82395.

The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 100 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. It should be noted that this test dataset is untrained and is generated with parameters that are different from that used for the training dataset. (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 11.37% and 1.59% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

The gradient and validation performance plots were also monitored during training. It can be seen that there is a steady decrease in the gradient and also that the number of validation checks at the 84th epoch was 6.



(a) **(b)**

Figure 6.69: (a) Error histogram for 4-5-1; and **(b)** Training state for 4-5-1

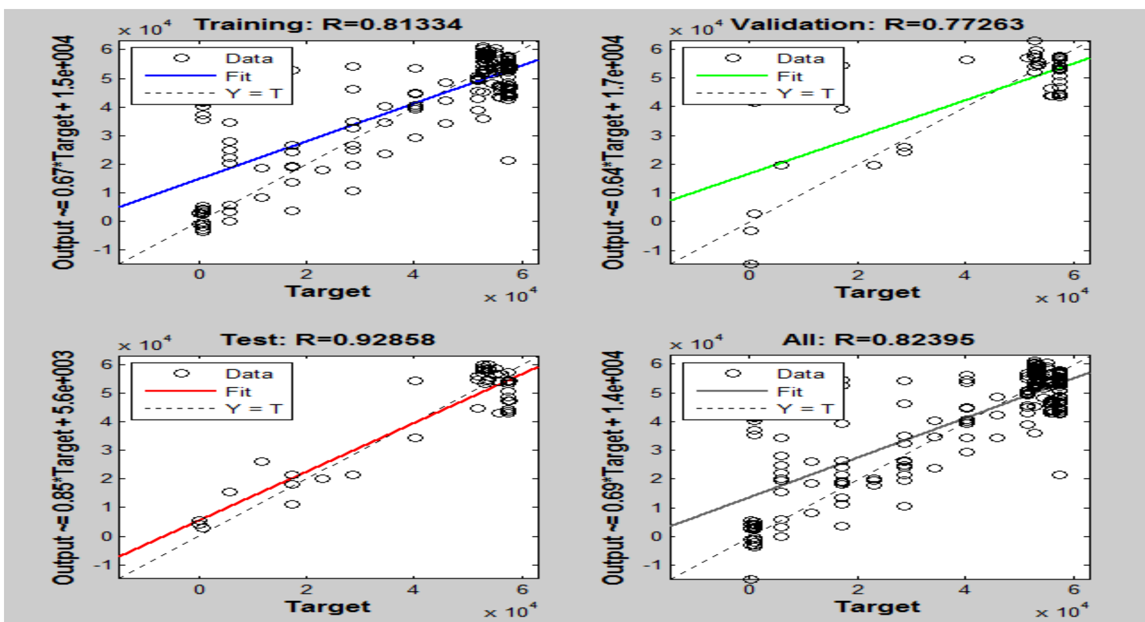


Figure 6.70: Regression plots for 4-5-1

6.6.2.2 Network 2 [4-5-1]

Retraining with L-M algorithm stopped after 48 epochs in 7 seconds. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.96009. The error histogram showed most errors falling between -4.394 and 3.542. Outliers existed at training points with an error of -4e+001, -2e+001, 1e+001, 7.510, 1.1e+001, 1.5e+001, 1.9e+001, and 3.1e+001.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was afterwards used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 9.61% and 0.85% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.2.3 Network 3 [4-10-1]

This network was trained with 10 neurons in a single hidden layer. The training algorithm was BR algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 241 epochs in 17 seconds.

The errors in the training points are as shown in Figure 6.71(a). Also, a plot of the best regression shows an overall correlation coefficient of 0.99174 as shown in Figure 6.71(b). A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 7.20% and 0.75% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

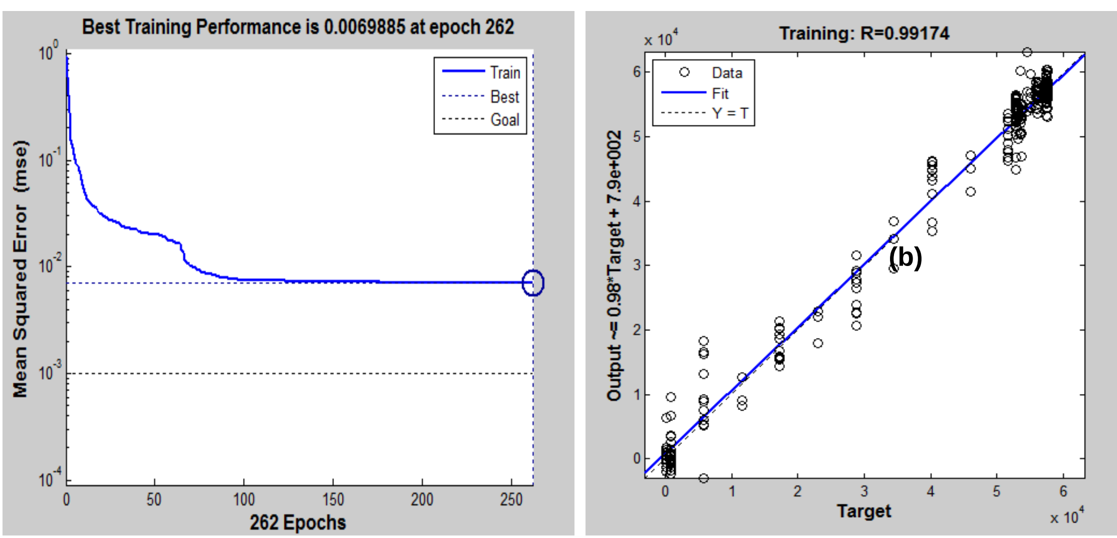


Figure 6.71: (a) Error histogram for 4-10-1; and **(b)** Regression plot of 4-10-1

6.6.2.4 Network 4 [4-20-1]

This network was trained with 20 neurons in a single hidden layer. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 22 epochs in 2 seconds.

The performance plots are shown in Figures 6.72 and 6.73 respectively. The performance plot showed further decrease in the MSE with convergence 0.00099589 at 499 iterations. Also, a plot of the best regression shows an overall correlation coefficient of 0.96424. The errors in the training, validation, and test points are as shown in Figure 6.72.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 5.65% and 0.50% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

Similarly, this network was trained with BR algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 655 epochs.

The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.99925. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 4.65% and 0.25% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

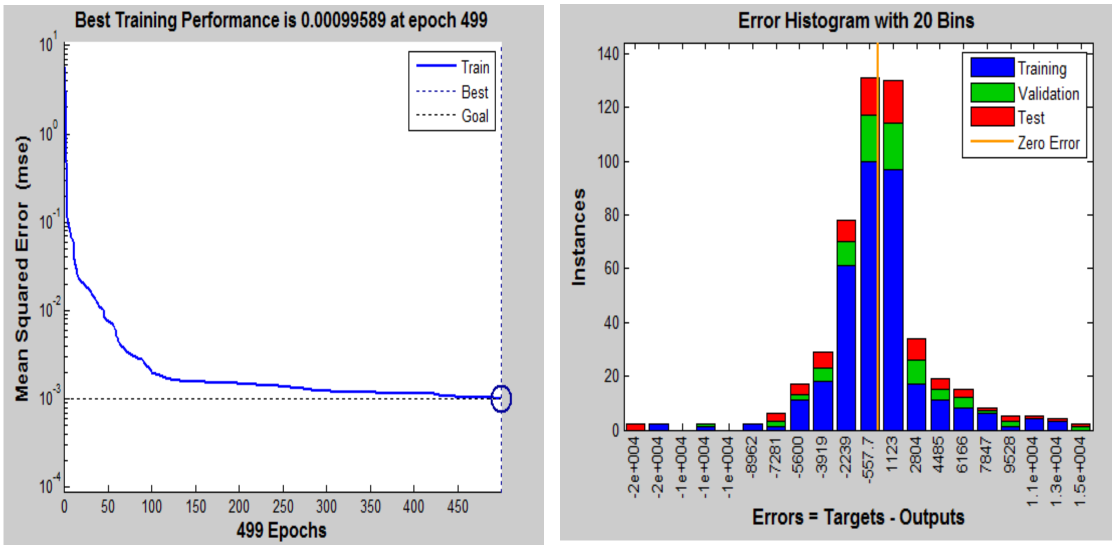


Figure 6.72: (a) Performance plot for 4-10-1; and **(b)** Error histogram for 4-10-1

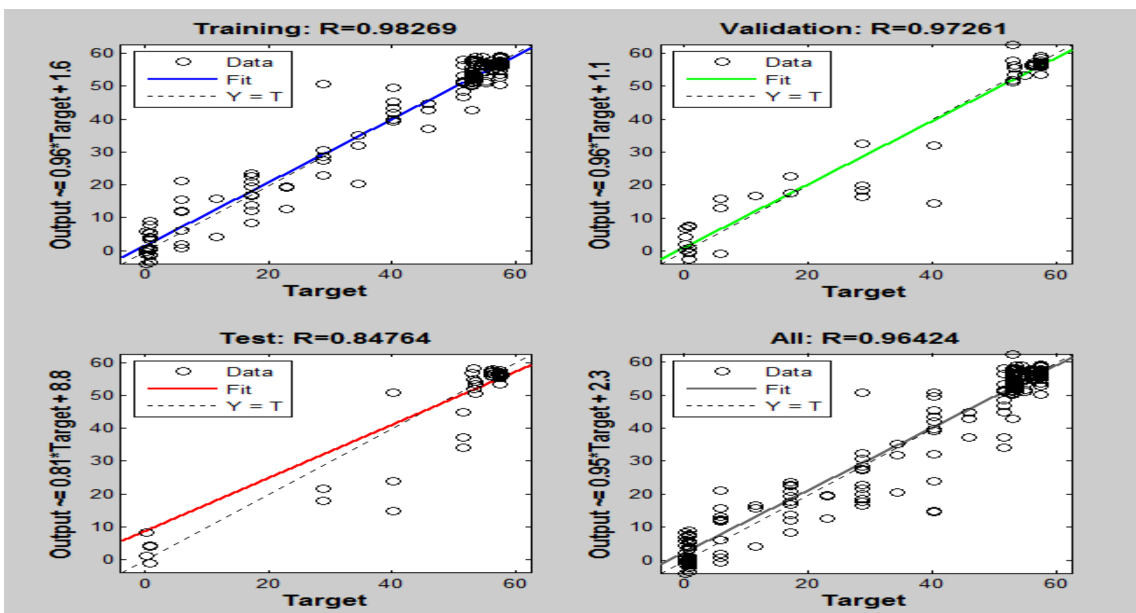


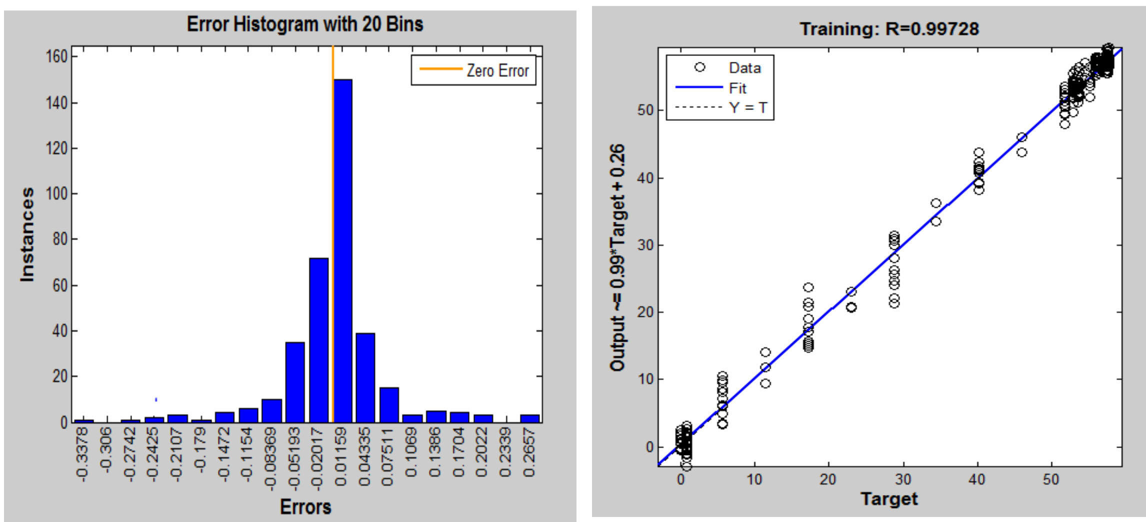
Figure 6.73: Regression plots for 4-10-1

6.6.2.5 Network 5 [4-5-5-1]

This network was trained with 5 neurons in two hidden layers. The training algorithm was BR algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

The errors in the training points are as shown in Figure 6.74(a). Also, a plot of the best regression analysis shows an overall correlation coefficient of 0.99728 as shown in Figure 6.74(b).

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 2.85% and 0.20% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.



(a) (b)

Figure 6.74: (a) Error Histogram for 4-5-5-1; and **(b)** Regression plot of 4-5-5-1

Retraining this network with L-M algorithm resulted to a performance goal of 0.0048 in 257 iterations after 10 seconds. Also, a plot of the best regression shows an overall correlation coefficient of 0.99623.

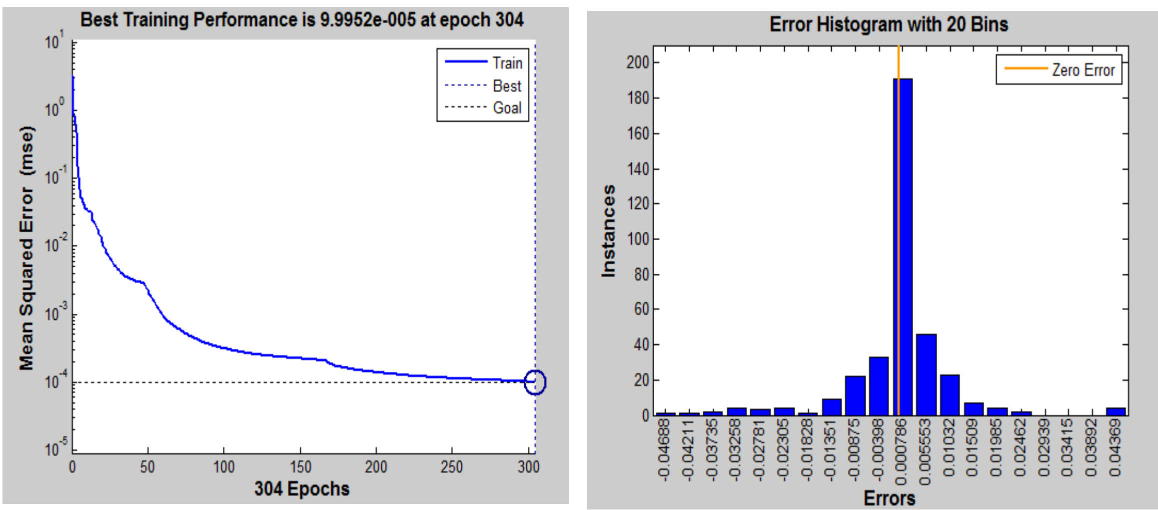
A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 3.55% and 0.80% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.2.6 Network 6 [4-15-15-1]

This network was trained with 15 neurons in two hidden layers respectively. The training algorithm was L-M algorithm. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

The performance goal of 0.0001 was achieved in 304 iterations. The errors in the training points are as shown in Figure 6.75(a). A plot of the best regression shows an overall correlation coefficient of 0.99989.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.65% and 0.07% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.



(a) (b)

Figure 6.75: (a) Performance plot for 4-15-15-1; and (b) Error histogram for 4-15-15-1

BR algorithm was also used to retrain the 4-15-15-1 network. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

A plot of the best regression shows an overall correlation coefficient of 0.99996.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 1.20% and 0.7% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.2.7 Discussion of the Results

The result obtained shows that the best structure to use is 4-15-15-1. The final structure is shown in Figure 6.76. Table 6.40 gives the regression analysis of the various training algorithm used. Tables 6.41-6.42 give a summary of the effect of increasing the number of hidden layer neurons and number of hidden layer respectively.

From these tables, it can be seen that the coefficient of regression obtained using Bayesian Regularization (BR) algorithm was the same with Levenberg-Marquardt (L-M) algorithm. However, BR algorithm took a longer time in training. Hence, L-M algorithm was used for chosen. Results also show that the Mean Square Error (MSE) and confusion matrix improved with increase in the number of hidden layers and hidden layer neurons. However, the guideline used in the thesis was to select the smallest neural network with good performance.

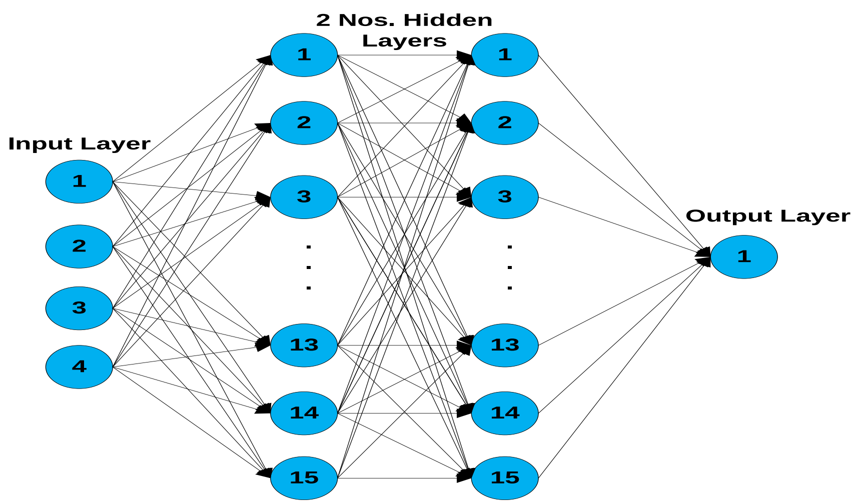


Figure 6.76: **Final ANN for 2 Ph. fault location (4-15-15-1)**

Table 6.40: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.9999	0.9999	0.9999
LM	0.9999	0.9999	0.9999
SCG	0.9967	0.9945	0.9977
RP	0.9927	0.9881	0.9966
OSS	0.9949	0.9920	0.9964

Table 6.41: **Effect of increasing the hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Purelin*	5	0.005	0.1	10	0.033800	0.96379
MLP	LM	*Tansig/Purelin*	10	0.005	0.1	10	0.007400	0.99219
MLP	LM	*Tansig/Purelin*	15	0.005	0.1	10	0.001800	0.99807
MLP	LM	*Tansig/Purelin*	20	0.005	0.1	10	0.001210	0.99878
MLP	LM	*Tansig/Purelin*	25	0.005	0.1	10	0.000310	0.99967
MLP	LM	*Tansig/Purelin*	30	0.005	0.1	10	0.000180	0.99981
MLP	LM	*Tansig/Purelin*	35	0.005	0.1	10	0.000091	0.99990
MLP	LM	*Tansig/Purelin*	40	0.005	0.1	10	0.000048	0.99995
MLP	LM	*Tansig/Purelin*	45	0.005	0.1	10	0.000057	0.99994

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.42: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Regression
1	5	LM	*TANSIG*	0.033800	0.96379
1	10	LM	*TANSIG*	0.007400	0.99219
1	15	LM	*TANSIG*	0.011800	0.99807
2	5	LM	*TANSIG/TANSIG*	0.004090	0.99569
2	10	LM	*TANSIG/TANSIG*	0.000120	0.99987
2	15	LM	*TANSIG/TANSIG*	0.000036	0.99996

6.6.3 Fault Location for Two Phase-Ground Faults

This section reports on the results of the neural network based fault location algorithm for two phase-to-ground faults.

Results are shown firstly for small neural networks, then for large ones with varying combination of hidden layer(s), and the number of neurons per hidden layer. Like in preceding subsections, the performance criteria used were based on the ANN regression result, MSE, error histogram, and independent testing using a hold-out test dataset. The gradient and validation performance plots were also monitored during training.

6.6.3.1 Network 1 [4-5-1]

The first network trained had a single hidden layer with 5 neurons. The learning rate and momentum were 0.7 and 0.8 respectively. The target MSE was 0.001. The training algorithm used was Bayesian Regularization (BR) algorithm. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

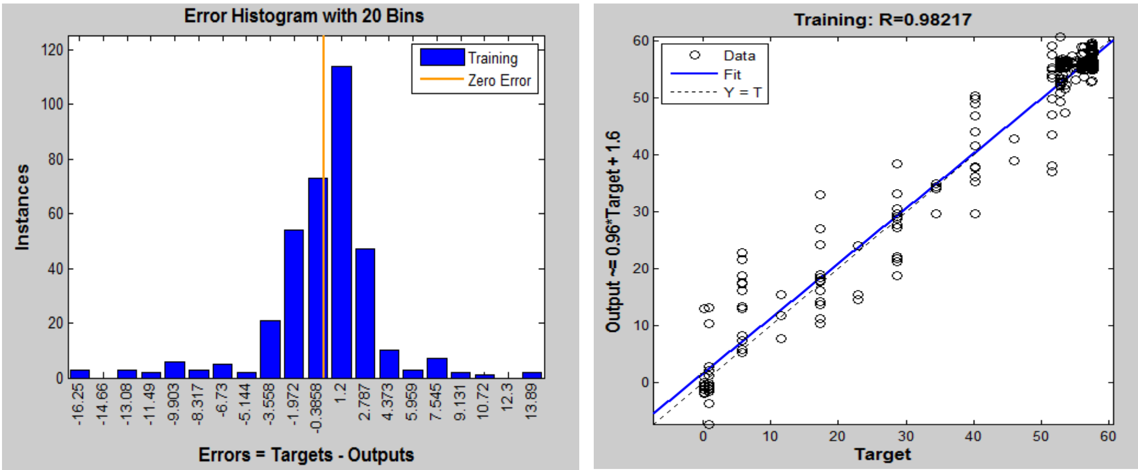
A number of performance analyses were carried out on the network. The first of these was the performance plot. The maximum *mu* was reached in 207 iterations.

The second means of testing was the error histogram as shown in Figure 6.77(a). The blue bars represent training data and the associated errors at various training points. The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network.

Thirdly, the training regression plot was analysed. This plot is as shown in Figure 6.77(b). The dotted line in the figure indicates the ideal regression fit, while the coloured solid lines indicate the actual fit of the neural network. It can be seen that these lines track each other very closely which is an indication of very good performance by the neural network. A plot of the best regression shows a correlation coefficient of 0.98217.

A total of 100 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. It should be noted that this test dataset are untrained and are generated with parameters that are different from that used for the training dataset.

Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 7.15% and 0.43% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.



(a) (b)

Figure 6.77: (a) Error histogram for 4-5-1; and **(b)** Regression plot for 4-5-1

6.6.3.2 Network 2 [4-5-1]

Retraining the 4-5-1 network using L-M algorithm showed a further decrease in the MSE. Training stopped after 12 epochs. Also, a plot of the best regression shows an overall correlation coefficient of 0.96391. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. The result obtained from the test dataset showed maximum and minimum errors of 5.03% and 0.21% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.3.3 Network 3 [4-10-1]

This network was trained with 10 neurons in a single hidden layer. The training algorithm was SCG. The target MSE was 0.001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 46 epochs.

The performance plot showed further decrease in the MSE. The errors in the training, validation, and test points are as shown in Figure 6.78. Also, a plot of the best regression as shown in Figure 6.79, gave an overall correlation coefficient of 0.93222.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 4.64% and 0.25% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

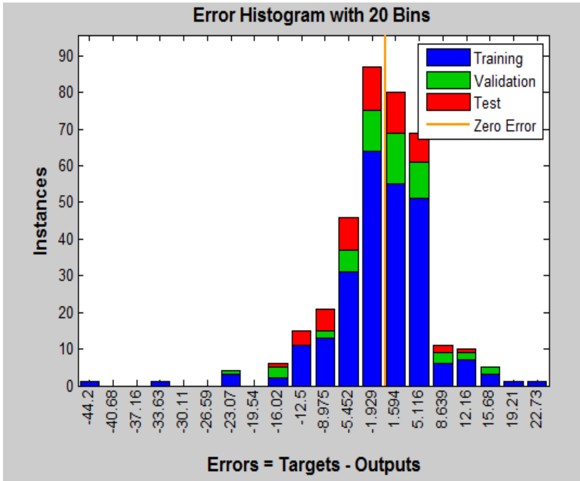


Figure 6.78: **Error histogram for 4-10-1**

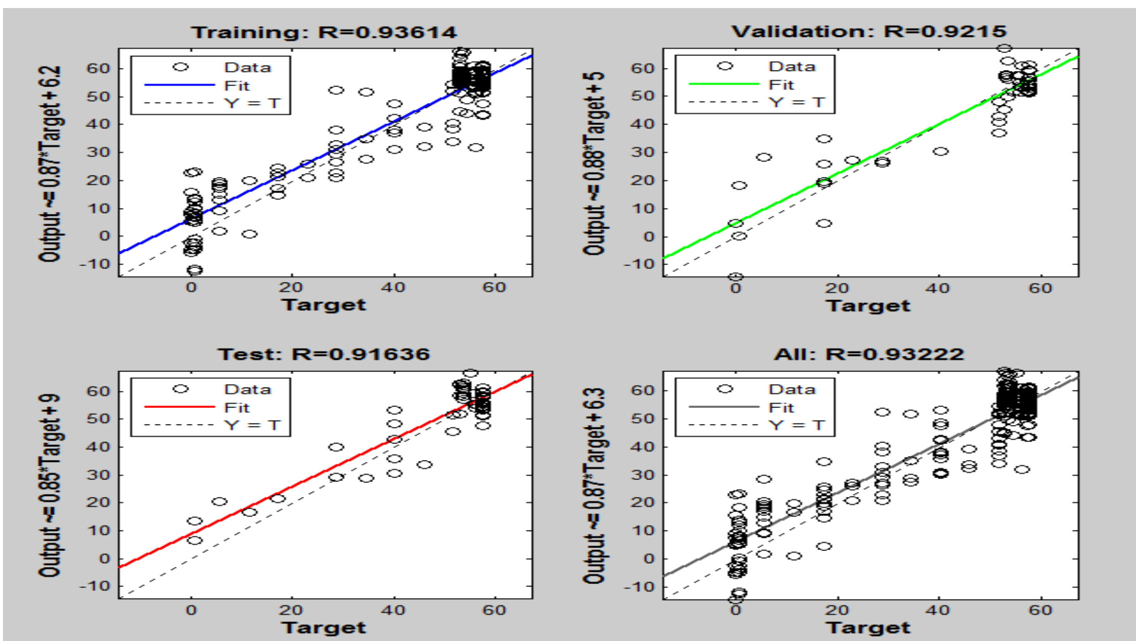


Figure 6.79: **Regression plots for 4-10-1**

6.6.3.4 Network 4 [4-5-1]

Retraining with L-M algorithm stopped after 39 epochs. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.98137.

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 3.20% and 0.15% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

Similarly, retraining stopped after 164 epochs. A plot of the best regression shows an overall correlation coefficient of 0.99396. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. The result obtained from the test dataset showed maximum and minimum errors of 2.10% and 0.85% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

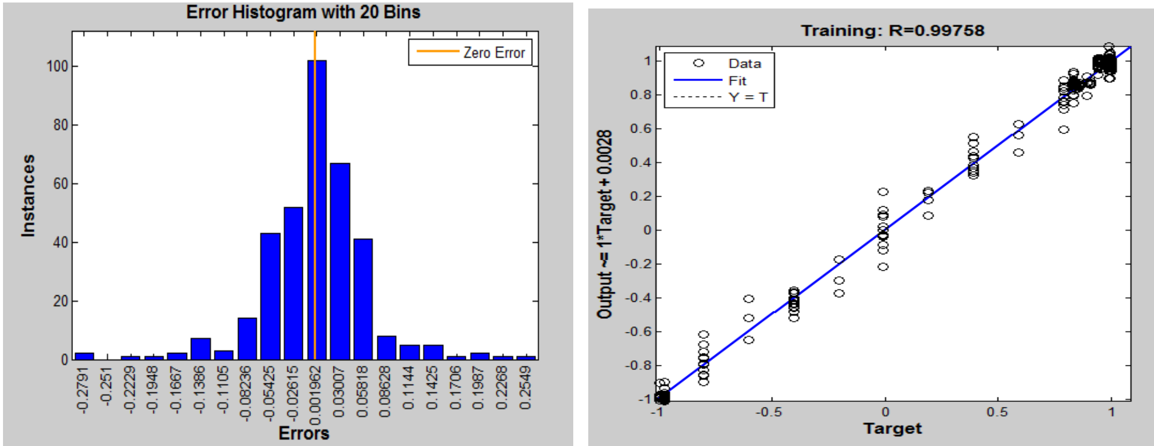
6.6.3.5 Network 5 [4-5-5-1]

This network was trained with 5 neurons in two hidden layers. The training algorithm was BR algorithm. The target MSE was 0.0001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 238 epochs.

The performance plots are shown in Figures 6.80. A plot of the best regression shows an overall correlation coefficient of 0.99758. The errors at the training points are as shown in Figure 6.80(a).

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 1.50% and 0.45% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

The 4-5-5-1 network described above was also retrained using L-M algorithm. Training stopped after 333 epochs. The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.99633. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 1.79% and 0.25% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.



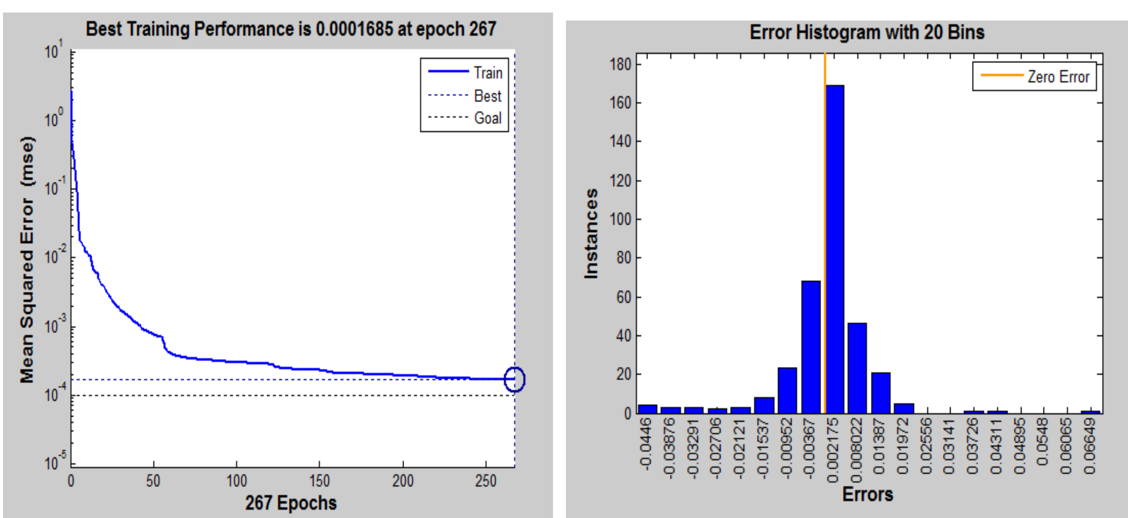
(a) (b)

Figure 6.80: (a) Error histogram for 4-5-5-1; and **(b)** Regression plot for 4-5-5-1

6.6.3.6 Network 6 [4-10-10-1]

This network was trained with 10 neurons in two hidden layers respectively. The training algorithm was L-M algorithm. The target MSE was 0.0001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 267 epochs.

The performance plots are shown in Figures 6.81 and 6.82 respectively. The MSE achieved was 0.0001685. Also, a plot of the best regression shows an overall correlation coefficient of 0.99982. The errors at the training points are as shown in Figure 6.81(b).



(a) (b)

Figure 6.81: (a) Performance plot for 4-10-10-1; and **(b)** Error histogram for 4-10-10-1

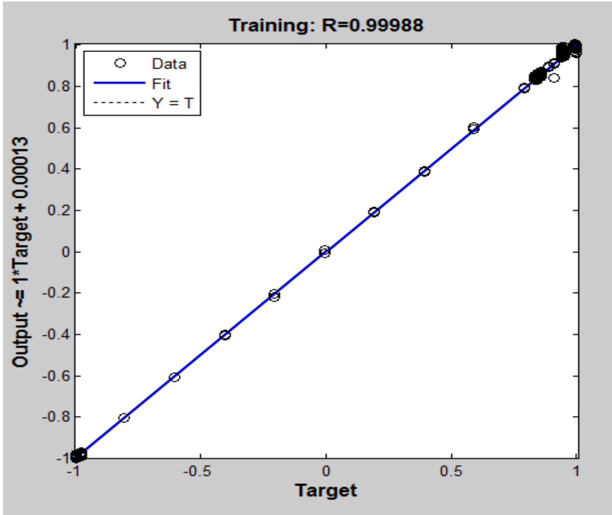


Figure 6.82: **Regression plots for 4-10-10-1**

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.55% and 0.10% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

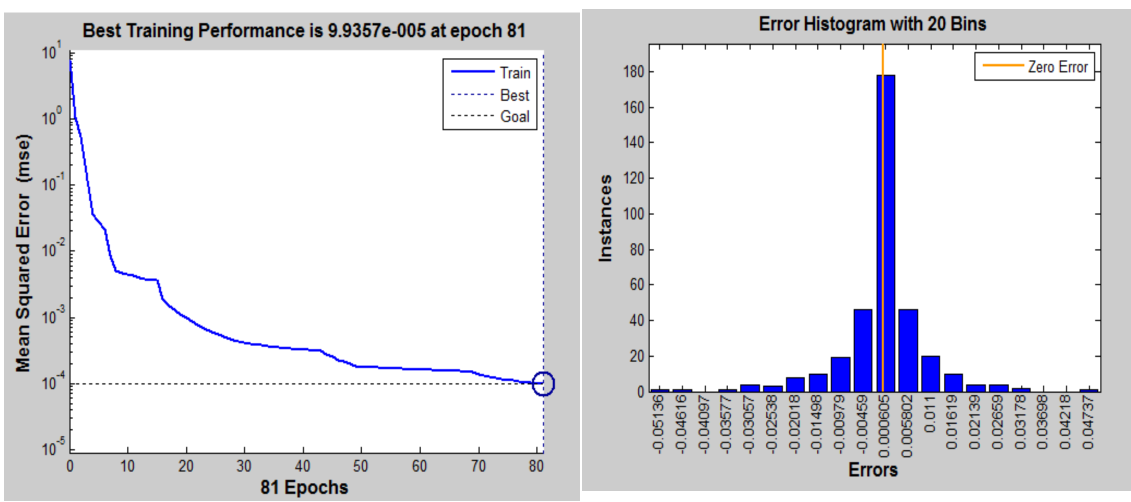
The 4-10-10-1 network described above was also retrained using BR algorithm. Training stopped after 291 epochs. The plot of the best regression shows an overall correlation coefficient of 0.99989. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.80% and 0.15% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

6.6.3.7 Network 7 [4-15-15-1]

This network was trained with 15 neurons in two hidden layers. The training algorithm was L-M algorithm. The target MSE was 0.0001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 81 epochs. The performance plots are shown in Figures 6.83 and 6.84 respectively. An MSE of 0.000099 was reached. Also, a plot of the best regression shows an overall correlation coefficient of 0.9999. The errors at the training points are as shown in Figure 6.83(b).

A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.43% and 0.34% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder.

The 4-15-15-1 network described above was also retrained using BR algorithm. Training stopped after 291 epochs. The plot of the best regression shows an overall correlation coefficient of 0.99989. A total of 100 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing.



(a) (b)

Figure 6.83: (a) **Performance plot for 4-15-15-1; (b) Error histogram for 4-15-15-1**

The result obtained from the test dataset showed maximum and minimum errors of 0.80% and 0.15% respectively. The maximum error recorded was also for a fault at the beginning of the main feeder. A plot of the best regression shows an overall correlation coefficient of 0.99999.

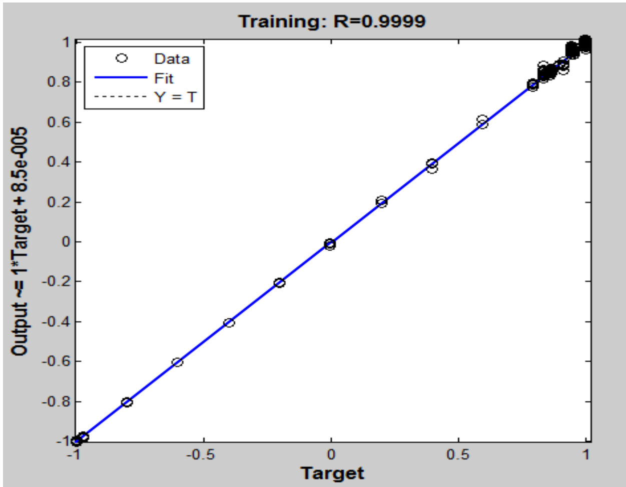


Figure 6.84: Regression plots for 4-15-15-1

6.6.3.8 Discussion of the Results

The result obtained shows that the best structure to use is 4-10-10-1. The final structure is shown in Figure 6.85.

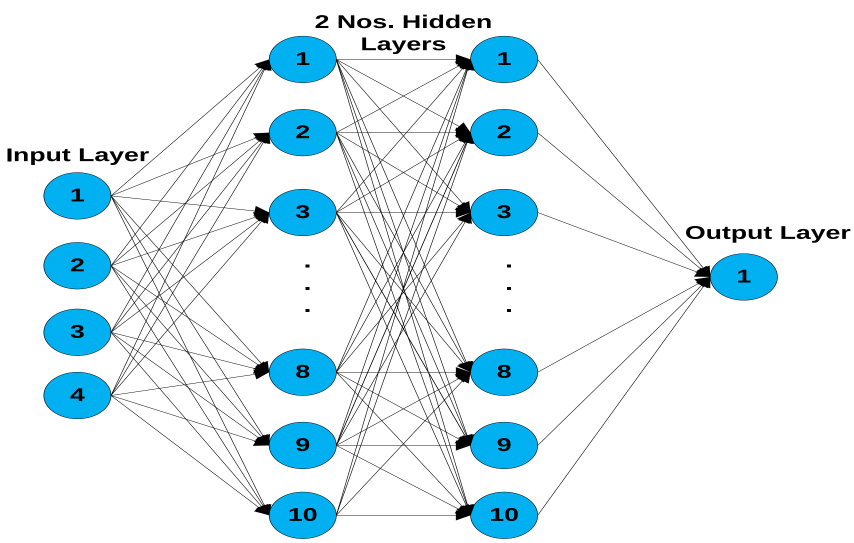


Figure 6.85: Final ANN for 2Ph.-G fault location (4-10-10-4)

From Tables 6.43-6.45, it can be seen that the mean coefficient of regression obtained using Levenberg-Marquardt algorithm (0.9998) was close to that obtained using Bayesian Regularization (BR) algorithm (0.9999). This implies that L-M algorithm is a good choice for this task with an advantage of quick training time compared with BR algorithm. Results also show that the Mean Square Error (MSE) and confusion matrix improved with increase in the number of hidden layers and hidden layer neurons. However, the guideline used in the thesis was to select the smallest neural network with good performance.

Table 6.43: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.9999	0.9999	0.9999
LM	0.9998	0.9997	0.9999
SCG	0.9974	0.9952	0.9986
RP	0.9949	0.9934	0.9966
OSS	0.9954	0.9915	0.9972

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.44: **Effect of increasing the hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Purelin*	5	0.005	0.1	10	0.027000	0.97110
MLP	LM	*Tansig/Purelin*	10	0.005	0.1	10	0.007380	0.99219
MLP	LM	*Tansig/Purelin*	15	0.005	0.1	10	0.002640	0.99721
MLP	LM	*Tansig/Purelin*	20	0.005	0.1	10	0.000947	0.99900
MLP	LM	*Tansig/Purelin*	25	0.005	0.1	10	0.000392	0.99959
MLP	LM	*Tansig/Purelin*	30	0.005	0.1	10	0.000158	0.99983
MLP	LM	*Tansig/Purelin*	35	0.005	0.1	10	0.000049	0.99995
MLP	LM	*Tansig/Purelin*	40	0.005	0.1	10	0.000035	0.99996
MLP	LM	*Tansig/Purelin*	45	0.005	0.1	10	0.000020	0.99998

Table 6.45: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Regression
1	5	LM	*TANSIG*	0.027000	0.97110
1	10	LM	*TANSIG*	0.007380	0.99219
1	15	LM	*TANSIG*	0.002640	0.99721
2	5	LM	*TANSIG/TANSIG*	0.003650	0.99614
2	10	LM	*TANSIG/TANSIG*	0.000085	0.99991
2	15	LM	*TANSIG/TANSIG*	1.7E-06	0.99999

6.6.4 Fault Location for Three Phase Faults

The results obtained for three phase fault location are given below. Several factors were considered while choosing the optimum architecture for this task. The best network was chosen based on their regression result, MSE, error histogram, and independent testing using a hold-out test dataset. The gradient and validation performance plots were also monitored during training. Results are shown firstly for small neural networks, then for large ones with varying combination of hidden layer(s) and number of neurons per hidden layer.

6.6.4.1 Network 1 [4-5-1]

The first network trained had a single hidden layer with 5 neurons. The target MSE was 0.0001. The training algorithm used was Scaled Conjugate Gradient (SCG). The activation functions in the hidden and output layers were *tansig* and *purelin* respectively.

A number of performance analyses were carried out on the network. The first of these was the performance plot. The training lasted for 35 iterations and stopped after 6 validation checks. The second means of testing was the error histogram as shown in Figure 6.86. The blue bars represent training data; the green bars represent the validation data, while the test data is represented by the red bars. Most errors fell between -0.005158 and 0.004698. Outliers were also found in the dataset. There were training points with errors of -0.01501, -0.01304, -0.01107, -0.009101, -0.007129, 0.006669, 0.00864, 0.01061, and 0.012, 0.058. The validation points were with errors at -0.1896, -0.09101, -0.007129, and 0.01258. These outliers were also visible on the testing regression plot at -0.01896, -0.09101, -0.007129, 0.006669, 0.00864, 0.01258, and 0.0185.

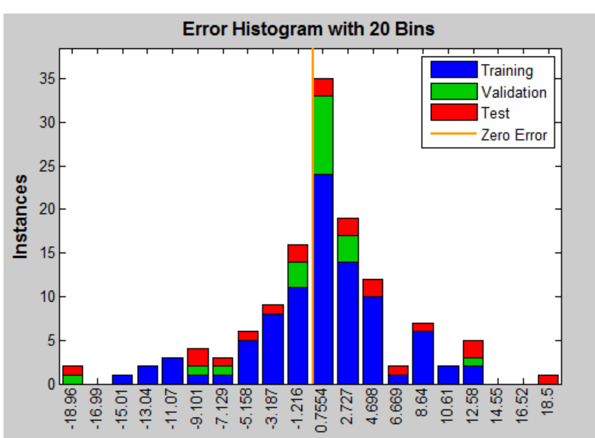


Figure 6.86: **Error histogram for 4-5-1**

Thirdly, the training regression plot was analysed. These plots are as shown in Figure 6.87. The dotted line in the figure indicates the ideal regression fit, while the coloured solid lines indicate the actual fit of the neural network. It can be seen that these lines track each other very closely which is an indication of very good performance by the neural network. A plot of the best regression shows an overall correlation coefficient of 0.95296.

The forth validation method in the testing process is to create a separate test dataset to analyze the performance of the trained neural network. A total of 30 fault cases have been simulated comprising of faults at different locations at different fault resistance and fault inception angles. The test dataset is untrained and is generated with parameters that are different from that used for the training dataset. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 11.3% and 0.59% respectively. The maximum error recorded was for a fault at the beginning of the main feeder.

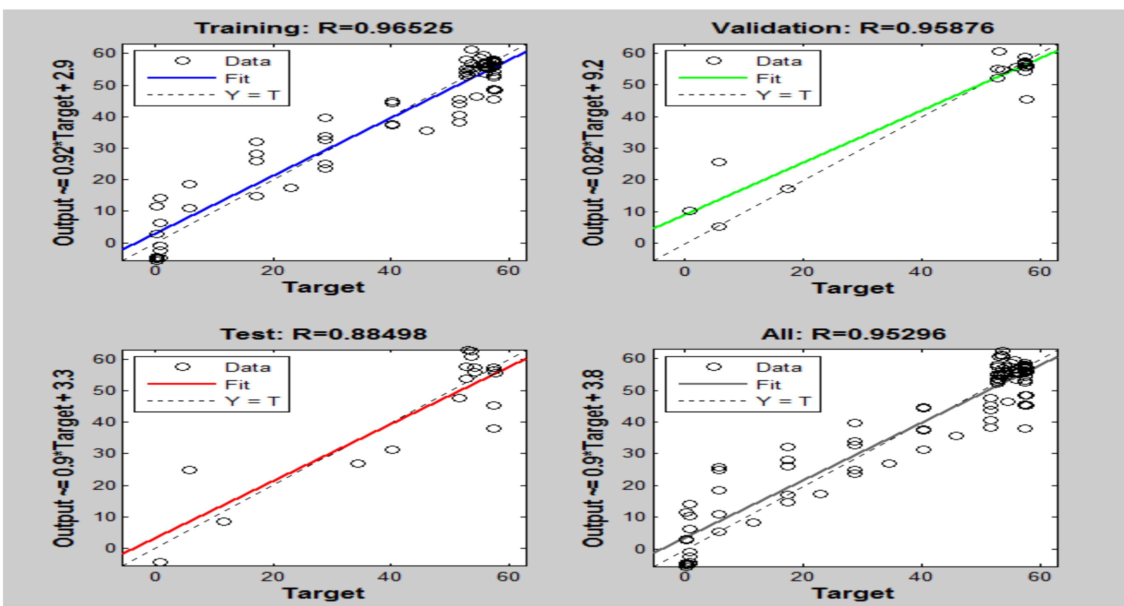


Figure 6.87: **Regression plots for 4-5-1**

6.6.4.2 Network 2 [4-5-1]

Retraining the above network using L-M algorithm stopped after 17 epochs. A plot of the best regression shows an overall correlation coefficient of 0.98821.

A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 2.1% and 0.51% respectively. The maximum error recorded was for a fault at the beginning of the main feeder.

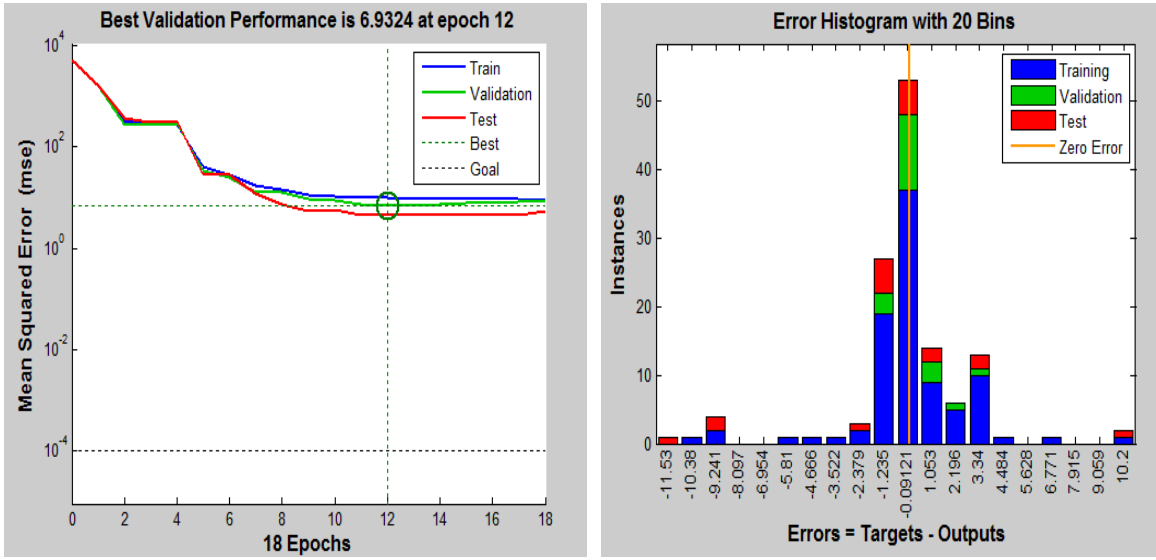
Further retraining using L-M algorithm and reducing the MSE goal to 0.001 resulted to an overall correlation coefficient of 0.99211. A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. The result obtained from the test dataset showed maximum and minimum errors of 3.3% and 0.70% respectively. The maximum error recorded was for a fault at the beginning of the main feeder.

6.6.4.3 Network 3 [4-10-1]

This network was trained with 10 neurons in a single hidden layer. The training algorithm was L-M algorithm. The target MSE was 0.0001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 18 iterations.

The performance plot and the errors at the training, validation, and test points are as shown in Figures 6.88(a) and 6.88(b) respectively.

A plot of the best regression shows an overall correlation coefficient of 0.99076 as shown in Figure 6.89. A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was also used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.28% and 0.01% respectively. With the BR algorithm, retraining stopped after 409 epochs.



(a) (b)

Figure 6:88: (a) Performance plot for 4-10-1; and **(b)** Error histogram for 4-10-1

The performance plot showed further decrease in the MSE. Also, a plot of the best regression shows an overall correlation coefficient of 0.9997. A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing.

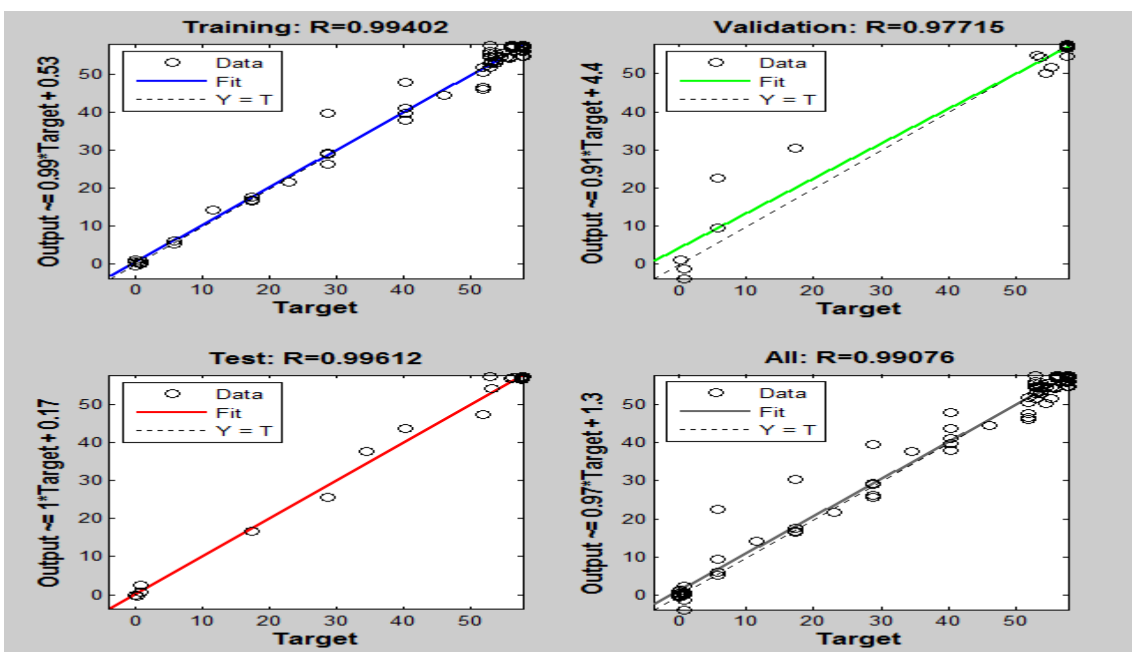


Figure 6.89: Regression plots for 4-10-1

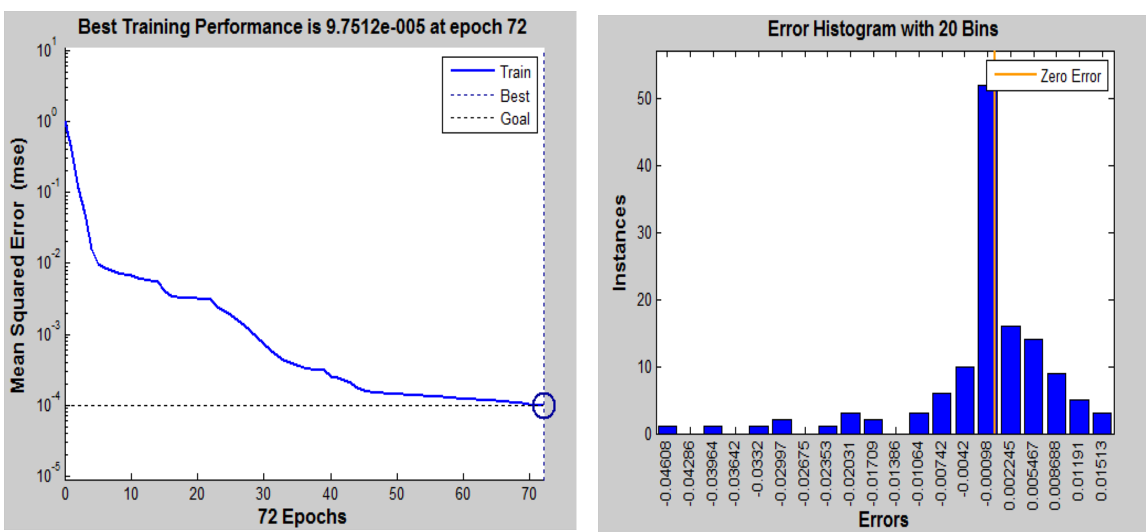
The result obtained from the test dataset showed maximum and minimum errors of 1.81% and 0.261% respectively. The maximum error recorded was for a fault at the beginning of the main feeder.

6.6.4.4 Network 4 [4-10-10-1]

This network was trained with 10 neurons in two hidden layers respectively. The training algorithm was L-M algorithm. The target MSE was 0.0001. The activation functions in the hidden and output layers were *tansig* and *purelin* respectively. Training stopped after 72 epochs with a MSE of 0.000098.

A plot of the best regression shows an overall correlation coefficient of 0.99991. The errors at the training points are as shown in Figure 6.90(b).

A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.47% and 0.02% respectively.



(a) (b)

Figure 6.90: (a) **Performance plot for 4-10-10-1; and (b) Error histogram for 4-10-10-1**

The performance goal of 0.0001 was achieved in 57 epochs when the above network was retrained with BR algorithm. Also, a plot of the best regression shows an overall correlation coefficient of 0.9999.

A total of 30 fault cases simulated at different locations at different fault resistance and fault inception angles were also used for independent testing. Equation (5.14) was used to compute the percentage error on the calculated fault location of the fault cases used for testing. The result obtained from the test dataset showed maximum and minimum errors of 0.59% and 0.04% respectively.

6.6.4.5 Discussion of the Results

From the foregoing, the best structure for the three phase fault location ANN based on the criteria hitherto mentioned, is 4-10-1. The final structure is shown in Figure 6.91.

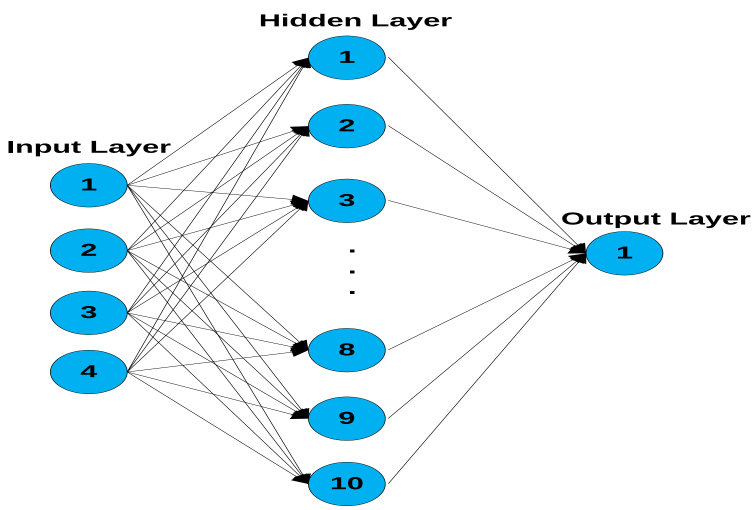


Figure 6.91: **Final ANN for 3ph. fault location (4-10-1)**

Table 6.46 gives the regression analysis of the various training algorithm used. Tables 6.47-6.48 give a summary of the effect of increasing the number of hidden layer neurons and number of hidden layer respectively.

From these tables, it can be seen that the best coefficient of regression was obtained using Levenberg Marquardt algorithm while Bayesian Regularization (BR) algorithm was next. Thus, L-M algorithm was selected. Results also show that the Mean Square Error (MSE) and confusion matrix improved with increase in the number of hidden layers and hidden layer neurons.

However, the guideline used in the thesis was to select the smallest neural network with good performance. This is because bigger networks easily over-learn the training dataset and would fail to generalise when presented with untrained dataset.

Table 6.46: **Analysis of the Regression Results for the Chosen Architecture**

Algorithm	Mean regression	Min. regression	Max. regression
BR	0.9988	0.9940	0.9999
LM	0.9999	0.9998	0.9999
SCG	0.9963	0.9959	0.9969
RP	0.9932	0.9915	0.9947
OSS	0.9951	0.9945	0.9955

Table 6.47: **Effect of increasing the hidden layer neurons**

Type of ANN	Learning algorithm	Activation function	No. of neurons in hidden layer	mu	mu_dec	mu_inc	MSE	Confusion matrix/(%)
MLP	LM	*Tansig/Purelin*	5	0.005	0.1	10	0.001910	0.99810
MLP	LM	*Tansig/Purelin*	10	0.005	0.1	10	0.000171	0.99983
MLP	LM	*Tansig/Purelin*	15	0.005	0.1	10	8.24E-05	0.99992
MLP	LM	*Tansig/Purelin*	20	0.005	0.1	10	4.80E-06	0.99999

where *mu* is the Marquardt adjustment parameter, *mu_dec* and *mu_inc* are the decrease and increase factors for *mu* respectively.

Table 6.48: **Effect of increasing the number of hidden layers**

No. of hidden layer	No. of neurons in hidden layer	Learning algorithm	Hidden layer activation function	MSE	Regression
1	5	LM	*TANSIG*	0.001910	0.99810
1	10	LM	*TANSIG*	0.000171	0.99983
1	15	LM	*TANSIG*	8.24E-05	0.99992
2	5	LM	*TANSIG/TANSIG*	0.00420	0.99581
2	10	LM	*TANSIG/TANSIG*	3.08E-06	0.99999
2	15	LM	*TANSIG/TANSIG*	6.58E-07	0.99999

6.7 Discussion of the Results

Excellent results were obtained for the HFDD method. The fault detection tasks were implemented with the wavelet energy entropy values of the three phase and zero sequence currents, while the fault classification task utilized the wavelet energy entropy per unit as was proposed in this thesis. Decisions for the detection and classification tasks were made by rules drawn up from the described extensive simulations.

Similarly, the fault section identification and fault location tasks made use of the wavelet energy entropy per unit weight of three phase and zero sequence currents. Decisions were made using ANNs respectively. Tables 6.49 and 6.50 give a summary of the final ANNs used for fault section identification and fault location respectively.

Table 6.49: **Summary of the ANNs used for the fault section identification task**

Fault Type	Final ANN Structure	Training Algorithm	Activation Function	MSE	Confusion Matrix (%)	Regression Coefficient	Test Accuracy (%)
Single Phase-to-Ground	4-18-18-4	L-M	*Tansig/Tansig*	0.03599	87.57	0.9676	89.0
Two Phase	4-21-4	L-M	*Tansig/Tansig*	0.01188	92.40	0.9374	91.0
Two Phase-to-Ground	4-10-4	L-M	*Tansig/Tansig*	0.17700	96.36	0.8979	89.0
Three Phase	4-12-4	L-M	*Tansig/Tansig*	0.05140	92.60	0.9671	93.3

Table 6.50: **Summary of the ANNs used for the fault location task**

Fault Type	Final ANN Structure	Training Algorithm	Activation Function	Regression Coefficient	Max/Min Fault Location Error (%)
Single Phase-to-Ground	4-21-1	L-M	*Tansig/Purelin*	0.9686	1.07/0.13
Two Phase	4-15-15-1	L-M	*Tansig/Purelin*	0.9999	1.20/0.70
Two Phase-to-Ground	4-10-10-1	L-M	*Tansig/Purelin*	0.9998	0.55/0.10
Three Phase	4-10-1	L-M	*Tansig/Purelin*	0.9999	0.28/0.01

6.7.1 Discrete Wavelet Transform

In order to select the best mother wavelet for the signal decomposition, a number of mother wavelets from different wavelet families were considered. Examination of the wavelet functions of the various wavelet families were done to decide on which to experiment with. Afterwards, investigations were carried using Daubechies db2, db3, db4, db5, db8, Coiflet-3, and Symlet-4.

Figures 6.1-6.4 show that the best mother wavelets matching the signal being decomposed was the Daubechies db4. Next to it was the Coiflet-3 mother wavelet.
Furthermore, Tables 6.1-6.4 show that Daubechies db4 mother wavelet displays the best characteristics that can be used for the fault detection and diagnosis tasks through the analyses of the wavelet energy and wavelet entropy values. Conclusion inferred from extensive experimentations was that level-1 detail decomposition was the best for fault detection, while level-5 was best for the other diagnostic tasks.

6.7.2 Fault Detection

The algorithm for the fault detection task was able to differentiate between fault events and 'no fault'. No faults conditions like load switching, capacitor switching, and load angle were simulated in addition to steady-state simulations.

From Table 6.5, it can be seen that the high short circuit current in the faulted phase(s) was translated into high wavelet energy in the faulted phase. The relative wavelet energy was calculated and the corresponding entropy was used as the basis for fault detection and classification in the thesis. The calculation of the relative wavelet energy makes the faulted phase(s) to be associated with low entropies.

Tables 6.7-6.10 show the results obtained at fault locations close to the upstream substation, at 70% of the length of the main feeder, at a lateral, and at a location 189,205 ft. away from the upstream substation respectively. Table 6.7 presents some of the values obtained for fault detection at 10% of the length of the main feeder (Line 806-808) for all the case studies.

The proposed technique was tested using several cases comprising of various fault types, fault conditions, and system parameters. The effects of the following are considered: Fault resistance, fault distance, fault inception angle, line extension, and the integration of DGs.

A couple of the simulation waveform plots showed the existence of mutual coupling between the phases especially for faults closer to the substation. From the results presented, magnetizing effect of mutual coupling did not have any negative effect on the fault detection capabilities of r_{p1}, $p=a,b,c,I_0$.

One of the capabilities of the proposed method is its ability to accurately distinguish between load switching conditions and faults. Load switching was simulated by switching on loads at different nodes at different time during the simulation. Similarly, changes in the peak load level were also modelled by varying the grid phase angle from 0^o to 90^o. This implies that the method would perform well in typical networks where the loading is constantly changing. Table 6.10 presents the result for a lateral. In the studies performed so far, the proposed method performed satisfactorily for the DG case studies and did not have any adverse affect on the detection of faults. Table 6.11 shows that the proposed method was able to detect faults for the DG and line extension modified case studies with the same threshold used in the base case without any modification. The line extension case study was detected and classified accurately and the pattern obtained was very similar to that of the base case.

6.7.3 Fault Classification

The algorithm for the fault classification task had to make decisions to distinguish between 1Ph.G, 2Ph., 2Ph.-G, and 3Ph. faults, and the phase(s) involved in this fault.
Tables 6.3-6.17 show the results for faults at various fault inception angles and fault resistances. For all the methods presented, the faulted phase is associated with values many times greater than the healthy/unfaulted phase(s). The proposed technique was tested using several cases comprising of various fault types, fault conditions, and system parameters. The effects of the following are considered: Fault resistance, fault distance, fault inception angle, line extension, and the integration of DGs.

A couple of the simulation waveform plots showed the existence of mutual coupling between the phases especially for faults closer to the substation. From the results presented, magnetizing effect of mutual coupling did not have any negative effect on the fault detection and classification capabilities of r_{p1} and λ_{p5}, $p = a,b,c,I_0$. Although, there were slight differences in the computed values of r_{a5}, r_{b5}, r_{c5} with different fault resistances, the phase entropy calculated per unit for a given fault still showed the same pattern. Thus, the method is robust against fault resistances.

Furthermore, the thresholds used for the fault detection and classification for the base case performed well even for the DG cases without the need to review these thresholds. Simulation plots and entropy results showed the existence of mutual coupling in the phases especially for faults in close proximity to the DG location. However, the algorithm was able to accurately distinguish between the healthy phase(s) and the faulted phase(s). Although, different entropy values were obtained for different combination of fault resistances and fault inception angles for the same location, that did not affect the classification performance since the results obtained per fault types were apparently above the pre-defined thresholds.

Results demonstrate that the location of the fault does not have any effect on the classification capabilities of the proposed method. The faulted phase(s) were distinct and remained low in terms of λ_{p5} values. The highest values of λ_{p5} were obtained for R_f of 0Ω and θ_{fA} of 45°. These tables further demonstrate the consistency of this method as the pattern obtained for these various conditions remained unchanged. For instance, the values of λ_{p5} for the faulted phase(s) remained the lowest of all irrespective of the fault resistance and fault inception angle.

The proposed algorithm performed satisfactorily for the DG case studies and did not have any adverse affect on the classification of faults except for areas in the immediate vicinity of the DG. The line extension case study was classified accurately and the pattern obtained was very similar to that of the base case. Table 6.18 show some results for the classification of faults for the modified case studies. The pattern remained consistent with the faulted phase(s) having the lowest λ_{p5} values. The distribution plots in Figure 6.7 demonstrate that entropy per unit calculated from level-5 detail coefficient is a good criterion for fault classification. Figure 6.8(b) presents a distribution plot for visualizing 1Ph.-G fault with entropies per unit computed from coefficients obtained from level-6 detail decomposition. This figure shows that it could be problematic distinguishing between different fault types. Figures 6.9(a) and 6.9(b) show the distribution plots of single phase-to-ground for DG1, and also 2Ph. faults for DG2. This clearly depicts the classification capability of the entropy per unit values derived from the level-5 DWT detail decomposition.

However, the proposed method failed in some aspects. In the base case, some faults with high fault resistance of 100Ω were misclassified for A-G at locations 7 to 10. These faults were misclassified as C-G faults. Similarly, faults located close to the DGs were misclassified for the modified case studies. For DG1 case study where the generator is located at Node 840, it was observed that A-G and BC-G were the only ones misclassified. This is as shown in Table 6.19. For DG2 case study with DG located at 844, only A-G and AB-G faults were misclassified. Lastly, for DG3 case study comprising of DGs at nodes 840 and 844, A-G, BC-G, and CA-G faults were misclassified. The reason for these misclassifications could be because of the mutual coupling in the phases.

Furthermore, the voltage improvement/fault current increase in phase A when DGs are integrated is another reason for these misclassifications. This increase causes an imbalance between phase A and the other phases and is demonstrated by phase A having a higher value. That implies that the energy in the healthy phase is high and almost equal to that in the faulted phase(s).

The misclassification can be minimized by complementing with wavelet energy indices, and of course, updating the rules. The values obtained for STD and MAD after the integration of DGs were not completely useful for fault classification because there were similarities between the values obtained for faulted phase(s) in one location and the values for healthy phase(s) at another location. Table 6.20 illustrates some of the errors obtained when using Standard Deviation (STD) and Mean Absolute Deviation (MAD). Entry 5 of Table 6.20 was misclassified as ‘no fault’. Entry 6 was also misclassified as A-B fault. Entries 7 and 8 were wrongly denoted as ‘no fault’ respectively. The values of STD and MAD showed a corresponding decrease in value for faults with high resistances and for faults located far away from the substation. This implies that STD and MAD are influenced by fault resistance and fault location. Thus, showing entropy to be of better performance.

6.7.4 Fault Section Identification

Several learning algorithms were used for the Multi-Layer Perceptron (MLP) NN. The best results were generally obtained from L-M training algorithm. However, these results were close to that obtained using BR and RP training algorithms. The activation functions for the hidden and output layers were *tansig*. Performance goals of 0.0001, 0.001, 0.01, and 0.1 were used respectively. The number of neurons in the hidden layer was varied for both single and two hidden layer(s) architectures.

Also, the number of epochs was varied in the course of this research. The effect of varying the structure of the NN by increasing the number of neurons in the hidden layer, increasing the number of hidden layers, increasing the number of epochs, and decreasing the performance goal on the performance/accuracy of the NN was investigated. Misclassification occurred between line sections at Lateral 836-838, Lateral 836-840, and main feeder lines around this section.

The results obtained are encouraging and the FSI algorithm can be further improved by adding information from fault path indicators or some other intelligent agents in order to differentiate between Lateral 836-838 and Lateral 836-840.

6.7.4.1 The Performance of the Training Algorithms

The performances of five learning algorithms were investigated. The results obtained for several variations of hidden layer neurons and performance goals showed that L-M training algorithm produced the best results in general. This is as shown in the confusion matrix and regression result tables at the end of sub-sections 6.5.1, 6.5.2, 6.5.3, and 6.5.4. From the tables, it is seen that LM algorithm produced the best result in terms of the confusion matrix and regression result analysis for single phase-to-ground faults. RP produced the best confusion matrix for two phase faults and two phase-to-ground faults respectively. However, the best regression results for two phase faults and two phase-to-ground faults were obtained with LM algorithm. For three phase faults, the best confusion matrix and regression result analysis was obtained with RP and LM algorithms respectively.

A summary of this comparison is given in tables at the end of sub-sections 6.5.1, 6.5.2, 6.5.3, and 6.5.4.

6.7.4.2 The Effect of Increasing the Number of Hidden Layer Neurons

The effects of increasing the number of hidden neurons on the performance of the ANNs were examined for all the fault types. The number of epochs was constant at 1000 and the performance goal was fixed at 0.01. Network pruning was implemented for single phase faults (H=55:5:5), while network growing was used for 2 phase, 2 phase-to-ground, and three phase faults (H=5:5:45/45/25) respectively. This implies that the first network for the single phase fault section identification had 55 neurons in its hidden layer which is the maximum number of hidden layer neurons possible based on derivations obtained from equations (5.10)-(5.12) for relating network equations and weights.

Conversely, network growing for 2 phase, 2 phase-to-ground, and three phase faults started from 5 hidden layer neurons.

It was observed that for single phase faults, lower MSE less than 0.1 was only possible when the hidden layer neurons was greater than or equal to 20. Confusion matrix values of 90% were obtained for these architectures. The same applies to 2 phase faults with the best confusion matrix obtained hidden layer neuron of 20.

Thus, an increase in the number of hidden layer neurons resulted to a decrease in the MSE for these cases. For two phase-to-ground faults, increase in the number of hidden layer neurons resulted to decrease in MSE but did not result to an increase in the confusion matrix.

The effect of increase in the number of hidden layer neurons resulted to a decrease in MSE and improvement in the confusion matrix up to 15 hidden layer neurons. Increasing the number of neurons to 20 showed a deterioration of the MSE and confusion matrix. It was observed that increasing the number of hidden layer neurons did not converge the network; neither did it improve the accuracy or generalization of the trained network.
A summary of this comparison is given in tables at the end of sub-sections 6.5.1, 6.5.2, 6.5.3, and 6.5.4.

6.7.4.3 The Effect of Increasing the Number of Hidden Layers

The effects of increasing the number of hidden layers on the performance of the ANNs were also examined for all the fault types. The effects of having 2 hidden layers was investigated for H=5:5:25. 25 being the maximum number of hidden layer neurons that can be in each hidden layer as per the network equation and weight rule.

Increasing the number of hidden layers improved the performance of the ANN for single phase-to-ground faults. However, increasing the number of hidden layers did not improve the performance of the network for two phase faults. For two phase-to-ground faults, an increase in the hidden layer neurons resulted to the depreciation of the confusion matrix, but did improve the regression result.
A summary of this comparison is given in tables at the end of sub-sections 6.5.1, 6.5.2, 6.5.3, and 6.5.4.

6.7.4.4 The Effect of Increasing the Number of Epochs

This section examines the effect of increasing the number of training iterations (epochs) on the network performance. The number of neurons in the hidden layer of the worst network was kept constant while the effect of increasing the number of epochs (iteration) was investigated. The increase in the number of iterations is directly proportional to the increase in training time. The number of iteration was increased from 1000 to 3000.

When the performance plots are examined, it was observed that the MSE obtained has not changed from that of 1000 epochs. The key point is to weight the improvement of the MSE value and the associated decrease in the generalization capability of the ANN as the number of epochs is increased.

6.7.4.5 The Effect of Decreasing the Performance Goal

The effects of decreasing the performance goal were investigated. The number of neurons in the hidden layer of the worst network was kept constant while the performance goal was decreased. The four performance criteria mentioned in this thesis were monitored. There was an improvement in the confusion matrix when the performance goal was reduced from 0.0001 to 0.001. The same applies to the classification accuracy. The neural networks were trained to a target output of 0 and 1. Where zero is ‘untrue’ and 1 is ‘true’. Generally, any result greater than 0.5 was taken as ‘true’ and less than 0.5 was taken as ‘untrue’.

6.7.5 Fault Location

Several learning algorithms were used for the MLP NN for the fault location task. The best results were generally obtained from L-M training algorithm. However, these results were close to that obtained using BR and RP algorithms. The activation functions for the hidden and output layers were *tansig*. Performance goals of 0.0001, 0.001, 0.01, and 0.1 were used respectively. The number of neurons in the hidden layer was varied for both single and two hidden layer(s) architectures. Also, the number of epochs was varied in the course of this research. The effect of varying the structure of the NN by increasing the number of neurons in the hidden layer, increasing the number of hidden layers, increasing the number of epochs, and decreasing the performance goal on the performance/accuracy of the NN was investigated. The algorithm did not show any signs of depreciation even as the fault resistance and fault inception angles changed.

However, it should be noted that the maximum errors obtained during the NN training and testing were for faults with high resistance of 100Ω. The only errors recorded in most cases were for faults located at the beginning of the feeder.

6.7.5.1 The Performance of the Training Algorithms

The performances of five learning algorithms were investigated. The results obtained for several variations of hidden layer neurons and performance goals showed that L-M algorithm produced the best results in general. This is as shown in the regression result table at the end of sub-sections 6.6.1, 6.6.2, 6.6.3, and 6.6.4. From the table, it is seen that BR and LM algorithms produced the best result in terms of the regression result analysis for single phase-to-ground faults. The same applies to two phase faults, two phase-to-ground faults, and three phase faults respectively.

6.7.5.2 The Effect of Increasing the Number of Hidden Layer Neurons

The effects of increasing the number of hidden neurons on the performance of the ANNs were examined for all the fault types. The number of epochs was constant at 1000 and the performance goal was fixed at 0.0001. Network growing was implemented (H=55:5:5) for single phase-to-ground, two phase, two phase-to-ground, and three phase faults respectively.

It was observed that for single phase faults, lower MSE less than 0.01 was only possible when the hidden layer neurons was greater than 15. Generally, very good regression values were obtained for the single phase-to-ground faults. Regression values greater than 0.95 was obtained even for five hidden layer neurons. The same applies to the other fault types. Subsequent increase in the number of hidden layer neurons resulted to a decrease in the MSE obtained and corresponding improvement in the regression values. Although, lower maximum and minimum errors were obtained with the 4-15-15-1 network for two phase-to-ground faults, the 4-10-10-1 network was chosen as the preferred final network because it has lower weights (171 weights) compared to the former (330 weights) and it presented an error of less than 1%.

A summary of this comparison is given in tables at the end of sub-sections 6.6.1, 6.6.2, 6.6.3, and 6.6.4.

6.7.5.3 The Effect of Increasing the Number of Hidden Layers

The effects of increasing the number of hidden layers on the performance of the ANNs were also examined for all the fault types. The effects of having 2 hidden layers was investigated for H=5:25. 25 being the maximum number of hidden layer neurons that can be in each hidden layer as per the network equation and weight rule.

Increasing the number of hidden layers improved the performance of the ANN for single phase-to-ground faults. The MSE obtained was lower than for a single layer network.

Very good regression values were obtained but the difference between the final architecture and the two hidden layer networks was low. Similar conclusions can also be drawn for the two phase, two phase-to-ground, and three phase faults respectively.

A summary of the effect of increasing the number of hidden layer is given in tables at the end of sub-sections 6.6.1, 6.6.2, 6.6.3, and 6.6.4.

6.8 Performance of the HFDD Method

The system design process mentioned in Figure 5.3 was followed in the thesis. The final stage of which include system integration/verification, and system validation. In this vein, all the algorithms developed in the thesis were incorporated together and tested. The algorithms in Appendices C.4, C.5, C.6, C.7, C.8, C.14, and C.15 were integrated and used for the performance test of the HFDD method.

Test dataset different from that used in training the neural networks were used. This involved simulations of faults at locations different from that used for training. Also, the fault resistances and fault inception angle used was different. Some of the results obtained are summarized in Table 6.51.

The detection of fault events was 100% accurate, and very promising results were obtained for the fault classification, fault section identification, and fault location tasks. The highest percentage errors were recorded for high resistance faults. As shown in Table 6.51, a percentage error of 1.53% was obtained for a B-G fault with 2.5Ω fault resistance at 60^o fault inception angle at lateral 854.

A percentage error of 2.60% was obtained for the same fault at the same location and fault inception angle, but with 5Ω fault resistance. Generally, percentage errors less than 3.0% were obtained for the test dataset. The performance test carried out using the

HFDD method demonstrates the capability of the proposed HFDD method. The only condition being that the fault type must be correctly identified in order to trigger the appropriate algorithms for fault section identification and fault location.

It must be mentioned that misclassification occurred with the fault section identification algorithm for fault sections in very close proximity. For instance, faults in lateral 836-862 was misclassified as lateral 836-840 because they are in close proximity, with similar entropy per unit values.

Table 6.51: **Performance analysis of the Hybrid Fault Detection and Diagnosis (HFDD) method**

Fault type	Fault resistance [Ω]	Fault inception angle [^{o}]	Fault section	Fault location [km]	Fault class	Fault type	Faulted section	Fault location [km]	% error
B-G	2.5	30	0010	12.1988	1Ph.	B-G	0010	11.9118	2.22
B-G	2.5	30	0100	34.9667	1Ph.	B-G	0100	34.4154	1.54
B-G	2.5	30	0101	45.0388	1Ph.	B-G	0101	45.7185	1.40
B-G	2.5	60	0101	42.0388	1Ph.	B-G	0101	41.2929	1.53
B-G	5.0	60	0101	42.0388	1Ph.	B-G	0101	40.7775	2.60
A-G	2.5	30	0111	54.4525	1Ph.	A-G	0111	53.7322	1.32
A-G	2.5	60	0111	53.4525	1Ph.	A-G	0111	52.5596	1.63
A-G	5.0	30	0111	54.4525	1Ph.	A-G	0111	55.5612	2.03
A-G	5.0	60	0111	54.4525	1Ph.	A-G	0111	55.9547	2.75
BCG	0.5	45	1000	56.2737	2Ph.	BC	0010	56.6047	0.57
C-G	2.5	30	1001	55.9251	1Ph.	C-G	1001	54.879	1.77
C-G	5.0	30	1001	57.9251	1Ph.	C-G	1001	58.7004	1.31
C-G	2.5	60	1001	57.4920	1Ph.	C-G	1001	57.6935	0.34
C-G	5.0	60	1001	57.4920	1Ph.	C-G	1001	56.8563	1.08
CA	0.5	30	0110	54.3215	2Ph.	CA	0110	52.7993	2.72
AB	0.5	0	1000	56.2737	2Ph.	AB	1000	57.2357	1.67
AB	0.5	30	0001	25.8673	2Ph.	AB	0001	25.8466	0.04
BC	0.5	30	0001	25.8673	2Ph.	BC	0001	24.4274	2.49
ABG	0.5	30	1001	57.4920	2Ph.-G	ABG	1001	57.1873	0.52
ABG	0.5	30	1010	57.4807	2Ph.-G	ABG	1010	57.1878	0.51
BCG	0.5	30	1010	57.4807	2Ph.-G	BCG	1010	58.3439	1.50
CAG	0.5	30	1010	57.4807	2Ph.-G	CAG	1010	57.7271	0.43
ABC	0.5	60	1010	57.4807	3Ph.	ABC	1010	57.7681	0.50
A-G	2.5	30	**0110**	52.3215	2Ph.-G	CAG	**1001**	51.4014	1.65

6.9 Conclusion

This chapter presented the results of the various algorithms required for the HFDD method. The fault detection and classification algorithms were achieved using decision-taking algorithm rules, while the fault section identification and fault location was by using ANNs. Excellent results were obtained for ten fault types comprising of 1Ph.G, 2Ph., 2Ph.-G, and 3Ph. faults. Also, the locations of the faults were varied including the fault resistance and fault inception angle.

The proposed algorithms have several advantages:

(a) Only current inputs are used

(b) Wider range of fault resistances and fault inception angles were considered

(c) The performance of the algorithm was good even for fault resistances above 0Ω.

(d) The effectiveness of the developed method was shown with its ability to differentiate between system conditions like load and capacitor switching even though they generate transients similar to fault transients.

Chapter 7 elaborates on the implementation of the HFDD method based on simulation result obtained from the Real-Time Digital Simulator (RTDS). These data are obtained from the runtime results in RSCAD software environment and also from event records triggered in an external Intelligent Electronic Device (IED) connected in-the-loop to the RTDS.

7.1 Introduction

After the development of the Hybrid Fault Detection and Diagnosis (HFDD) method in Chapter Five, results for the HFDD method were presented in Chapter Six. Chapter Seven provides a means of validating the fault detection, fault classification, and fault section identification algorithms of the developed HFDD method.

This was done by testing the HFDD method with data obtained in real-time using the Real-Time Digital Simulator (RTDS). The study network used was the IEEE 13 node benchmark test feeder. The validation of the HFDD method was necessary so as to prove that the proposed method works. Also, the validation of the HFDD method using this test feeder confirms the scalability of the method.

What is being envisaged in this thesis are the possible ways the Hybrid Fault Detection and Diagnosis (HFDD) method proposed in this research can be applied in reality to aid control centre dispatchers and engineers during and after faults. Figure 7.1 illustrates this. The figure shows a typical electric power system and communication network architecture.

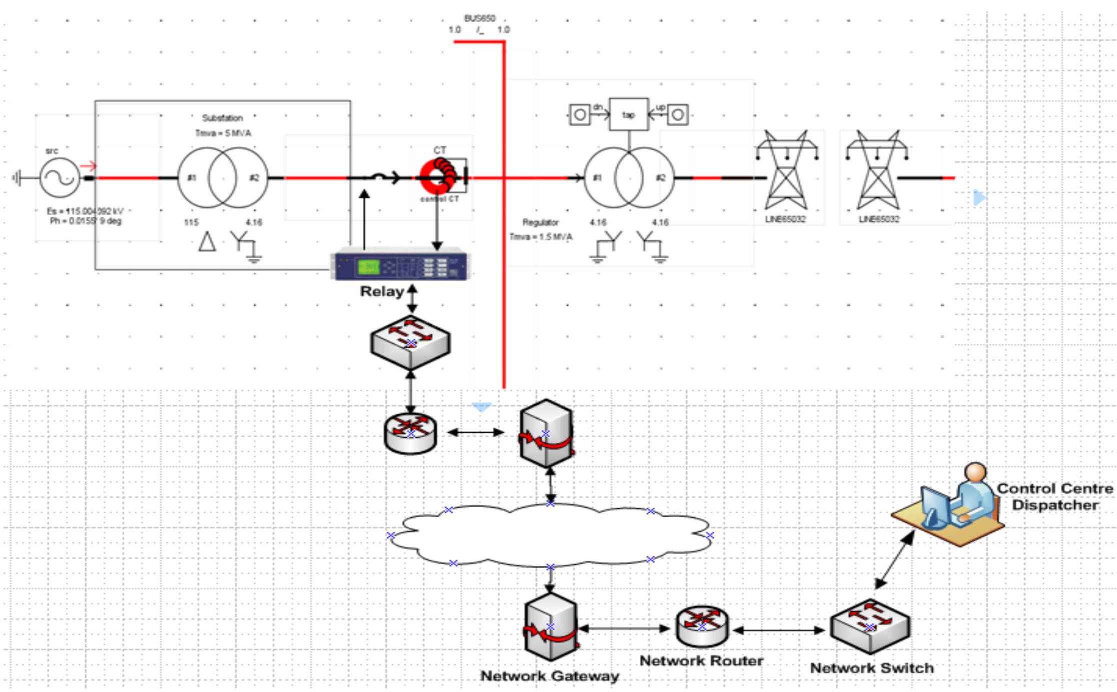


Figure 7.1: Typical implementation of the HFDD method

In order to carry out fault diagnosis, a remote control centre operator can issue out commands to retrieve fault event files from a relay or digital fault recorder. This can be done by establishing a connection to the relay or digital fault recorder of interest through the IP address. The event files can be retrieved afterwards.

After the fault event files are retrieved, the HFDD method can be applied to determine if indeed the triggered event was caused by a fault or not, classify the type of fault and faulted phases, identify the faulted section, and locate the distance where the fault is from the measurement point. With the result obtained, the dispatcher can take informed decision and give instructions to the maintenance crew for fault clearing and line restoration.

The scenario mentioned above was implemented in the laboratory through the use of the Real-Time Digital Simulator (RTDS), Intelligent Electronic Device (IED), signal amplifier, network switches, and work stations.

The software used for the modelling of the IEEE 13 node test feeder was RSCAD software. It is a proprietary software for use with the Real-Time Digital Simulator (RTDS). Although, the steps for modelling and simulation using RSCAD software is given below, further information can be obtained by using the ‘Manuals’ tab available in RSCAD/FileManager window.

An external Intelligent Electronic Device (IED) was connected in a closed-loop configuration with the RTDS. This IED was configured to record fault events during real-time batch mode simulation with the RTDS. The fault records were downloaded from the IED afterwards and used as inputs to the Hybrid Fault Detection and Diagnosis (HFDD) method.

This chapter covers the modelling and simulation of the test feeder in RSCAD software. Real-time simulations were carried out using the Real-Time Digital Simulator (RTDS) connected in closed-loop with an external IED. Also, the procedure followed in configuring the external IED is given herein. Finally, results obtained from RSCAD runtime environment and event records obtained from the IED are presented for analysis.

7.2 Real-Time Digital Simulator (RTDS) Hardware

The Real-Time Digital Simulator (RTDS) is a power system simulator capable of solving electromagnetic transient simulation in real time. It is made up of specially designed hardware like Digital Signal Processors (DSPs) and Reduced Instruction Set Computer (RISC). The RTDS utilizes advanced parallel processing techniques in order to achieve the computation speeds required to maintain continuous real-time operation (RTDS hardware manual, 2009).

The various modules that make up the RTDS are mounted in racks which together with input/output cards and power supply units are housed in cubicles. Four different cubicle types are available. These are: The full size cubicle, mid-size cubicle, mini cubicle, and the portable cubicle.

The RTDS can be utilized for:

- High speed simulations
- Closed-loop testing of protection equipment e.g. relays
- Closed-loop testing of control equipment e.g. exciters, voltage regulators, FACTS devices, power system stabilizers, etc.
- Hardware-in-the loop applications

Some of the available modules at the Centre for Substation Automation and Energy Management Systems (CSAEM) include:

- Gigabit Transceiver Work Station Interface (GTWIF) card
- Gigabit Processor Card (GPC) cards
- Gigabit Transceiver Network Interface Card (GTNET) cards
- Gigabit Transceiver Analogue Output Card (GTAO)
- Gigabit Transceiver Front Panel Interface (GTFPI) card
- Digital I/O panel
- HV Patch Panel

More information on the above-mentioned hardware modules can be found in RTDS manual, (2009).

7.3 RSCAD Software

The RSCAD software suite is the user’s main interface to the RTDS hardware. All aspects of the power system such as circuits, runtime environment, simulation results, etc. are controlled through RSCAD software. RSCAD software enables the user to schematically construct and simulate a circuit, and view/analyze results. System parameters like online plotting, controls, and metering can be viewed and manipulated during simulation.

The RSCAD software has two main modules: The Draft and the Runtime environments. The Draft module handles assembling of the circuits, power system blocks and entering of the parameter settings, while the Runtime module controls the operation of the hardware. Through the Runtime module, users can control actions such as start and stop the simulation, initiate disturbance, online monitoring of system quantities, trigger data acquisition system, etc. The interface between the RTDS and the RSCAD workstation is via ethernet cable between the GTWIF module and the RSCAD workstation.
During installation, a directory named 'RTDS_USER' was created. This user directory holds the user's RSCAD simulation cases and projects can be created within it to hold the RSCAD files as shown in Figure 7.2 (RTDS tutorial manual, 2010).

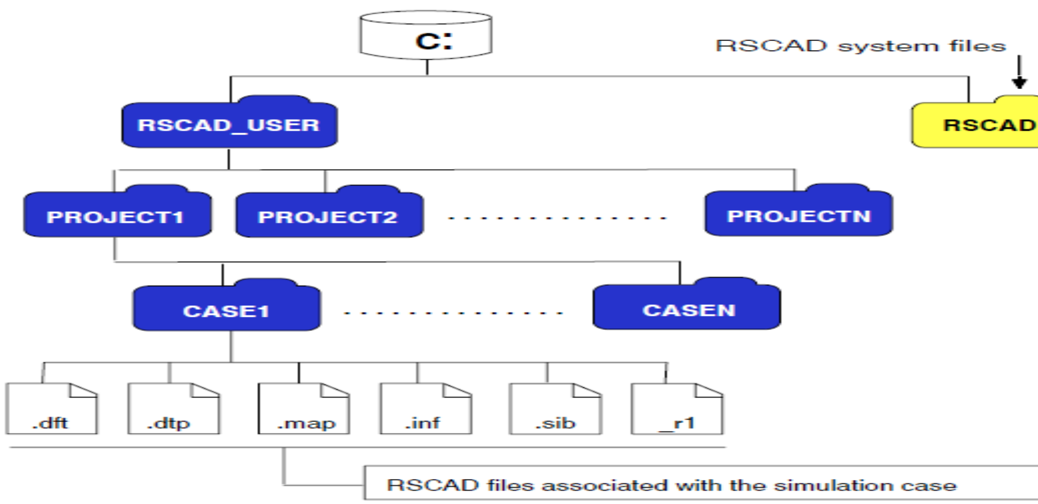


Figure 7.2: **RSCAD User File Structure (RSCAD manual, 2010)**

7.4 Modelling of the IEEE 13 Node Test Feeder

In order to validate the proposed HFDD method, it was necessary to test the concept and scalability by using field data from a real distribution network. As a result of the difficulty in obtaining such data, the RTDS was perfectly suitable for this. However, a single RTDS rack (as currently installed at CSAEM) is limited to 22 nodes. For this reason, the IEEE 13 node test feeder (Figure 7.3) was used in lieu of the IEEE 34 node test feeder.

The IEEE 13 node test feeder is one of the benchmark distribution feeders published by the Distribution Subcommittee of the IEEE Power Engineering Society. It is a short and highly loaded feeder with a nominal voltage of 4.16 kV.

The components found in the test feeder include a substation transformer, in-line transformer, shunt capacitor banks, voltage regulator, loads, and of course, overhead and underground distribution lines of various configuration.

This distribution test feeder was modelled using RSCAD software and it was made up of ten line segments. Figure 7.4 is a single line diagram of the model in RSCAD software environment.

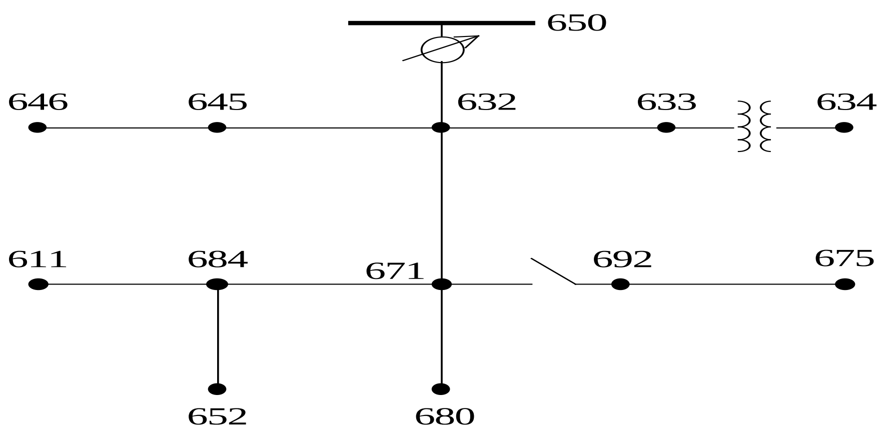


Figure 7.3: **The IEEE 13 Node Test Feeder**

7.4.1 Nodes

The ten line segments of the test feeder consist of a main feeder, four laterals, and two sub-laterals. Figure 7.4 shows the single line diagram in RSCAD software environment. Table 7.1 gives a summary of the lines and the various components connected to them.

Table 7.1: **Summary of laterals in the IEEE 13 node test feeder**

Nr.	Feeder Section	Lateral Nomenclature	Phase Type	Length (ft)	Line Type	Voltage Level (kV)	Circuit Component(s)
1.	632-646	L.632A	CBN	800	Overhead	4.16	Loads
2.	632-634	L.632B	CABN	500	Overhead	4.16	Load
3.	671-684	L.671A	ACN	300	Overhead	4.16	Load
4.	671-675	L.671B	ABCN	500	Underground	4.16	Load & capacitors
5.	684-611	L.684A	CN	300	Overhead	4.16	Load & capacitors
6.	684-652	L.684B	A-N	800	Underground	4.16	Load
10.	650-680	Main feeder	BACN	5000	Overhead	4.16	Transformer, load, & regulators

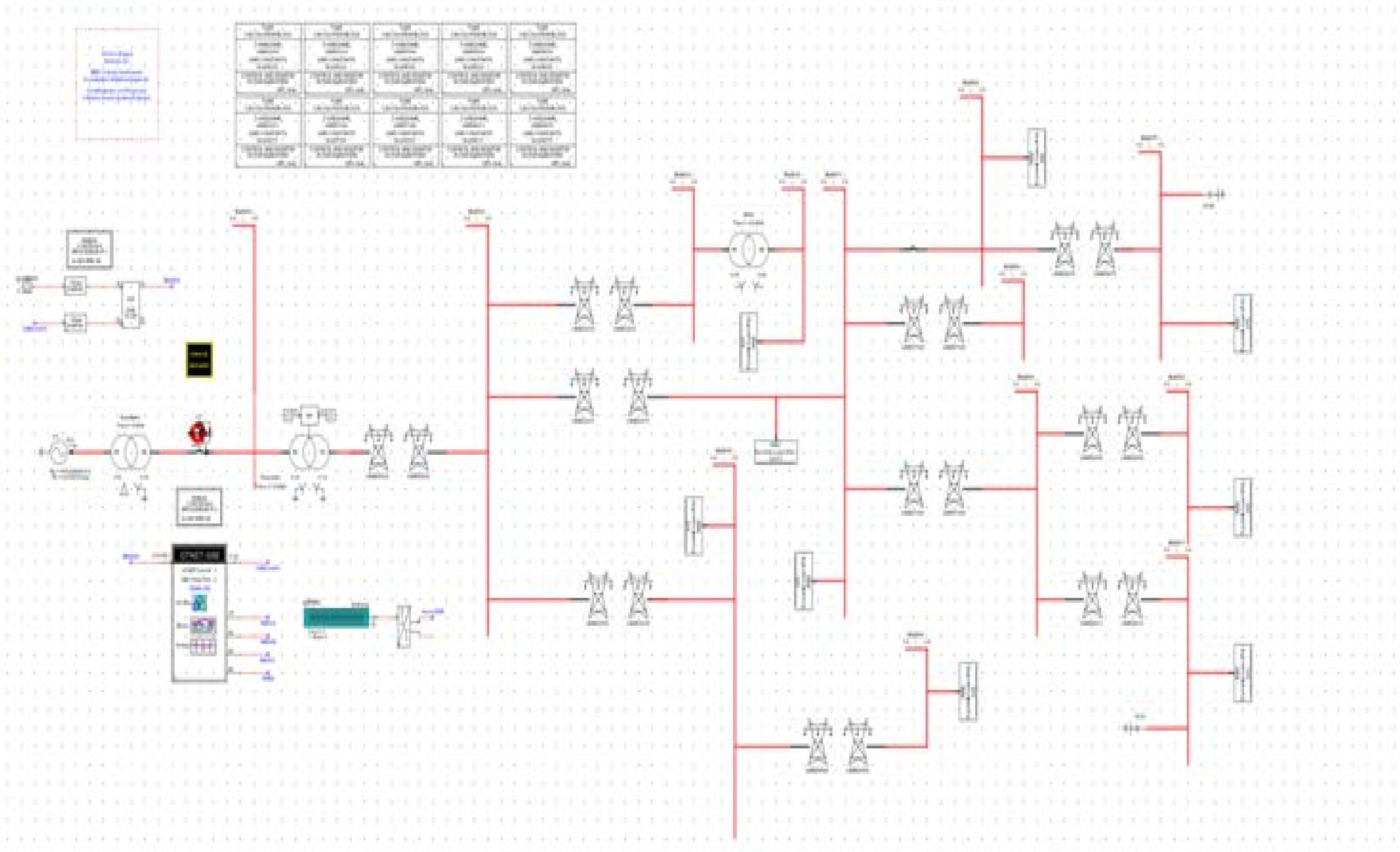


Figure 7.4: **Single line diagram of the test feeder in RSCAD software**

7.4.2 Transformer Models

A three phase source model was used for the voltage supply to the feeder. Two transformer models were used to transform the voltages at different points in the feeder. The first one is an upstream transformer located at node 650, and the second is in-line within the feeder at node 633. The upstream transformer is a 5MVA 115/4.16kV transformer and was modelled with the 'three phase two winding transformer' from the RSCAD library.

The 0.5MVA, 4.16/0.48kV in-line transformer was modelled with a 'three-phase two winding transformer' model. The parameters used are given in Appendix B.1. The transformer impedance 0.005+j0.04 p.u. was divided between the primary and secondary windings respectively.

7.4.3 Line Models

The *pi* model was used to model all the line segments since the feeder is made up of short lines. The main feeder has a total length of 8,200ft (2,499.36 metres) and it was modelled with configuration type 601 as given in Appendix B.2. Similarly, the various single phase laterals were modelled using the 602, 603, 604, 605, 606, and 607 configuration types. While 601-605 configurations are for overhead lines, 606-607 are for underground cable configurations.

The positive and zero sequence resistance, inductance, and susceptance values were entered under the **'PARAMETER'** tab in RSCAD software.

Carson's equation and Kron's reduction method were used to transform the impedance and susceptance matrices to positive and zero sequence impedances and shunt susceptances. The derivations were given in equations (4.1)-(4.6). Line lengths are given in Appendix B.3.

Table 7.2 lists the values of the positive and negative impedance and susceptance calculated from equations (4.5) and (4.6) respectively.

Table 7.2: **Line parameters for the test feeder**

Line Configuration	$R_{11} + jX_{11}$ /(Ohm/mile)	$R_{00} + jX_{00}$ /(Ohm/mile)	B_1 /(Ohm/mile)	B_0 /(Ohm/mile)
601	0.2380+j0.7640	0.5494+j1.5725	6.8785	4.1410
602	0.6461+j0.9274	0.9555+j1.7359	6.0146	4.2739
603	0.8155+j0.7483	1.0221+j1.2073	3.4252	2.5253
604	0.8844+j0.9013	0.8844+j0.9013	3.1252	3.1252
605	1.3292+j1.3475	1.3292+j1.3475	1.5064	1.5064
606	0.5824+j0.4103	1.2208+j0.4759	96.8897	96.8897
607	0.4475+j0.1708	0.4475+j0.1708	29.6637	29.6637

7.4.4 Load Model

The IEEE 13 Node Test feeder has eight spot loads and one distributed load. The sum of the entire feeder load is 3,466 kW. The load types specified either as wye or delta loads are modelled as constant power (PQ) load, constant current (I) load, and constant impedance (Z) load. The main settings on the '**PARAMETER'** tab in the RSCAD software are the **Component Name** (all the lines should have unique names so as to avoid error during compiling), **Balanced Load = yes or no** depending on if the load is a spot load or distributed load, **P & Q controlled by = Const Z** depending on the load type. Const Z is for constant impedance loads, while CC is used for constant current loads. The load parameters for the spot and distributed loads are given in Appendix B.4.

7.4.5 Voltage Regulator Models

A voltage regulator located at nodes 650-632 is used in the feeder. No voltage regulator component is available in the RSCAD software. Therefore, the regulator was modelled using three-phase two winding transformers. The primary side voltages were set equal to the regulator's input voltages, while the secondary side voltages were set equal to the regulator's output voltages. For a tap changer transformer, the '**Transformer Model Type'** on the **'CONFIGURATION'** tab in RSCAD software must be set to **ideal or saturation**, and not linear. The limits can be set by specifying the '**Tap Changer Inputs'** to **Step/Limit**. The positions can also be specified by the **'Pos Table'** option. The **'Step/Limit'** was used because only the parameters for limits were specified in Appendix B.5.

7.4.6 Shunt Capacitor Models

The capacitor banks used in the feeder are located at nodes 611 and 675 respectively. The capacitor banks are located at the extreme end of the laterals in order to regulate the voltage. The shunt capacitor data is given in Appendix B.6.

7.5 Simulations

After the model was saved and compiled, load flow was done by using the 'Run Loadflow' button in the tools bar. Frequency and tolerance can be entered into the options in the pop up page. In order to prepare the test feeder for further studies, the steady-state load flow results of the modelled feeders were compared with the IEEE's published results [Kersting, 2004].

Faults were simulated on the study network by using a script file for the batch mode operation. This was done in order to run/execute several simulations automatically without any manual interaction from the user. Figure 7.5 is an illustration of the flow chart for the batch mode operation. The "C" like script function used for the batch mode operation is given in Appendix D.

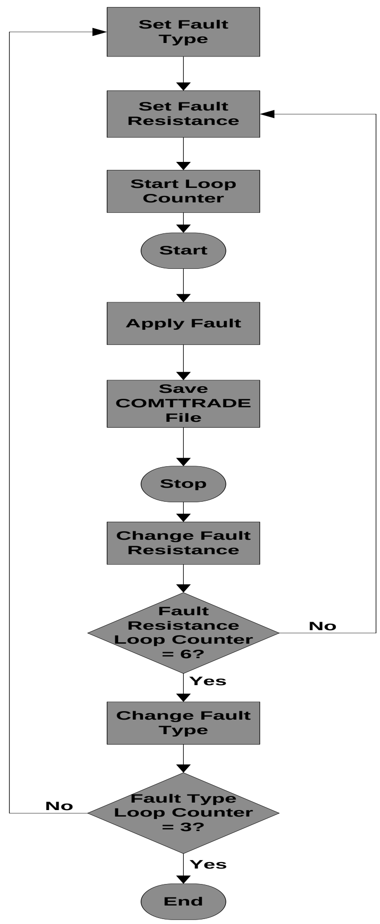


Figure 7.5: **Flow chart for batch mode operation**

After this, 130 fault cases were simulated on the feeder. These 130 cases represent 4 steady-state conditions with different loading angles of 15^o, 30^o, 60^o, and 90^o. 126 fault cases consist of different fault types at different line sections simulated with varying fault resistances of 0.1Ω, 1.0Ω, 5.0Ω, 10.0Ω, 20.0 Ω, and 100.0Ω respectively.

Steady-state and faulted case simulations were run to depict the normal operating condition of the system and conditions during faults. A fifteen cycle dynamic simulation was carried out and was monitored along Line 650-832. Also, an external SEL-451 IED was configured to serve as a digital fault recorder by using AcSELerator QuickSet software and connected to node 650. The simulation outputs from the runtime environment of the RTDS were exported via the GTAO card. The transfer of the simulation quantities from the RTDS is made possible through a GTAO analogue output component within the draft model in RSCAD software. An Omicron CMS156 Amplifier was connected to the GTAO card in order to convert the ±10V signal level at the GTAO card to the actual voltage and current levels as obtained within the simulation through the use the appropriate scale factors. It is these real quantities that are made available to the Current Transformer (CT) and Voltage Transformer (VT) modules of the IED. Figure 7.6 depicts this connection. Figure 7.7 shows the simulation control palette used in RSCAD runtime for plotting and control functions.

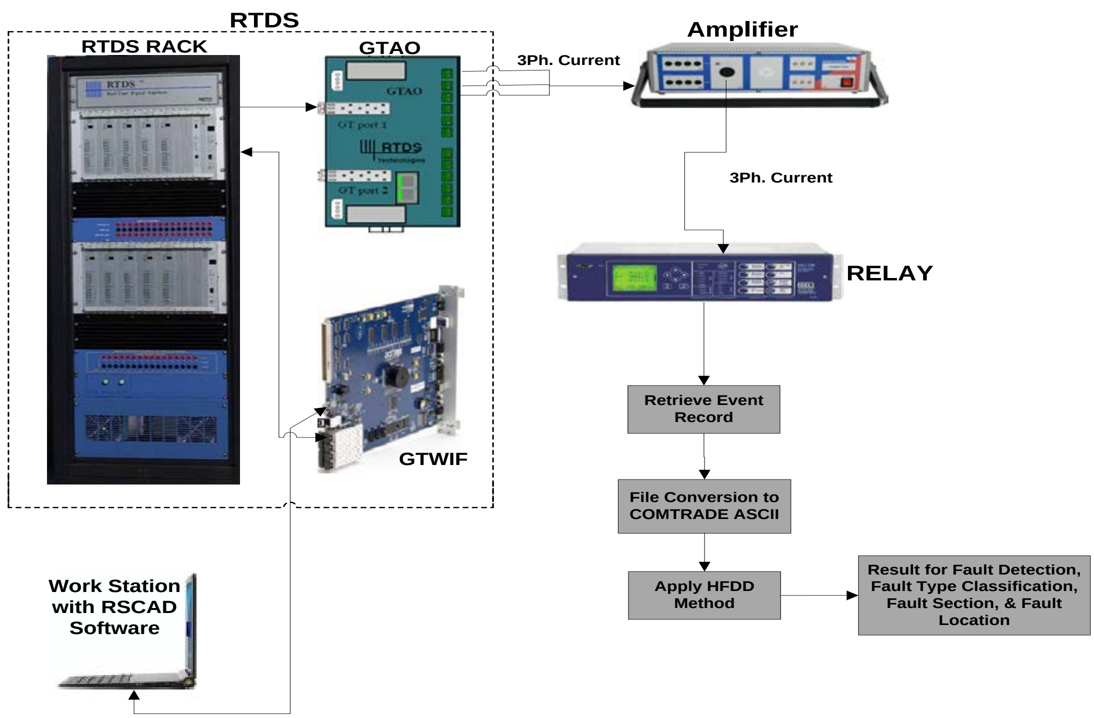


Figure 7.6: **RTDS-IED 'hardware-in-the-loop' connection**

7.6 IED Configuration

The SEL-451 IED is a distribution relay with an extensive metering and data recording capability (SEL-451 user manual, 2009). This IED was connected in closed-loop with the RTDS and used to serve as a disturbance recorder for this investigation. It was also configured for overcurrent and earth fault protection in order to send a trip signal to the Circuit Breaker (CB) within the draft model when those conditions occur.

The basic configuration parameters and the steps involved in configuring the SEL-451 IED using AcSElerator QuickSet software environment are as follow:

Frequency Setting

Global > General Global Settings (Set Nominal System Frequency = 60Hz)

Current Transformer Ratio

Group 1 > Set 1 > Line Configuration (Set Current Transformer Ratio = 800)

Pickup Setting

Group 1 > Set 1 > Relay Configuration > Phase Instantaneous/Definite-Time Overcurrent 1 (Set 50P1P=1.09 and 67PID = 0)

Pickup Setting (Earth fault)

Group 1 > Set 1 > Relay Configuration > Residual Ground Instantaneous Overcurrent 1 (Set 50G1P= 0.16 and 67PID = 0)

Trip Logic

Group 1 > Set 1 > Relay Configuration > Trip Logic (Set to '50P1 OR 50G1')

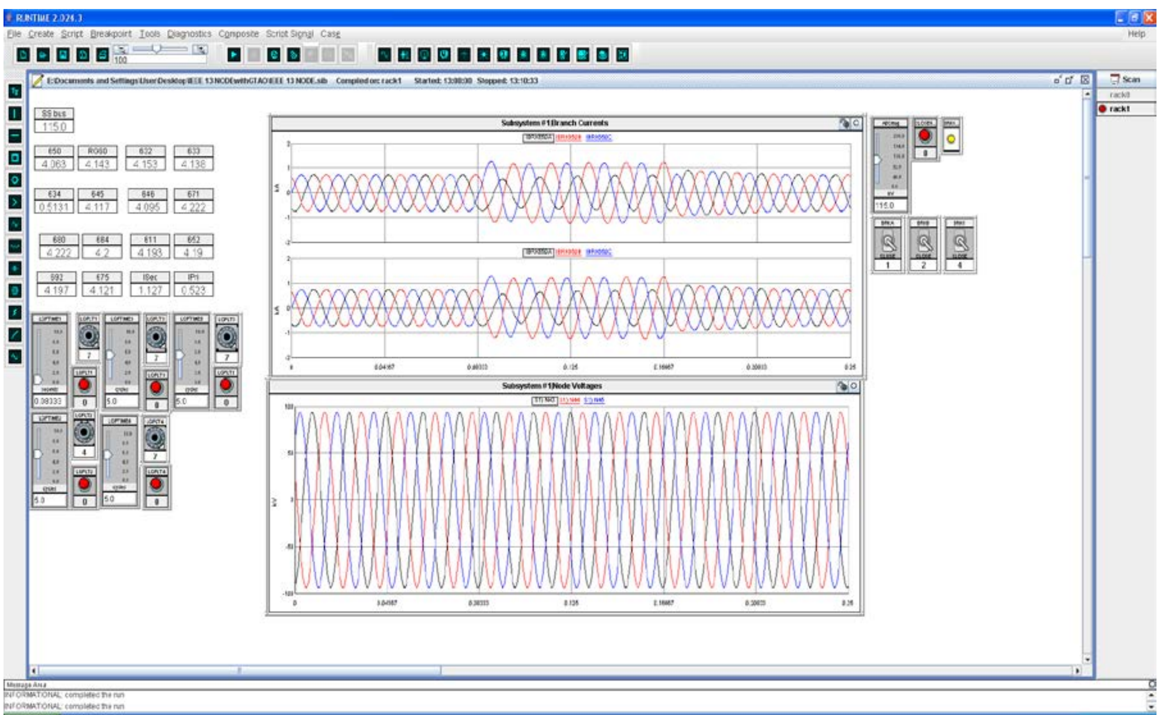


Figure 7.7: **Controls palette in RSCAD runtime**

Trigger Equation

Group 1 > Set 1 > Relay Configuration > Trip Logic (Set Event Report Trigger Equation = R_TRIG 50P1 OR R_TRIG 50G1)

Trip Output

Outputs > Main Board (Set OUT 101 Main Board Output =T3P1 OR TRIP)

Breaker Status Report

Breaker Monitor > Breaker 1 (Set Breaker 1 Monitoring = 'Y' and 'N/O Contact Input = IN101')

Event Reporting

The event reporting configuration used for this research is shown in Figure 7.8. The configuration is elaborated in Section 7.7.

7.7 Disturbance Recording

This sub-section presents the simulation results using the RTDS and event files retrieved from the external 'hardware-in-the-loop' IED which in this case was the SEL-451.

The IED was configured for 8 kHz sampling frequency. 8 kHz was chosen out of the available options (1kHz, 2kHz, 4kHz, and 8kHz) since typical disturbance recorders have sampling frequencies around this value. The length of the event report was set to 0.25 seconds (15 cycles), while the pre-fault length was 0.08 seconds corresponding to 5 cycles. Figure 7.8 shows the configuration of this function.

Figure 7.8: **Configuration of the IED for disturbance recording**

7.8 Post-Simulation Operations

The SEL-451 provides two types of event data capture. These are: high resolution oscillography and event report oscillography. The former uses raw sample per second data, while the latter uses filtered samples per cycle data.

The following steps were done to download the event files from the relay.

- From the AcSElerator QuickSet toolbar, click ***View Event History***
- Select the events needed and click the ***Get Selected Event*** button
- Choose the file type. High resolution oscillography is obtained by selecting ***Binary COMTRADE file***. Event report oscillography is indicated by ***Compressed Events (CEV) 4 samples/cyc or CEV 8 samples/cyc***
- Save file(s) when download is completed
- View the events by using the AcSELerator Analytical Assistant software

Figure 7.9 shows an oscillograph of an event obtained using the AcSELerator Analytical Assistant software. The figure gives the report summary, waveform plots of the three phase currents, and the phasor diagram for the measured quantities.

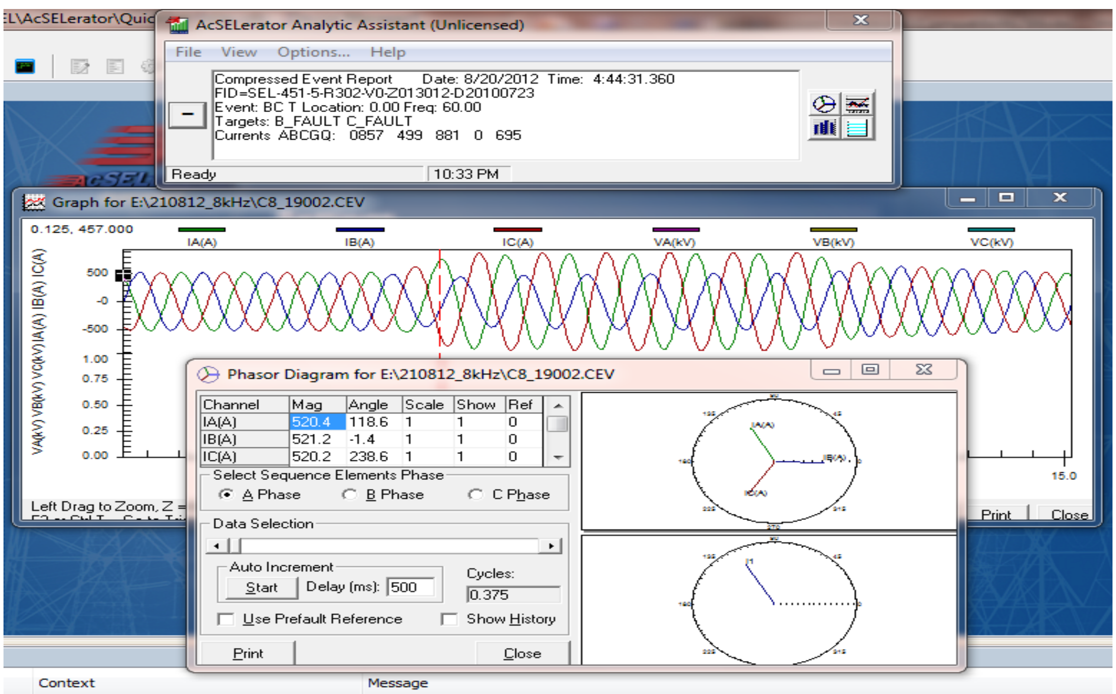


Figure 7.9: **SEL-451 event viewing using AcSELerator Analytical Assistant**

Using event report oscillography of CEV 4 samples/cyc gives a sampling frequency of 240Hz for a 60Hz nominal frequency, while a CEV 8 samples/cyc gives a sampling frequency of 480Hz for a 60Hz nominal frequency. The binary COMTRADE file option was used in this thesis to retrieve disturbance records from the relay. These records are sampled based on the sampling frequency pre-configured in the relay.

The RSCAD runtime waveform plots and the retrieved event files from the SEL-451 were saved as COMTRADE files. Software sub-routines written in MATLAB were used to decode the data stored in COMTRADE format as defined in IEEE C37.111-1999. This involves the opening of two files, one containing the configuration (.cfg) information and the other containing the data (.dat) information. The necessary data is extracted and written to MATLAB workspace. Afterwards, the HFDD method can be used to perform diagnosis.

`

7.9 Results and Discussion

7.9.1 Results for Fault Detection and Classification

The results obtained at a location on the main feeder, lateral, and at a sub-lateral are shown for the verification of the fault detection and classification tasks of the HFDD method.

Table 7.3 shows the values of wavelet entropy obtained for different levels of decomposition from the SEL-451 disturbance records and the RTDS waveform plot at R_f = 0.1 along Line 650-632. Similarly, Table 7.4 shows the values of wavelet entropy obtained for various fault types along Line 692-675.

Table 7.3: **Wavelet entropy obtained from disturbance records for faults at Line 650-632**

		SEL-451 Records			RTDS Records		
Decomposition	**Fault type**	r_a	r_b	r_c	r_a	r_b	r_c
Level 1	AB	**5.9535**	**6.0098**	6.1074	**4.2145**	**3.985**	5.4102
	BC	6.1046	**6.0440**	**6.0286**	5.6385	**5.5205**	**4.6871**
	ABC	**6.0041**	**5.9002**	**6.0013**	**5.1334**	**3.6449**	**2.5638**
Level 4	AB	**1.5164**	**1.4304**	1.7739	**1.8448**	**1.5907**	1.2198
	BC	1.5440	**1.3397**	**1.7394**	**1.8360**	**1.4173**	**1.4347**
	ABC	**1.4359**	**1.6663**	**1.3252**	**1.4520**	**1.6602**	**1.3345**
Level 5	AB	**5.4936**	**5.5062**	5.5492	**5.4822**	**5.4229**	5.5616
	BC	5.5247	**5.5372**	**5.4777**	5.5878	**5.5341**	**5.4109**
	ABC	**5.4041**	**5.4071**	**5.1928**	**5.3773**	**5.4289**	**5.1940**
Level 6	AB	**4.8789**	**4.2511**	**1.3571**	**3.9260**	**4.7318**	1.4278
	BC	4.7578	**4.6226**	**0.2195**	4.5940	**4.8164**	**1.3777**
	ABC	**4.7174**	**3.5776**	**1.3949**	**4.6598**	**3.6742**	**1.4078**

Table 7.4: **Wavelet entropy and wavelet entropy per unit obtained from disturbance records at Line 692-675**

	Entropy			Entropy per unit		
	Level 5			Level 5		
Fault type	r_{a5}	r_{b5}	r_{c5}	λ_{a5}	λ_{b5}	λ_{c5}
A-G	**1.5170**	1.5530	1.6150	**0.3240**	0.3320	0.3450
AB	**1.5170**	**1.5550**	1.5850	**0.3260**	**0.3340**	0.3400
BC	1.1820	**0.9970**	0.7500	0.4030	**0.3400**	**0.2560**
ABG	**1.8850**	**1.5950**	2.1860	**0.3330**	**0.2810**	0.3860
ABC	**1.5230**	**1.5800**	**1.5270**	**0.3290**	**0.3410**	**0.3300**

Furthermore, Tables 7.5-7.6 present the result obtained for varying load angles for B-G and CA respectively along the main feeder. Table 7.7 shows the result for a A-G fault for the main feeder, lateral, and sub-lateral respectively.

Table 7.5: **Wavelet entropy and wavelet entropy per unit for B-G for main feeder at location 650-632**

	Entropy			Entropy per unit		
	Level 5			Level 5		
Load angle/(^{o})	r_{a5}	r_{b5}	r_{c5}	λ_{a5}	λ_{b5}	λ_{c5}
15	5.8805	**5.5917**	5.8769	0.3390	**0.3220**	0.3390
30	5.8395	**5.7735**	5.7082	0.3370	**0.3330**	0.3300
60	5.8534	**5.6559**	5.7814	0.3390	**0.3270**	0.3340
90	5.8404	**5.6791**	5.7656	0.3380	**0.3290**	0.3340

Table 7.6: **Entropy values for CA for main feeder at location 650-632**

	Entropy			Entropy per unit		
	Level 5			Level 5		
Load angle/(^{o})	r_{a5}	r_{b5}	r_{c5}	λ_{a5}	λ_{b5}	λ_{c5}
15	**5.7489**	5.8027	**5.7711**	**0.3320**	0.3350	**0.3330**
30	**5.8238**	5.8480	**5.6844**	**0.3360**	0.3370	**0.3280**
60	5.8017	**5.7706**	**5.7420**	**0.3350**	0.3330	**0.3320**
90	**5.7659**	5.8161	**5.7359**	**0.3330**	0.3360	**0.3310**

Table 7.7: **Entropy values for A-G for main feeder at various location**

	Entropy			Entropy per unit		
	Level 5			Level 5		
Location	r_{a5}	r_{b5}	r_{c5}	λ_{a5}	λ_{b5}	λ_{c5}
Main feeder	5.7010	5.8530	5.8320	0.3280	0.3370	0.3350
Sub-lateral 684-611	5.5518	5.6677	5.6011	0.3301	0.3369	0.3329
Sub-lateral 684-652	4.8250	4.9230	4.9050	0.3290	0.3360	0.3350
Lateral 692-675	5.4849	5.5425	5.6739	0.3284	0.3319	0.3397

7.9.2 Results for Fault Section Identification using ANN

In the design of the Artificial Neural Network (ANN) for the IEEE 13 node test feeder fault section identification task, the procedure used in Section 5.6 was followed.

Four neural network classifiers corresponding to the fault classes (SLG, 2Ph., 2Ph.-G, and 3Ph.) are needed. However, the results presented here are for single phase to ground faults only.

The wavelet energy entropies computed from the batch mode hardware-in-the-loop simulation were collected and organized for training using Microsoft Excel. The performance criteria used are the size of the neural network, the learning process, confusion matrix, Receiver Operating Characteristics (ROC) plot, and classification accuracy. Several architectures were investigated. The datasets selected for the neural network training are normalized to the range [-1, 1] before the training and testing by using equation (5.13).

The activation function used was the *tansig* for the hidden and output layers respectively. Also, the number of epochs was determined by experimentation. The number of epoch for the training was set between the range of 1000 and 2000.

Table 7.8 gives a summary of the parameters used to generate the training and testing datasets. Table 7.9 gives the Fault Section Identification (FSI) association table representing the fault sections classes.

Table 7.8: Parameters for training and testing the fault section identification ANNs

Parameter	**Main Feeder**
Training	
Fault Section	Mainfeeder, L.632A, L.632 B, L.671A, L.671B, L.684A, L.684B
Fault Resistance (Ω)	0.1, 1.0, 5.0, 10.0, 20.0, 100.0
Testing	
Fault Section	Mainfeeder, L.632A, L.632 B, L.671A, L.671B, L.684A, L.684B
Fault Resistance (Ω)	2.5, 7.5, 15.0,

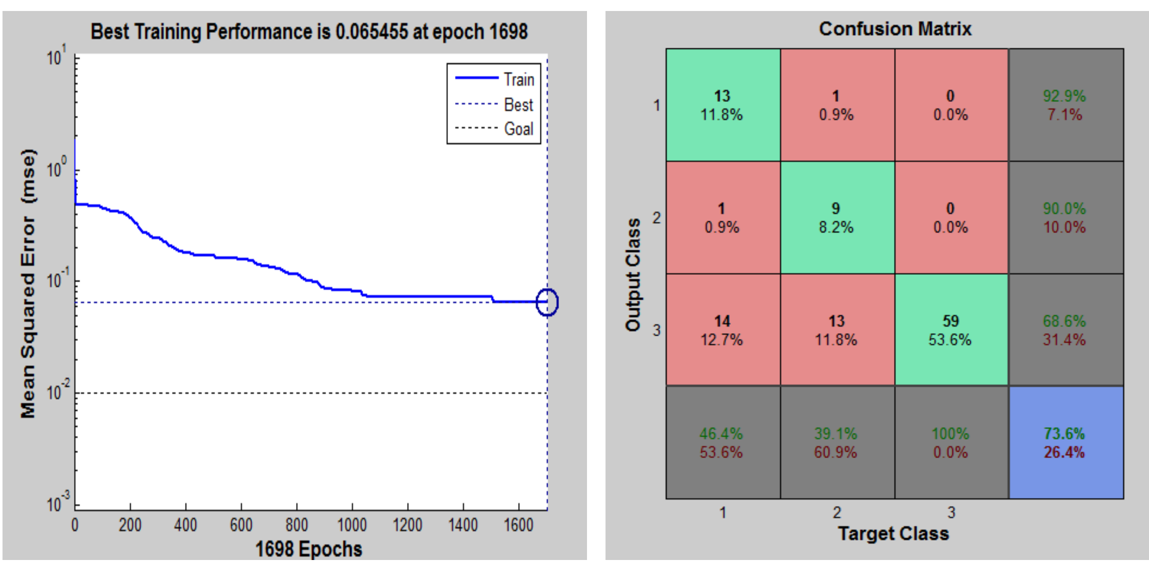
Network growing was used in training the neural network. The number of neurons in the hidden layer was decided by experimentation which involves different network configurations. The process started with the lowest number of neurons in the hidden layer and was increased until a suitable network having satisfactory performance is established. Preference was given to the network with the smallest size in terms of the lowest number of hidden layer weights.

Table 7.9: **FSI association table representing the fault section class**

Nr.	Section	Classification Class		
1	Main feeder	0	0	1
2	L.632A	0	1	0
3	L.632B	0	1	1
4	L.671A	1	0	0
5	L.684A	1	0	0
6	L.684B	1	1	0
7	L.671B	1	1	1

The first network had a single hidden layer with 5 neurons. The number of input and output neurons was 3 respectively. Target MSE of 0.01 and 0.001 were experimented with, while the maximum number of epochs was 2000. The training algorithm used was Levenberg-Marquardt (L-M) algorithm. The activation functions in the hidden and output layers were *tansig* respectively.

The performance plot in Figure 7.10(a) shows that the training converged slowly and the network did not show any sign of improvement after 1698 iterations.

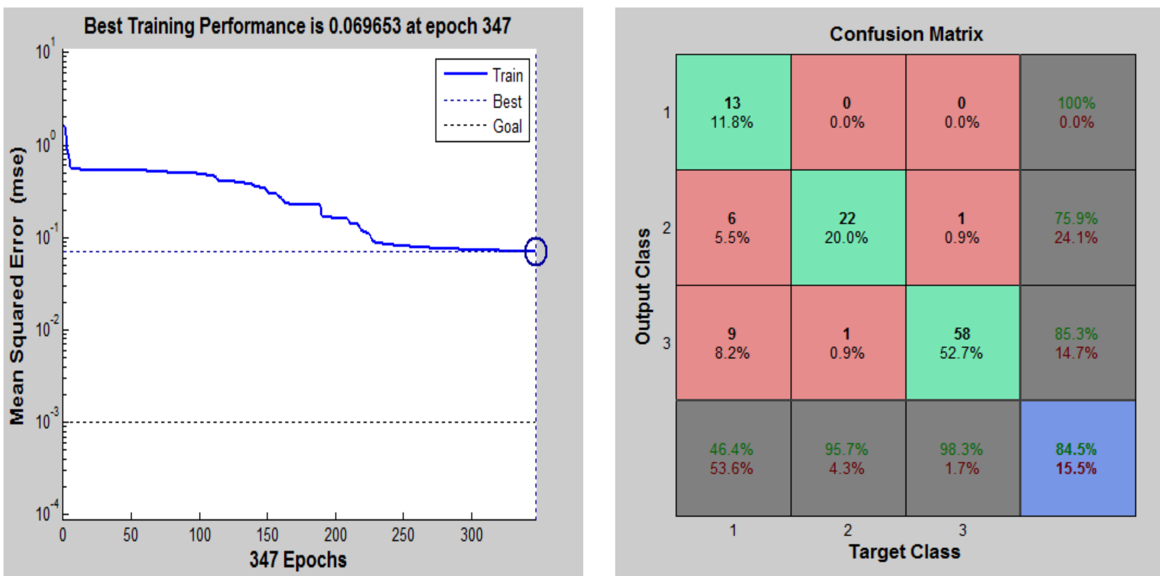


(a) **(b)**

Figure 7.10: (a) **Training performance curve for 3-5-3; and** **(b)** **Confusion matrix plot showing the training state for 3-5-3**

Also, when tested with new dataset, the network was not able to generalize and accurately classify the fault section. The confusion matrix (Figure 7.10(b)) of 73.6% was obtained. In addition, testing with new (untrained) dataset of 39 fault cases gave an accuracy of 29 correct classification out of 39 (74.36%).

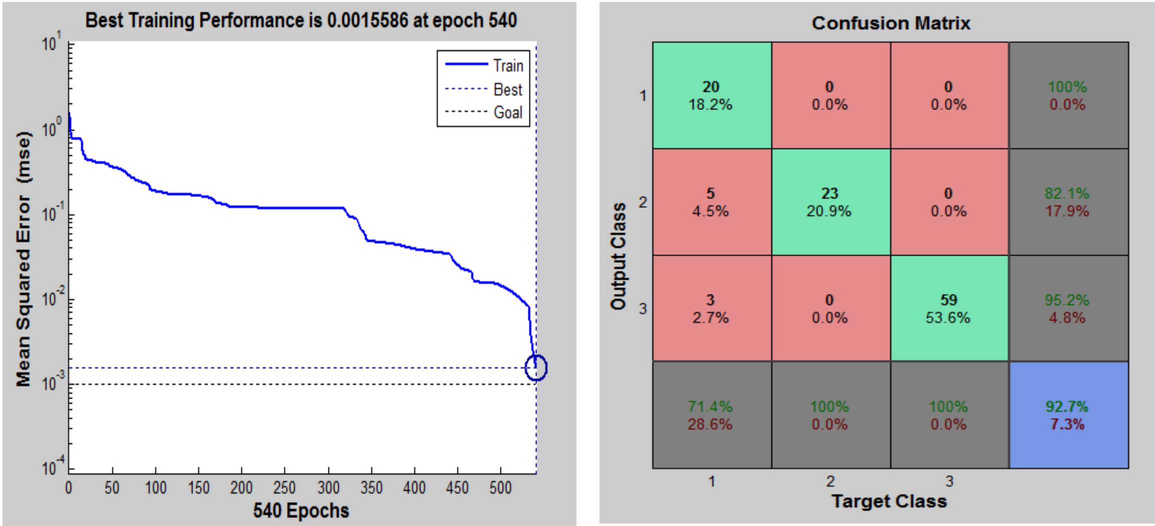
Further increasing the number of hidden layer neurons by a step of 5 upto 25 neurons did not improve the performance of the trained network. The result obtained using 25 neurons in the hidden layer is given in Figure 7.11. A performance goal of 0.069653 was achieved after 347 iterations and did not show any sign of improvement.
Also, a confusion matrix of 84.5% was obtained. Testing the network using untrained dataset of 39 fault cases resulted to an accuracy of 32 correct classification out of 39 (82.05%).



(a) **(b)**

Figure 7.11: (a) Training performance curve for 3-25-3; and (b) Confusion matrix plot showing the training state for 3-25-3

The number of hidden layer was increased to two and a network with 10 neurons in each hidden layer was found to give the best performance as shown in Figure 7.12. A performance goal of 0.001 was achieved after 540 iterations. Also, a confusion matrix of 92.7% was obtained. Testing the network using untrained dataset of 39 fault cases resulted to an accuracy of 38 correct classification out of 39 (97.44%).



(a) **(b)**

Figure 7.12: (a) **Training performance curve for 3-10-10-3; and (b) Confusion matrix plot showing the training state for 3-10-10-3**

7.9.3 Discussion

In the processing of the disturbance records from the relay, some of the COMTRADE files were not readable. An alternative pursued was the use of The Output Processor (TOP) software by Electrotek Concepts, Inc. This software was used to convert the IEEE COMTRADE binary format to IEEE COMTRADE ASCII. The latter was then read into MATLAB by using the function for COMTRADE file reader.

The wavelet entropy results obtained from the disturbance records of the IED and the RTDS showed different values. Nevertheless, they still exhibited similar trends. This difference could be as a result of the logic that governs the recording trigger within the IED. Generally, DWT levels-4 and -5 gave good WEE values for the various fault sections, fault types, and fault resistances. For the event files retrieved from the relay with a sampling frequency of 8kHz, levels-4 and -5 correspond to 250Hz and 125Hz respectively. The same applies to the RTDS disturbance waveform with a sampling frequency of 8kHz.

The sampling frequency of the RTDS on the other hand, can be determined based on the simulation time step and the number of points being plotted in runtime. In this case, the time step was changed to 125μS, while the sampling was done at every point. This was done in order to arrive at a sampling frequency of 8kHz.

The faulted phases showed lower values of wavelet entropies and wavelet entropy per unit obtained from level-5 respectively. During the closed-loop simulation of the RTDS with the relay, the faulted phases for some faults were incorrectly identified by the relay. However, the HFDD method was accurate in its diagnosis. For instance, the event in Figure 7.9 showed a BC fault on the relay, but was correctly identified as a CA fault by the HFDD method.

For the fault section identification task, artificial neural network of various architectures were investigated for the classification of the faulty section. Artificial neural networks are composed of neurons, weights, and transfer functions. Feed-forward multilayer neural networks using Levenberg-Marquardt learning algorithm with *tansig* activation function have been shown to give better performance for fault section identification in Chapter Six. Feed-forward multilayer neural networks are known for their effectiveness and ease of training.

For a single hidden layer, the number of neurons was varied from 5 to 25 with a step of 5. The convergence characteristics during training and accuracy when tested with untrained dataset were observed. From the analysis done on the neural network architectures, it was found that increasing the number of neurons beyond 25 did not improve the performance of the trained network or the accuracy of the test dataset.

Figure 7.10(a) shows the convergence characteristics for a 3-5-3 neural network architecture. The network was able to cross a MSE of 0.1 after about 900 iterations and did not show any sign of improvement until the training was stopped after 1698 iterations. The analysis of the trained network showed that 29 fault cases out of 110 fault cases used in training the network were misclassified.

Figure 7.11(a) shows the convergence characteristics of the network training for 3-25-3 neural network architecture. The network was able to cross a MSE of 0.1 after about 220 iterations and did not show any sign of improvement until the training was stopped after 347 iterations. The analysis of the training dataset showed that 17 fault cases out of 110 fault cases used in training the network were misclassified.

Figure 7.12(a) shows the performance curve for two hidden layers with a 3-10-10-3 network architecture. The network converged after about 540 iterations with a MSE of 0.001. The analysis of the training dataset showed that 8 fault cases out of 110 fault cases used in training the network were misclassified. The best performance was also obtained with this network when tested with untrained dataset of 39 fault cases. An accuracy of 38 correct classifications out of 39 (97.44%) fault cases was obtained.

The above results demonstrate the capability of wavelet entropy and wavelet entropy per unit for distribution network fault detection, fault classification, and fault section identification. Appropriate thresholds need to be chosen for the fault detection and fault classification tasks, while for fault section identification, the procedure would be to generate enough data for the training of the neural networks as demonstrated in Chapters Five and Six respectively.

7.10 Conclusion

The HFDD method was implemented in real-time using the RTDS and RSCAD software. The IEEE 13 Node Radial Test Feeder was modelled in RSCAD software. Simulations to obtain event records for the HFDD method were conducted on the study network under varying scenarios. Event waveforms from the runtime of the simulation in RSCAD software and also from the fault event records of the IED were retrieved and processed using the designed HFDD method for the fault detection, fault classification, and fault section identification tasks. Results show that the method is robust, scalable, and accurate.

Preliminary studies carried out have demonstrated the capability of the proposed HFDD method. This implies that the developed algorithms can be applied to any power system network even where the operating characteristics are frequently changing. However, the thresholds and rules for the algorithm would change and the neural networks would need to be retrained to cater for changes in the operating scenario.

Chapter 8 presents the deliverables of the thesis and the areas where the developed method can be applied. Also, recommendations for future work and publications associated with the thesis are given therein.

CHAPTER EIGHT

CONCLUSION AND RECOMMENDATIONS

8.1 Introduction

This thesis has studied the use of Discrete Wavelet Transform (DWT) based wavelet energy spectrum entropy, decision-taking rules, and artificial neural networks as a method for fault detection, fault classification, fault section identification, and fault location on a typical distribution network. The methods employed made use of entropy/entropy per unit values of DWT decomposed three phase and zero sequence currents as inputs to the rule-based decision-making algorithm and neural networks. Various possible kinds of faults namely single line-ground, line-line, double line-ground, and three phase faults have been taken into consideration in this work and individual ANNs for fault section identification and fault location have been proposed for each of these fault classes.

All the neural networks investigated in the thesis belong to the back-propagation Multi-Layer Perceptron (MLP) neural network architecture. A fault location scheme for distribution system, right from the verification of a fault event on the line to the fault classification stage, to the fault section identification stage, and finally to the fault location stage has been devised successfully by using a hybrid method comprising of wavelet energy spectrum entropy, algorithm rules, and artificial neural networks.

The simulation results obtained proved that satisfactory performance has been achieved by the proposed method. As further illustrated, depending on the application of the neural network and the size of the training data set, the size of the ANN (the number of hidden layers and number of neurons per hidden layer) varied. The importance of choosing the most appropriate ANN configuration in order to get the best performance from the network, has been stressed upon in this work. The sampling frequency adopted for the current waveforms was 7.68 kHz.

As a proof of concept, the IEEE 13 Node Test Feeder was modelled in RSCAD software and simulations with different operating scenarios were carried out. The results of these scenarios from RSCAD runtime plots and disturbance records from an external IED (SEL-451) were exported to the developed HFDD method for fault detection verification and fault classification tasks respectively.

The inputs to the developed method are based on current measurement. Thus, there is no need for new equipment since all distribution networks have Current Transformers (CTs). Results obtained from several operating conditions/scenarios simulated show that this method is efficient, accurate, and robust. Also, the computational burden was very low because of the use of DWT. In addition, the HFDD method showed good performance even for fault resistance greater than 0Ω. Thus, further improvements to the algorithm could possibly lead to real-world application.

8.2 Deliverables

The need for a Fault Detection and Diagnosis (FDD) method has been emphasized in Chapter One. Therefore, the deliverables of this project are based on the aim and objectives highlighted therein.

8.2.1 Modelling and Simulation

- Modelling of the IEEE 34 node test feeder in DIgSILENT PowerFactory (DPF).
- Integration of Distributed Generators (DGs) and line extension of the IEEE 34 node test feeder in DIgSILENT PowerFactory (DPF) for the purpose of introducing disturbance in the network.
- Simulation of various operating and fault scenarios in the IEEE 34 node test feeder in DIgSILENT PowerFactory (DPF) for the purpose of generating data for steady state and dynamic conditions during faults.

8.2.2 Signal Processing using Discrete Wavelet Transform and calculations

- Investigations of wavelet transform mother wavelets.
- Selection of the best mother wavelet.
- Investigation of the best level of wavelet decomposition to use.
- Feature extraction and selection based on wavelet entropy.
- Feature selection based on the proposed wavelet entropy per unit formulation.

8.2.3 Development of Rule-Based Algorithms

- Development of fault detection algorithm based on decision-taking rules obtained empirically.
- Development of a fault classification algorithm based on decision-taking rules.

8.2.4 Neural Network Based Algorithms

- Development of neural network based fault section identification algorithms.
- Development of neural network based fault location algorithms.

8.2.5 Development of a Hybrid Fault Detection and Diagnosis (HFDD) Method

- Integration of the various algorithms.
- Performance testing of the HFDD method.
- Development of software in MATLAB for all the above-mentioned deliverables as shown in Table 8.1.

Table 8.1: **MATLAB script file developed**

MATLAB script file	Function
dwt_exp.m	Wavelet decomposition using different mother wavelets
splash.m	Splash screen for the HFDD method
faultmain.m	Fault detection algorithm
faulty.m	Fault classification algorithm for 1Ph.-G fault
faulty2ph.m	Fault classification algorithm for 2Ph. fault
faulty2phg.m	Fault classification algorithm for 2Ph.G and 3Ph. fault
nnfsitraining	Neural network training for fault section identification using early stopping method
nnfsitraining_reg.m	Neural network training for fault section identification using hold-out and regularization methods
Nnfltraining.m	Neural network training for fault location using early stopping method
nnfltraining_reg.m	Neural network training for fault location using hold-out and regularization methods
nntesting_fsi.m	Neural network testing for fault section identification
nntesting_fl.m	Neural network for fault location
(fsi1ph/fsi2ph/fsi2phg/fsi3ph).m	Algorithms for fault section identification using neural network
(fl1ph/flph/fl2phg/fl3ph).m	Algorithms for fault location using neural network
read_comtrade_rtdswave.m	Reading and processing of COMTRADE file from RTDS runtime environment
read_comtrade_sel451events.m	Reading and processing of COMTRADE file from SEL-451 IED
wavent1.m	computes wavelet entropy using level-1 detail coefficients
wavent5.m	computes wavelet entropy using level-5 detail coefficients
wee_visualplot.m	Visualization plots of entropy per unit used for fault classification
waveletfamilyplotting.m	Plotting of wavelet functions
activation_functions.m	Plotting of neural network activation functions

8.2.6 Real-Time Testing

- Modelling and simulations of the IEEE 13 node test feeder in RSCAD software for the purpose of generating data for steady state and dynamic conditions during faults in order to validate the HFDD method.
- Hardware-in-the-loop connection using the Real-Time Digital Simulator (RTDS) and an external Intelligent Electronic Device (IED) for event recording.
- Event record retrieval from the external IED (SEL-451) for further processing with the HFDD method.
- Validation of the HFDD method using real-time data from the RTDS.

8.3 Application of the HFDD Method

In reality, faults on distribution lines are inevitable. Thus, there is the need to have in place a contingency plan to detect, diagnose, and mitigate the effects of such faults.

The proposed Hybrid Fault Detection and Diagnosis (HFDD) method can be software oriented running on a workstation computer, or implemented as stand-alone equipment.

8.3.1 Practical Application in Distribution Networks

The HFDD method can be applied to assist control centre dispatchers and engineers in the following tasks:

- Network monitoring and checking for threshold violations for fault detection.
- Verification of fault events.
- Validation of protection relay operation and protection schemes.
- Identification of fault type and faulted phase(s).
- Fault section identification.
- Narrowing the search for the faulted line through fault location.
- Reduction of the cost of callout of personnel and man-hours spent in locating faults.
- Reduction of the mean down-time and the attendant loss of revenue.
- Improving reliability as a result of the reduction in down-time.

8.3.2 Academic/Research Application

- Students can utilise the IEEE 34 node test feeder modelled in DIgSILENT PowerFactory as a benchmark for other power system models/components and also for future research work in distribution systems.
- Simulations using the model above can help students understand the dynamics of power system simulations and fault analysis.
- The discrete wavelet transform algorithm used for feature extraction and selection can help students understand the application of signal processing using wavelet transform and its application in power systems.
- The developed method encompasses algorithms for fault detection, fault classification, fault section identification, and fault location using decision-taking rules and neural networks, and would be useful as a teaching aid to students on the concept of fault detection and diagnosis.

8.4 Future Work

In the future, it would be beneficial to have an integrated method with the capability of running alongside a power system simulation package in real-time or 'near real-time'.

Also, automated IED event file retrieval and report polling can be developed to analyse disturbance records on the fly.

Furthermore, an automated fault analysis package with web service can be carried out in future so that engineers/system operators can have access to data/reports even when away from their workstations. This would also provide enterprise access to relevant departments and personnel irrespective of their location.

This work can further be implemented in hardware. In doing this, the hardware should be designed with the possibility of having a modification process for the algorithm i.e. provision should be made for the modification of the algorithm rules used in fault detection and classification, and retraining of the NN for the FSI and FL tasks respectively. This requires that the last versions of rules should be archived and saved before modification, and incorporated with roll-back option should the need arise to go to previous settings.

8.5 Publications Related to the Thesis

a. Adewole, A.C. and Tzoneva, R. (2011). 'A Review of Methodologies for Fault Detection and Location in Distribution Power Networks'. *International Review on Modelling and Simulation (IREMOS)*, Vol.4 No.6, pp. 3214-3231.

b. Adewole, A.C. and Tzoneva, R. (2012). 'A Method for Distribution Network Fault Detection and Classification based on Wavelet Energy Spectrum Entropy'. *PAC World Conference 2012,* Budapest, 25-28 June 2012, pp. 1-22.

c. Adewole, A.C. and Tzoneva, R. (2012). 'Fault Detection and Classification in a Distribution Network Integrated with Distributed Generators. *IEEE PES PowerAfrica 2012 Conference and Exhibition,* Johannesburg, South Africa, 9-13 July, 2012. pp. 1-8.

d. Adewole, A.C., Tzoneva, R., Behardien, S. 2012. "Wavelet Entropy Algorithm for Distribution Network Fault Detection and Classification", submitted to *Turkish Journal of Electrical Engineering and Computer Sciences*.

e. Adewole, A.C., Tzoneva, R., Behardien, S. 2012. "Distribution Network Fault Section Identification and Fault Location using Wavelet Entropy and Neural Networks", submitted to *Advances in Electrical and Computer Engineering Journal*.

Adewole, A.C. and Tzoneva, R. 2011. A Review of Methodologies for Fault Detection and Location in Distribution Power Networks. *International Review on Modelling and Simulation (IREMOS)*, vol.4 no.6, pp. 3214-3231.

Adewole, A.C. and Tzoneva, R. 2012a. A Method for Distribution Network Fault Detection and Classification based on Wavelet Energy Spectrum Entropy. *PAC World Conference 2012,* Budapest, 25 - 28 June 2012, pp. 1-22.

Adewole, A.C., Tzoneva, R. 2012b. Fault Detection and Classification in a Distribution Network Integrated with Distributed Generators. *IEEE PES PowerAfrica 2012 Conference and Exhibition*, Johannesburg, South Africa, 9-13 July 2012, pp. 1-8.

Aggarwal, R.K., Johns, A.T., Song, Y.H., Dunn, R.W., and Fitton, D.S. 1994. Neural-network based adaptive single-pole autoreclosure technique for EHV transmission systems. *IEE Proceedings on Generation, Transmission and Distribution,* vol. 141, Issue 2, pp. 155 – 160.

Aggarwal, R. and Johns, A. 1997. Artificial Intelligence Techniques in Power Systems: AI for protection systems. *IEE Power Engineering Series*, vol. 22, pp. 109-142.

Akorede, M.F., Katende, J. 2010. Wavelet Transform Based Algorithm for High-Impedance Faults Detection in Distribution Feeders. *Proceedings of European Journal of Scientific Research, vol.41, n.2,* pp. 238-248.

Alsberg, B.K., Woodward, A.M., Kell, D.B. 1997. An introduction to wavelet transforms for chemometricians: A time-frequency approach. *Chemometrics and Intelligent Laboratory Systems,* vol. 37, pp.215-239.

Al-Shaher, M., Sabra, M.M., Saleh, A.S. 2003. Fault Location in Multi-Ring Distribution Network Using Artificial Neural Network. *Electric Power Systems Research, vol. 64 n. 2,* pp. 87-92.

Amorim, H. P. Cepe1 and L. Huais, 2004. Faults location in Transmission Lines through Neural Networks. *IEEE/PES Transmission and Distribution Conference & Exposition: Latin America*. pp. 691 – 695.

Anderson, J.A., and Rosenfeld, E., eds. (1988). *Neurocomputing: Foundations of Research.* Cambridge, MA: The MIT Press.

Aslan, Y., Türe, S. 2011. Location of Faults in Power Distribution Laterals using Superimposed Components and Programmable Logic Controllers. *Electrical Power and Energy Systems, vol. 33,* pp. 1003–1011.

Assef, Y., Chaari, O., Meunier, M. 1996. Classification of power distribution system fault currents using wavelets associated to artificial neural networks. *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pp. 421-424.

Aucoin, B.M. and Russell, B. D. 1985. Detection of distribution high impedance faults using burst noise signals near 60 Hz. *IEEE Transactions on Power Apparatus and System*, vol. PAS-104, no. 6, pp. 1643-1650.

Aygen, Z.E., Seker, S., Bagriyanik, M., Bagriyanik, F.G., Ayaz, E. 1999. Fault section estimation n electrical power systems using artificial neural network approach. *IEEE Transmission and Distribution Conference*, vol. 2, pp. 466-469.

Baqui, I., Zamora, I., Mazón, J., Buigues, G., 2011. High Impedance Fault Detection Methodology Using Wavelet Transform and Artificial Neural Networks. *Electric Power Systems Research, vol. 81,* pp. 1325–1333.

Barker, P. de Mello, R.W. 2000. Determining the impact of distributed generation on power systems: part 1 – Radial power systems. *In Proc. IEEE Power Eng. Soc. Summer Meeting,* vol. 1, pp. 1645–1658.

Barros J., Perez E., Pigazo A. 2003. Real Time System For Identification Of Power Quality Disturbances. *17th International Conference on Electricity Distribution, CIRED, Barcelona*, 12-15 May 2003.

Bekker, J.G., Keller, P.G. 2008. Enhancement of an Expert System Philosophy for Automatic Fault Analysis. *In Proceedings of 11th Annual Georgia Tech Fault and Disturbance Analysis Conference*, Atlanta, Georgia, USA.

Bhowmik, P.S., Purkait, P., Bhattacharya, K. 2008. A novel wavelet assisted neural network for transmission line fault analysis. *Annual IEEE India Conference, INDICON 2008,* vol. 1, pp. 223-228.

Bishop, C.M., 1996. *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Bollen, M. H.J., Gu, I. Y.H., Santoso, S., McGranaghan, M.F., Crossley, P.A., Ribeiro, M.V., Ribeiro. P.F. 2009. Bridging the Gap between Signal and Power: Assessing power system quality using signal processing techniques: *IEEE Signal processing Magazine*, 12 July, 2009, pp. 12-31.

Borghetti, A., Corsi, S., Nucci, C.A., Paolone, M., Peretto, L., Tinarelli, R. 2006. On the use of continuous-wavelet transform for fault location in distribution power systems. *Electrical Power and Energy Systems,* vol. 28, pp. 608–617.

Borghetti, A., Bosetti, M., Di Silvestro, M., Nucci, C.A., Paolone, M. 2007. Continuous-Wavelet Transform for Fault Location in Distribution Power Networks: Definition of Mother Wavelets Inferred From Fault Originated Transients. *International Conference on Power Systems Transients (IPST'07) in Lyon, France*, June 4-7 2007, pp. 1-9.

Boutsika, T.N., Papathanassiou, S.A. 2008. Short-circuit calculations in networks with distributed generation. *Electric Power Systems Research*, vol. 78, pp. 1181–1191.

Butler, K.L., Momoh, J. 1993. Detection and Classification of Line Faults on Power Distribution Systems Using Neural Networks. *Proceedings of the 36th Midwest Symposium on Circuits and Systems*, *vol. 1,* 16-18 August 1993, pp. 368-371.

Butler, K.L., Momoh, J.A., Sobajic, D.J. 1997. Field studies using a neural-net based approach for fault diagnosis in distribution networks. *IEE Proc. Gener. Transm, Distrib,* vol. 144, No. 5, September, 1997.

Butler-Purry, K.L., Cardoso, J. 2008. Characterization of Underground Cable Incipient Behavior Using Time-Frequency Multi-Resolution Analysis and Artificial Neural Networks. *Proceedings of*

IEEE Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 20-24 July 2008, pp. 1-11, Pittsburgh, USA.

Campoccia, A., Di Silvestre, M.L., Incontreraa, I., Riva Sanseverinoa, E., Spotob, G. 2010. An Efficient Diagnostic Technique for Distribution Systems Based on Under Fault Voltages and Currents. *Electric Power Systems Research vol. 80,* pp. 1205-1214.

Chan, D.T.W., Lu, C.Z. 2001. Distribution System Fault Identification by Mapping of Characteristic Vectors. *Electric Power Systems Research vol. 57,* pp. 15–23.

Chanda, D., Kishore, N.K., and Sinha, A.K. 2003. A wavelet multiresolution analysis for location of faults on transmission lines. *Electric Power and Energy Systems*, 25. Pp. 59-69.

Choppin, B., 2004. *Artificial Intelligence Illuminated*. Sudbury, Massachusetts: Jones and Bartlett Publishers.

Cormane, J.A., Vargas, H.R., Ordóñez, G., Carrillo, G. 2006. Fault Location in Distribution Systems by Means of a Statistical Model. *Proceeding of IEEE PES Transmission and Distribution Conference and Exposition Latin America,* 2006, pp. 1-7, Venezuela.

Chunju, F., Li, K.K., Chan, W.L., Weiyong, Y., Zhaoning, Z. 2007. Application of Wavelet Fuzzy Neural Network in Locating Single Line To Ground Fault (SLG) in Distribution Lines. *Electric Power and Energy System vol. 29,* pp. 497-503.

Das, R., Saha, M.M., Verho, P., Novosel, D. 2003. Fault Location Techniques For Distribution Systems. *CIRED 17th International Conference on Electricity Distribution, paper no. 49-1 session 3,* 12-15 May 2003, pp. 1-6, Barcelona, Spain.

Das, R., Sachdev, M.S., Sidhu, T.S. 2000. A Fault Locator for Radial Subtransmission and Distribution Lines. *IEEE Power Engineering Society Summer Meeting, vol. 1*, pp. 443– 448, Seattle, USA.

Dashti, R., Sadeh, J. 2010. A New Method for Fault Section Estimation in Distribution Network. *International Conference on Power System Technology,* 24-28 October 2010, pp. 1-5, Hangzhou, China.

Daubechies, I. 1992. *Ten Lectures on Wavelets*, SIAM, Philadelphia.

Davila, H. 2011. Records from DFRs vs. Records from Microprocessor-Based Relays. *In Proceedings of 11th Annual Georgia Tech Fault and Disturbance Analysis Conference*, Atlanta, Georgia, USA, pp.1-8.

De Almeidaa, M.C., Costab, F.F., Xavier-de-Souzac, S., Santana, F. 2001. Optimal Placement of Faulted Circuit Indicators in Power Distribution Systems. *Electric Power Systems Research, vol. 81,* 2001, pp. 699–706.

Demuth, H., Beale, M., Hagan, M. 2004. Neural Network Toolbox for Use with MATLAB, Users Guide Version 4.

DIgSILENT PowerFactory Manual, Version 13.1. 2005. Gomaringin, Germany.

Dugan, R.C., Kersting, W.H. 2006. Induction machine test case for the 34-bus test feeder-description. *IEEE Power Engineering Society General Meeting,* pp. 1-4.

Dunne, R., 2007. A Statistical Approach to Neural Networks for Pattern Recognition. John Wiley & Sons, Inc.

Dwivedi, U.D., Singh, S.N., Srivastava, S.C. 2008. A Wavelet Based Approach for Classification and Location of Faults in Distribution Systems. *Annual IEEE India Conference, INDICON 2008, vol. 2* pp. 488 – 493, India.

Ekici, S., Yildirim, S., Poyraz, M. 2008. Energy and Entropy-Based Feature Extraction for Locating Fault on Transmission Lines by using Neural Network and Wavelet Packet Decomposition. *Expert Systems with Applications, vol. 34*, pp. 2937–2944.

Eskom integrated report2011http://financialresults.co.za/2012/eskom_ar2012/divisional-report/distribution.php [11 August 2012].

Eskomintegratedreport2012http://financialresults.co.za/2011/eskom_ar2011/cnb_distribution03.php [21 February 2011].

Fausett, L.V., 1994. *Fundamentals of Neural Networks.* Prentice Hall.

Filomena, A.D., Resener, M., Salim, R.H., Bretas, A.S. 2009. Fault Location for Underground Distribution Feeders: An Extended Impedance-Based Formulation with Capacitive Current Compensation. *Electrical Power and Energy Systems vol. 31,* 2009, pp. 489–496.

Fugal D.L. 2009. Conceptual Wavelets in Digital Signal Processing: An In-depth Practical Approcah for the Non-Mathematician.

Fukuyama, Y., Ueki, Y. 1993. Fault analysis system using neural networks and artificial intelligence. *Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems,* pp. 20-25.

Gers, J.M. and Holmes, E.J. 1998. *Protection of electricity distribution networks*. London: The Institution of Electrical Engineers.

Girgis, A.A., Fallon, C.M., Lubkerman, D.L. 1993. A Fault Location Technique for Rural Distribution Feeder. *IEEE Transaction on Industry Application, vol. 29 n. 6,* pp. 1170-1175.

Gohokar, V.N., Khedkar, M.K. 2005. Faults Locations in Automated Distribution System. *Electric Power Systems Research vol. 75,* 2005, pp. 51–55.

Gong, Y., Guzman, A. 2011. Distribution Feeder Fault Location Using IED and FCI Information. *64th Annual Conference for Protection Relay Engineers*, Texas, USA, 11-14 April, 2011, pp. 1-10.

Guo-fang, Z., Yu-ping, L., 2008. Development of Fault Location Algorithm for Distribution Networks with DG. *International Conference on Sustainable Energy Technologies*, 24-27 Nov., 2008, pp. 1-5.

Gupta, J.B. 2004. *Switchgear and Protection*. Delhi: S.K. Kataria & Sons.

Hagan, M.T., Demuth, H.B., Beale, M. 1996. *Neural Network Design*. Brooks/Cole Publishing Company, USA.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation.* 2nd ed. Singapore: Pearson-Prentice-Hall.

Herraiz, S., Meléndez, J., Ribugent, G., Sánchez, J., Brunet, E. Fault Location in Distribution Power Systems by Means of a Toolbox Based on N-ary Tree Data Structures. *CIRED 19th International Conference on Electricity Distribution, No. 0656*, 21-24 May 2007, pp. 1-4, Vienna, Austria.

Hewitson, L.G., Brown, M. and Balakrishnan, R. 2005. *Practical Power Systems Protection*. Oxford: Newnes.

Hizam, H., Crossley, P.A. 2007. Estimation of Fault Location on a Radial Distribution Network Using Fault Generated Travelling Waves Signal. *Journal of Applied Sciences, vol. 7*, 2007, pp. 3736-3742.

Horowitz, S.H., Phadke, A.G. 1992. *Power System Relaying*. John Wiley & Sons Inc.

Jain, L. and Fanelli, A.M. 2000. *Recent Advances in artificial Neural Networks Design and Applications.* CRC Press LLC, Florida, USA.

Jain, A., Thoke, A.S., and Patel, R.N. 2009. Double Circuit Transmission Line Fault Distance Location using Artificial Neural Network. *World Congress on Nature and Biologically Inspired Computing*. pp. 13-18.

IEEE Distribution System Analysis Subcommittee. Radial Test Feeders [Online]. Available: http://www.ewh.ieee.org/soc/pes/dsacom/testfeeders.html.

IEEE guide for determining fault location on AC transmission and distribution lines, 2005. *IEEE Power Engineering Society Publ.*, New York, IEEE Std. C37.114.

Jalali, D. and Moslemi, N. 2005. Fault Location for Radial Distribution Systems Using Fault Generated High-Frequency Transients and Wavelet Analysis. *CIRED 18th International Conference on Electricity Distribution Turin,* session n. 3, 6-9 June 2005 pp. 1-4.

Järventausta, P., Verho, P. and Partanen, J. 1994. Using Fuzzy Sets to Model the Uncertainty in the Fault Location of Distribution Feeders. *IEEE Transactions on Power Delivery,* vol. 9, n. 2, pp. 954 – 960.

Jenkins, W.K. 1999. Fourier Series, Fourier Transforms, and the DFT. *Digital Signal Processing Handbook,* Ed. Vijay K. Madisetti and Douglas B. Williams. Boca Raton: CRC Press LLC.

Jones, T., 2008. *Artificial Intelligence: A Systems Approach*. Hingham, Massachusetts: Infinity Science Press LLC.

Jung, H., Park, Y., Han, M., Lee, C., Park, H. and Shin M, 2007. Novel Technique for Fault Location Estimation on Parallel Transmission Line Using Wavelet. *Electrical Power and Energy Systems,* vol. 29, pp.76–82.

Kersting, W.H. 2000. Radial distribution test feeders. Distribution System Analysis Subcommittee Report Power Engineering Society Summer Meeting, pp. 1-5.

Kersting, W. H. 2002. Distribution System Modeling and Analysis. New York: CRC Press LLC.

Kezunovic, M., Vasilic, S. and Ristanovic D. 2002. Interfacing Protective Relays and Relay Models to Power System Modeling Software and Data Files. *Proceedings of International Conference of Power System Technology*, 13-17 October 2002, pp. 253-259.

Kezunovic, M., Popovic, T., Sternfeld, S., Datta-Barua, M., Maragal, D. Automated Fault and Disturbance Analysis: Understanding the Configuration Challenge. *14th Annual Georgia Tech Fault and Disturbance Conference*, Atlanta, Georgia, May 2011, pp. 1-6.

Kohonen, T., 1987. *Self-Organization and Associative Memory, 2nd Ed.* Berlin: Springer-Verlag.

Kumar, K.S., Jayabarathi, T., Naveen, S., 2011. Fault identification and location in distribution systems using Support Vector Machines. *European Journal of Scientific Research,* vol. 51 n.1, pp.53-60.

Laithwaite, E.R. and Freris, L.L. 1980. *Electric Energy: Its generation, transmission and use*. UK: Mc Graw-Hill.

Lee, S.J., Choi M.S., Kang, S.H., Jin, B.G., Lee, D.S., Ahn, B.S., Yoon, N.S., Kim, H.Y., Wee, S.B. 2004. An Intelligent and Efficient Fault Location and Diagnosis Scheme for Radial Distribution Systems. *IEEE Transactions on Power Delivery,* vol. 19 n. 2, pp. 524-532.

Leondes, C., 1998. *Algorithms and Applications: Neural Network Systems Techniques and Applications.* Academic Press.

Li, Z., Li, W., Liu, R. 2005. Applications of entropy principles in power system: a survey. *IEEE/PES Transmission and Distribution Conference and Exhibition*, pp. 1–4.

Luger, G.F., Stubblefield, W.A., 1998. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* 3rd ed. Massachusetts: Addison Wesley Longman, Inc.

Magnago, F.H., Abur, A. 1999. A New Fault Location Technique for Radial Distribution Systems Based on High Frequency Signals. *IEEE Power Engineering Society Summer Meeting*, vol. 1, pp. 426-431.

Makming, P., Bunjongjit, S., Kunakorn, A., Jiriwibhakorn, M. and Kando M. 2002. Fault Diagnosis in Transmission Lines Using Wavelet Transform Analysis. *IEEE Transmission and Distribution Conference and Exhibition*, pp. 2246-2250.

Makram, E.B., Bou-Rabee, M.A. and Girgis, A.A. 1987. Three-phase modeling of unbalanced distribution systems during open conductors and/or shunt fault conditions using the bus impedance matrix. *Electric Power Systems Research 13*, pp. 173-183.

Mallat, S. 2009. *A Wavelet Tour of Signal Processing: The Sparse Way.* Massachusetts: Academic Press.

Malathi, V., Marimuthu, N.S., Baskar, S., Ramar, K. 2011. Application of extreme learning machine for series compensated transmission line protection. *Engineering Applications of Artificial Intelligence,* vol. 24, pp.880–887.

Martins, L.S., Pires, V.F. and Alegria, C.M. 2002. A New Accurate Fault Location Method Using Aβ Space Vector Algorithm. *Proceeding of 14th PSCC, session 08 paper 3,* 24-28 June 2002, pp.1-6, Sevilla, Spain.

Martins, L.S., Martins, J.F., Alegria, C.M., Pire,s V.F. 2003. A Network Distribution Power System Fault Location Based on Neural Eigenvalue Algorithm. *In Proceeding of IEEE Bologna Power Tech Conference,* 23-26 June 2003, pp. 1-6, Bologna, Italy.

Mehrotra, K., Mohan, C.K. and Ranka, S. 1996. *Elements of Artificial Neural Networks*. Bradford books.

Misiti, M., Misiti, Y., Oppenheim, G., & Poggi, J. M. 2004. Wavelet toolbox for use with Matlab, User's Guide, Ver. 3.

Mohamed, E.A. and Rao, N.D. 1995. Artificial Neural Network Fault Diagnostic System for Electric Power Distribution Feeders. *Electric Power Systems Research,* vol. 35, pp. 1-10.

Mohamed, E.A., Abdelaziz, A.Y., Mostafa, A.S. 2005. A neural network-based scheme for fault diagnosis of power transformers. *Electric Power Systems Research,* vol. 75, pp.29–39.

Mooney, J. 2012. Determining fault location on transmission and distribution lines. *PAC World Magazine*, September 2012, vol.21, pp. 18-25.

Mora, J.J., Bedoya, J.C. and Meléndez, J. 2006. Extensive Events Database Development Using ATP and MATLAB to Fault Location in Power Distribution Systems. *Proceedings of IEEE PES Transmission and Distribution Conference and Exposition Latin America*, 15-18 August 2006, pp. 1-6, Venezuela.

Mora-Fl`orez, J., Mel´endez, J. and Carrillo-Caicedo, G. 2008. Comparison of Impedance Based Fault Location Methods for Power Distribution Systems. *Electric Power Systems Research vol. 78 n. 4,* April 2008, pp. 657-666.

Mora-Flórez, J., Cormane-Angarita, J. and Ordó˜nez-Plata, G. 2009. K-Means Algorithm and Mixture Distributions for Locating Faults in Power Systems. *Electric Power Systems Research,* vol. 79, pp. 714–721.

Morales-España, G., Mora-Flórez, G. and Vargas-Torres, H. 2009. Elimination of Multiple Estimation for Fault Location in Radial Power Systems By Using Fundamental Single-End Measurements. *IEEE Transactions on Power Delivery,* vol. 24 n. 3, July 2009, pp. 1382-1389.

Nedic, D., Bathurst, G., Heath, J. 2007. A Comparison of Short Circuit Calculation Methods and Guidelines for Distribution Networks. *CIRED 19th International Conference on Electricity Distribution Vienna*, 21-24 May 2007, Session 3, Paper No 0562, pp. 1-4.

Negnevitsky, M., 2002. *Artificial Intelligence: A Guide to Intelligent Systems.* Addison Wesley.

Network Protection & Automation Guide (NPAG), 2011. *T&D Energy Automation & Information*. France: Published by Alstom Grid.

Ngaopitakkul, A., Apisit, C., Pothisarn, C., Jettanasen, C. and Jaikhan, S. 2009. Identification of Fault Locations in Underground Distribution System Using Discrete Wavelet Transform. *Proceedings of the International Multi-Conference of Engineers and Computer Scientists,* vol II, March 17-19 2010, pp.1-5, Hong Kong.

Nilsson, N.J., 1998. *Artificial Intelligence: A New Synthesis.* Morgan Kaufmann Publishers, Inc.

Öhrström, M., Geidl, M., Söder, L., Andersson, G. 2005. Evaluation of Travelling Wave Based Protection Schemes for Implementation in Medium Voltage Distribution Systems. *Proceedings of CIRED 18th International Conference on Electricity Distribution,* session n. 3, 6-9 June 2005, pp. 1-5, Turin, Italy.

Oliveira de, K.Z.C., Salim, R.H., Shuck Jr., and Bretas, A.S. 2009. Faulted Branch Identification on Power Distribution Systems Under Noisy Environment. *Paper submitted to the International Conference on Power Systems Transients (IPST 2009),* June 3-6 2009, pp. 1-5, Kyoto, Japan.

Panigrahi, B.K., Pandi, V.R. 2009. Optimal feature selection for classification of power quality disturbances using wavelet packet-based fuzzy k-nearest neighbor algorithm. *IET Generation, Transmission & Distribution,* vol. 3, iss. 3, pp. 296–306.

Park, M.H. 1996. Neural Network Control of a Chlorine Basin, University of California, pp 20-52.

Partridge, D., *Artificial Intelligence and Software Engineering: Understanding the Promise of the Future.* Chicago: Glenlake Publishing Company Ltd.

Pereira, R.A.F., da Silva, W.L.G., Kezunovic, M. and Mantovani, J.R.S. 2006. Location of Single Line-To-Ground Faults on Distribution Feeders Using Voltage Measurements. *IEEE PES Transmission and Distribution Conference,* 15-18 August 2006, pp.1-6, Venezuela.

Pereira, R.A.F., Kezunovic, M. and Mantovani, J.R.S. 2009. Fault Location Algorithm for Primary Distribution Feeders Based on Voltage Sags. *International Journal of Innovations in Energy Systems and Power,* vol. 4 no. 1, pp. 1-8.

Peretto, L., Sasdelli, R., Tinarelli, R. 2005. On uncertainty in wavelet based signal analysis. *Instrumentation and Measurement, IEEE Transactions on*, vol. 54, no. 4, pp. 1593 – 1599.

Prandoni, P. and Vetterli, M. 2008. *Signal Processing for Communications*. Boca Raton, Florida: Taylor and Francis Group, LLC.

Real-Time Digital Simulator Tutorial Manual (RSCAD version). 2010. RTDS Technologies, Winnipeg, Manitoba, Canada.

Report to the System Protection Subcommittee of the Power System Relaying Committee of the IEEE Power Engineering Society. 2006. Considerations for Use of Disturbance Recorders.

Rezaei, N., Javadian, S.A.M., Khalesi, N., Hagfiham, M.R. 2011. Diagnosis of impedance fault in distributed generations using radial basis function neural network. *IEEE International Conference on Smart Measurements for Future grids*, pp. 79-83.

Rayuda, R.K. 2010. A Knowledge-Based Architecture for Distributed Fault Analysis in Power Networks. *Engineering Applications of Artificial Intelligence vol. 23*, pp. 514-525.

RTDS Hardware Manual. 2009. RTDS Technologies, Winnipeg, Manitoba, Canada

Russel, S., Norvig, P., 2010. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall.

Saha, M.M., Provoost, F., Rosolowski, E. 2001. Fault Location Method for MV Cable Network. *7th International Conference on Developments in Power Systems Protection, Amsterdam, Netherlands,* pp.323–326.

Saha, M.M., Rosolowski, E. and Izykowski, J. 2005. ATP-EMTP Investigation for Fault Location in Medium Voltage Networks. *International Conference on Power Systems Transients (IPST'05), paper no. IPST 05-220,* on June 19-23 2005, pp. 1-6, Montreal, Canada.

Saha, M.M., Izykowski, J. and Rosolowski, E. 2010. *Fault Location on Power Networks*. London: Springer-Verlag Limited.

Salim, R.H., Caino de Oliveira, K.R. and Bretas, A.S. 2007. Fault Detection in Primary Distribution Systems Using Wavelets. *International Conference on Power Systems Transients (IPST'07),* June 4-7 2007, pp. 1-6, Lyon, France.

Samantaray, S.R., Panigrahi, B.K. and Dash, P.K. 2008. High Impedance Fault Detection in Power Distribution Networks Using Time-Frequency Transform and Probabilistic Neural Network. *IET Generation, Transmission, and Distribution,* vol. 2 n. 2, pp. 261–270.

Salim,R.H., Caino de Oliveira, K.R., Filomena, A.D., Resener, M., Bretas, A.S. 2008. Hybrid Fault Diagnosis Scheme Implementation for Power Distribution Systems Automation. *IEEE Transactions on Power Delivery,* vol. 23, n. 4, pp. 1846-1856.

Salim, R.H., Resener, M., Filomena, A.D., Caino De Oliveira, K.R., Bretas, A.S. 2009. Extended Fault-Location Formulation for Power Distribution Systems. *IEEE Transactions on Power Delivery,* vol. 24 n.2, pp. 508-516.

Samaan, N., McDermott, T., Zavadil, B., and Li, J. 2006. Induction machine test case for the 34-bus test feeder-steady state and dynamic solutions. *IEEE Power Engineering Society General Meeting,* pp. 1-5.

Santoso, S., Dugan, R.C., Lamoree, J., Sundaram, A. 2000. Distance Estimation Technique for Single Line-To-Ground Faults in a Radial Distribution System. *Proceedings of IEEE Power Engineering Society Winter Meeting vol. 4,* 23-27 January 2000, pp. 2551-2555.

Santoso, S., Zhou, Z. 2006. Induction machine test case for the 34-bus test feeder: a wind turbine time domain model. *IEEE Power Engineering Society General Meeting,* pp. 1-4.
SEL 451 User Manual. 2009. Schweitzer Engineering Laboratories.

Senger, E.C., Manassero, G., Goldemberg, C., Pellini, E.L. 2005. Automated Fault Location System For Primary Distribution Networks. *IEEE Transactions on Power Delivery,* vol. 20, n. 2. pp. 1332-1340.

Short, T., Kim, J., Melhorn, C. 2009. Update on Distribution System Fault Location Technologies and Effectiveness. *CIRED 20th International Conference on Electricity Distribution,* session 3 paper n. 0973, 8-11 June, 2009, pp. 1-4, Prague, Czech republic.

Silva, J.A., Funmilayo, H.B., Butler-Purry, K.L. 2007. Impact of Distributed Generation on the IEEE Node Radial Test Feeder with Overcurrent Protection. *Proceedings of 39th North American Power Symposium*. pp. 49-57.

Sinclair, A. 2011. Distance Protection in Distribution Systems: How it Assists with Integrating Distributed Resources. *Western* Protective *Relay Conference,* 18-20 October, 2011, Spokane, Washington. Pg 1-12.

Sodagar, I. 2000. *Time-Varying Analysis-Synthesis Filter Banks*. CRC Press LLC.

Subrahmanyam, J.B.V , Radhakrishna, C. A. 2010. Simple Approach of Three phase Distribution System Modeling for Power Flow Calculations. *International Journal of Electrical and Electronics Engineering*, vol. 4, no. 7, pp. 486-491.

Swingler, K. 2001. *Applying Neural Networks: A Practical guide.* San Francisco, CA: Morgan Kaufman Publishers, Inc.

Takagi, T., Yamakoshi, Y., Yamaura, Y., Kondow, R. Matsushima, T. 1982. Development of a New Type Fault Locator Using the One Terminal Voltage and Current Data. *IEEE Trans. Power Apparatus and Systems,* vol.101 n.8*,* pp. 2892-2898.

Takani, H., Kurosawa, Y., Kawano, F. 2007. High accuracy fault location algorithm with distributed parameters and evaluation using actual fault data. *10th Annual Georgia Tech Fault and Disturbance Analysis Conference*, April 30-May 1, 2007, pp.1-12.

Tang, Y., Wang, H.F., Aggarwal, R.K., Johns, A.T. 2000. Fault Indicators in Transmission and Distribution Systems. *International Conference on Electric Utility Deregulation and Restructuring and Power Technologies,* 4-7 April 2000, London, United Kingdom, pp. 238 – 243.

Tang, Y.Y. 2009. Wavelet Theory Approach to Pattern Recognition. *Series in Machine Perception and Artificial Intelligence*-vol. 74. World Scientific Publishing Co. Pte. Ltd.

Teo, C.Y. 1993. Automation of Knowledge Acquisition and Representation of Fault Diagnosis in Power Distribution Networks. *Electric Power Systems Research,* vol. 27, pp. 183-189.

Theodoridis, S., Koutroumbas, K., 2005. *Pattern Recognition* (2nd ed.) Academic Press.

Theraja, B.L., Theraja, A.K. 2008. *A Text Book of Electrical Technology*. 24th ed. New Delhi: S. Chand & Company Ltd.

Thomas, D.W.P., Carvalho, R.J.O., Pereira, E.T. 2003. Fault Location in Distribution Systems Based on Traveling Wave. *Proceedings of IEEE Bologna Power Technology Conference,* 23-26 June 2003, pp. 468-472, Bologna, Italy.

Thukaram, D., Khincha, H. P., Vijaynarasimha, H.P. 2005. Artificial Neural Network and Support Vector Machine Approach for Locating Faults in Radial Distribution Systems. *IEEE Transactions on Power Delivery,* vol. 20, n. 2, pp. 710-721.

Toman, P., Paar, M. and Batora, B. 2007. Using of the Artificial Neural Networks to the Localization of the Earth Faults in Radial Networks*. CIRED 19th International Conference on Electricity Distribution* Vienna, 21-24 May 2007, session 3, paper no 0591, pp. 1-3.

Ukil, A., Zivanovi´c, R., 2006. Abrupt Change Detection In Power System Fault Analysis Using Adaptive Whitening Filter and Wavelet Transform. *Electric Power Systems Research,* vol. 76, pp. 815–823.

Veelenturf, L.P.J., 1995. Analysis and Application of Artificial Neural Networks. Prentice Hall. NJ.
Wang,C., Nouri, H., Davies, T.S. 2000. A Mathematical Approach for Identification of Fault Sections on the Radial Distribution Systems. *10th Mediterranean Electrotechnical Conference (Melecon),* vol. 3, 29-31 May 2000, pp. 882-886.

Webb, A.R., 2002. *Statistical Pattern Recognition.* 2nd ed. West Sussex, England: John Wiley & Sons, Ltd.

Wen, F., Chang, C.N. 1996. A New Approach to Fault Diagnosis in Electrical Distribution Networks Using A Genetic Algorithm. *Proceeding of Artificial Intelligence in Engineering vol. 12,* pp. 69-80.

Whitaker, J.C. 2007. *AC Power Systems Hand book*. 3rd ed. Boca Raton: CRC Press, Taylor & Francis Group.

Williams C., McCarthy, C. and Cook, C.H. 2008. Predicting Reliability Improvements. *Tailoring Distribution Feeder reliability to Optimize Cost Per Customer Minute Saved*. IEEE Power & Energy Magazine, March/April, 2008, pp. 53-60.

Yang, M.T., Guan, J.L., Gu, J.C. 2007. High Impedance Faults Detection Technique Based on Wavelet Transform. *World Academy of Science, Engineering and Technology, vol. 28,* pp. 308-312.

Yanqiu, B., Zhao, J., Zhang, D. 2004. Single-phase-to-ground fault feeder detection based on transient current and wavelet packet. *International Conference on Power System Technology*, Singapore, 21-24 November, 2004, pp. 2006-2010.

Yilmaz, A.S., Subasi, A., Bayrak, M., Karsli, V.M., Ercelebi, E. 2007. Application of lifting based wavelet transforms to characterize power quality events. *Energy Conversion and Management,* vol. 48, pp. 112–123.

Yuan, C., Zeng, X.,Xia, Y. 2008. Improved algorithm for fault section location in distribution network with distributed generations. *International Conference on Intelligent Computation Technology and Automation*, pp. 893-896.

Yuehai Y., Yichuan B., Guofu X., Shiming X., Jianbo L. 2004. Fault Analysis Expert System for Power System. *International Conference on Power System Technology - POWERCON 2004 Singapore*, 21-24 November 2004, pp. 1-5.

Zayandehroodi, H., Mohamed, A., Shareef, H., Mohammadjafari, M. 2011. Determining exact fault location in a distribution network in presence of DGs using RBF neural networks. *International Conference on Information Reuse and Integration* (IRI), 3-5 August, 2011, Las Vegas, USA, pp. 434-438.

Zhang, N., Kezunovic, M. 2007. Transmission line boundary protection using wavelet transform and Neural Network. *IEEE Transactions on Power Delivery*, vol. 22, pp. 859–869.

Zhao, W., Song, Y.H., Min, Y. 2000. Wavelet Analysis Based Scheme for Fault Detection and Classification in Underground Power Cable Systems. *Proceedings of Electric Power Systems Research, vol. 53*, pp. 23–30.

Zheng-you H, Xiaoqing C, Guoming L. 2006. Wavelet entropy definition and its application for transmission line fault detection and identification (Part I: definition and methodology). *Proceeding of International Conference on Power System Technology*.

Zhengyou, H.E., Gao, S., Xiaoqin, C., Jun, Z., Zhiqian, B., Qingquan, Q. 2011. Study of a new method for power system transients classification based on wavelet entropy and neural network. Electrical Power and Energy Systems 33, pp. 402–410.

Zimath, S.L., Dutra, C.A., Seibel, C., Ramos, M.A.F., da Silva Filho, J.E. 2009. Comparison of Impedance and Traveling Wave Methods of Fault Location Using Real Faults. *Presented at the*

Georgia Tech Fault & Disturbance Analysis Conference, Atlanta, Georgia, 20-21 April, 2009, pp. 1-10.

Zimmerman, K., Costello, D. 2005. Impedance-based fault location experience. *58th Annual Conference for Protection Relay Engineers, Texas, USA, 5-7 April, 2005,* pp. 211-226.

Ziolkowski, V., da Silva, I.N., Flausino, R., Ulson, J.A. 2006. Fault identification in distribution lines using intelligent systems and statistical methods. *IEEE Mediterranean Electrotechnical Conference, Malaga, Spain, 16-19 May, 2006,* pp. 1122-1125.

APPENDICES

APPENDIX A: DATA FOR IEEE 34 NODE TEST FEEDER

The feeder parameters of the IEEE 34 node test feeder obtained from http://ewh.ieee.org/soc/pes/dsacom/testfeeders.html are listed in the following appendix. The load flow results can be obtained from the above-mentioned link.

Appendix A.1: Transformer Data

	kVA	kV-high	kV-low	R - %	X - %
Substation:	2500	69 - D	24.9 -Gr. W	1	8
XFM -1	500	24.9 - Gr.W	4.16 - Gr. W	1.9	4.08

Appendix A.2: Line Configuration

Overhead Line Configurations (Config.)

Config.	Phasing	Phase ACSR	Neutral ACSR	Spacing ID
300	B A C N	1/0	1/0	500
301	B A C N	#2 6/1	#2 6/1	500
302	A N	#4 6/1	#4 6/1	510
303	B N	#4 6/1	#4 6/1	510
304	B N	#2 6/1	#2 6/1	510

Appendix A.3: Line Data

Node A	Node B	Length(ft.)	Config.
800	802	2580	300
802	806	1730	300
806	808	32230	300
808	810	5804	303
808	812	37500	300
812	814	29730	300
814	850	10	301
816	818	1710	302
816	824	10210	301
818	820	48150	302
820	822	13740	302
824	826	3030	303
824	828	840	301
828	830	20440	301
830	854	520	301
832	858	4900	301
832	888	0	XFM-1
834	860	2020	301
834	842	280	301
836	840	860	301
836	862	280	301
842	844	1350	301
844	846	3640	301
846	848	530	301
850	816	310	301
852	832	10	301
854	856	23330	303
854	852	36830	301
858	864	1620	302
858	834	5830	301
860	836	2680	301
862	838	4860	304
888	890	10560	300

Appendix A.4: Load Parameters

Spot Loads

Node	Load Model	Ph-1 kW	Ph-1 kVAr	Ph-2 kW	Ph-2 kVAr	Ph-3 kW	Ph-4 kVAr
860	Y-PQ	20	16	20	16	20	16
840	Y-I	9	7	9	7	9	7
844	Y-Z	135	105	135	105	135	105
848	D-PQ	20	16	20	16	20	16
890	D-I	150	75	150	75	150	75
830	D-Z	10	5	10	5	25	10
Total		344	224	344	224	359	229

Distributed Loads

Node A	Node B	Load Model	Ph-1 kW	Ph-1 kVAr	Ph-2 kW	Ph-2 kVAr	Ph-3 kW	Ph-3 kVAr
802	806	Y-PQ	0	0	30	15	25	14
808	810	Y-I	0	0	16	8	0	0
818	820	Y-Z	34	17	0	0	0	0
820	822	Y-PQ	135	70	0	0	0	0
816	824	D-I	0	0	5	2	0	0
824	826	Y-I	0	0	40	20	0	0
824	828	Y-PQ	0	0	0	0	4	2
828	830	Y-PQ	7	3	0	0	0	0
854	856	Y-PQ	0	0	4	2	0	0
832	858	D-Z	7	3	2	1	6	3
858	864	Y-PQ	2	1	0	0	0	0
858	834	D-PQ	4	2	15	8	13	7
834	860	D-Z	16	8	20	10	110	55
860	836	D-PQ	30	15	10	6	42	22
836	840	D-I	18	9	22	11	0	0
862	838	Y-PQ	0	0	28	14	0	0
842	844	Y-PQ	9	5	0	0	0	0
844	846	Y-PQ	0	0	25	12	20	11
846	848	Y-PQ	0	0	23	11	0	0
Total			262	133	240	120	220	114

Appendix A.5: Regulator Data

Regulator Data

Regulator ID:	1		
Line Segment:	814 - 850		
Location:	814		
Phases:	A - B -C		
Connection:	3-Ph,LG		
Monitoring Phase:	A-B-C		
Bandwidth:	2.0 volts		
PT Ratio:	120		
Primary CT Rating:	100		
Compensator Settings:	Ph-A	Ph-B	Ph-C
R - Setting:	2.7	2.7	2.7
X - Setting:	1.6	1.6	1.6
Volltage Level:	122	122	122

Regulator ID:	2		
Line Segment:	852 - 832		
Location:	852		
Phases:	A - B -C		
Connection:	3-Ph,LG		
Monitoring Phase:	A-B-C		
Bandwidth:	2.0 volts		
PT Ratio:	120		
Primary CT Rating:	100		
Compensator Settings:	Ph-A	Ph-B	Ph-C
R - Setting:	2.5	2.5	2.5
X - Setting:	1.5	1.5	1.5
Volltage Level:	124	124	124

Appendix A.6: Capacitor Data

Node	Ph-A kVAr	Ph-B kVAr	Ph-C kVAr
844	100	100	100
848	150	150	150
Total	250	250	250

Appendix A.7: DG Synchronous Generator Parameters (Silva *et al.*, 2007)

Vrated(V)	480
kVArated	410
Prated (kW)	350
Vscheduled (pu)	1
Qmax(pu)	0.5
Qmin(pu)	-0.25
pf	0.8536585
Xd(pu)	1.76
Xq(pu)	1.66

X'd(pu)	0.21
X'q(pu)	0.18
X"d(pu)	0.13
X"q(pu)	0.11
ra(pu	0
r(pu)	0
rlr(pu)	0
Xlr(pu)	0
X0(pu)	0.065

The feeder parameters of the IEEE 13 node test feeder obtained from http://ewh.ieee.org/soc/pes/dsacom/testfeeders.html are listed in the following appendix. The load flow results can be obtained from the above-mentioned link.

Appendix B.1: Transformer Data

	kVA	kV-high	kV-low	R - %	X - %
Substation:	5,000	115 - D	4.16 Gr. Y	1	8
XFM -1	500	4.16 – Gr.W	0.48 – Gr.W	1.1	2

Appendix B.2: Line Configuration

Overhead Line Configuration Data:

Config.	Phasing	Phase	Neutral	Spacing
		ACSR	ACSR	ID
601	B A C N	556,500 26/7	4/0 6/1	500
602	C A B N	4/0 6/1	4/0 6/1	500
603	C B N	1/0	1/0	505
604	A C N	1/0	1/0	505
605	C N	1/0	1/0	510

Underground Line Configuration Data:

Config.	Phasing	Cable	Neutral	Space ID
606	A B C N	250,000 AA, CN	None	515
607	A N	1/0 AA, TS	1/0 Cu	520

Appendix B.3: Line Data

Node A	Node B	Length(ft.)	Config.
632	645	500	603
632	633	500	602
633	634	0	XFM-1
645	646	300	603
650	632	2000	601
684	652	800	607
632	671	2000	601
671	684	300	604
671	680	1000	601
671	692	0	Switch
684	611	300	605
692	675	500	606

Appendix B.4: Load Data

Spot Load Data:

Node	Load	Ph-1	Ph-1	Ph-2	Ph-2	Ph-3	Ph-3
	Model	kW	kVAr	kW	kVAr	kW	kVAr
634	Y-PQ	160	110	120	90	120	90
645	Y-PQ	0	0	170	125	0	0
646	D-Z	0	0	230	132	0	0
652	Y-Z	128	86	0	0	0	0
671	D-PQ	385	220	385	220	385	220
675	Y-PQ	485	190	68	60	290	212
692	D-I	0	0	0	0	170	151
611	Y-I	0	0	0	0	170	80
	TOTAL	1158	606	973	627	1135	753

Distributed Load Data:

Node A	Node B	Load	Ph-1	Ph-1	Ph-2	Ph-2	Ph-3	Ph-3
		Model	kW	kVAr	kW	kVAr	kW	kVAr
632	671	Y-PQ	17	10	66	38	117	68

Appendix B.5: Regulator Data

Regulator ID:	1		
Line Segment:	650 - 632		
Location:	50		
Phases:	A - B -C		
Connection:	3-Ph,LG		
Monitoring Phase:	A-B-C		
Bandwidth:	2.0 volts		
PT Ratio:	20		
Primary CT Rating:	700		
Compensator Settings:	Ph-A	Ph-B	Ph-C
R - Setting:	3	3	3
X - Setting:	9	9	9
Volltage Level:	122	122	122

Appendix B.6: Capacitor Data

Node	Ph-A	Ph-B	Ph-C
	kVAr	kVAr	kVAr
675	200	200	200
611			100
Total	200	200	300

This section presents the software codes for all the processes implemented for the Hybrid Fault Detection and Diagnosis (HFDD) method in the thesis. These codes are implemented in MATLAB 7.12.0 (R2011a).

C.1: Wavelet Entropy Calculation for Level-1 Detail Coefficient: MATLAB Script-*wavent1.m*

```
% INVESTIGATION OF METHODOLOGIES FOR FAULT DETECTION AND DIAGNOSIS
%                IN ELECTRIC POWER SYSTEM PROTECTION
%
%    **********************M.TECH RESEARCH*******************
%
%
%
% ************************A.C. ADEWOLE***********************
%
% **********************STUDENT NR. 211224863****************
%
% **********CAPE PENINSULA UNIVERSITY OF TECHNOLOGY**********
%
%
% Wavent Entropy (wavelet energy spectrum entropy)
% wavent(d1) returns the level-1 wavelet energy spectrum entropy
% of the wavelet detail coefficients of the DWT decomposition.
%
% ejk is the wavelet energy of signal at scale j instant k
% ej = sum(ejk);is the signal energy summation of n samples
% pjk = ejk/ej; is the relative wavelet energy
% WEE is the wavelet energy entropy
%
clear all
close all
function [wee]= wavent1(d1)
%
ejk = abs(d1).^2;
%
ej = sum(ejk);
%
pjk = ejk/ej;
%
wee = -sum((pjk).*log(eps+pjk));
% A.C. Adewole, Cape Peninsula University of Technology.
% 12 Jan., 2012
% end;
```

C.2: **Wavelet Entropy Calculation for Level-5 Detail Coefficient: MATLAB Script-*wavent5.m***

```
% Calculation of the wavelet entropy from level-5 detail coefficients
%
function [wee]= wavent5(d5)
%
% Wavent Entropy (wavelet energy spectrum entropy).
% wavent(d5) returns the level-5 wavelet energy spectrum entropy
% of the wavelet detail coefficients of the DWT decomposition
%
% ejk is the wavelet energy of signal at scale j instant k
% ej=sum(ejk);is the signal energy summation of n samples
% pjk=ejk/ej; is the relative wavelet energy
% WEE is the wavelet energy entropy
%
% tic;
ejk = abs(d5).^2;
%
ej = sum(ejk);
%
pjk = ejk/ej;
%
wee = -sum((pjk).*log2(eps+pjk));
% toc;
% A.C. Adewole, Cape Peninsula University of Technology.
% 12 Jan., 2012
% end;
```

C.3: **Wavelet Decomposition for the Wavelet Family Investigation: MATLAB Script-*dwt_exp.m***

```
% Performs the import of 3 phase and zero seq. current waveforms
% DWT decomposition of these waveforms into detail and approximation %
% coefficients
% Wavelet families investigated include: Daubechies db2, db3, db4, db5,
db8, % Coiflet-3, & Symlet-4
% xlsread is used to read in the ASCII file saved in MS Excel
%
[num] = xlsread('C:\Users\Charles\MyDocuments\CPUT\Thesis\Sim\Thesis_DWT_family
_analysis\l1slga2090.xls');
%
time = num(:,1); % Assigns entries in column 1 of the excel file to time
currentA = num(:,2);  % Assigns entries in column 2 of the phase A current
currentB = num(:,3);   % Assigns entries in column 3 of the phase B current
currentC = num(:,4);   % Assigns entries in column 4 of the phase C current
currentI0 = num(:,5); % Assigns entries in column 5 of the phase I0 current
%
% Decomposition of phase A current
% Vector C gives the detail coefficients for all the levels and the
% approximate coefficients for the last level
% Vector L gives the lengths of each the coefficients
[C,L] = wavedec(currentA,6,'db8');
%
% detcoef extracts the detail coefficients from the vector C
[d1,d2,d3,d4,d5,d6] = detcoef(C,L,[1,2,3,4,5,6]);
%
```

```
% Wavelet energy spectrum entropy function for levels-1,4,5,&6 decomp.
phase A
% by calling the respective m.files for the entropy calculations
WEEa1 = wavent1(d1)
WEEa2 = wavent2(d2)
WEEa3 = wavent3(d3)
WEEa4 = wavent4(d4)
WEEa5 = wavent5(d5)
WEEa6 = wavent6(d6)
%
%
% Wavelet Decomposition of Phase B
[C,L] = wavedec(currentB,6,'db8');
[d1,d2,d3,d4,d5,d6] = detcoef(C,L,[1,2,3,4,5,6]);
%
% Wavelet energy spectrum entropy function for levels-1,4,5,&6 decomp.
% phase B
WEEb1 = wavent1(d1)
WEEb2 = wavent2(d2)
WEEb3 = wavent3(d3)
WEEb4 = wavent4(d4)
WEEb5 = wavent5(d5)
WEEb6 = wavent6(d6)
%
%
% Wavelet Decomposition of Phase C
[C,L] = wavedec(currentC,6,'db8');
[d1,d2,d3,d4,d5,d6] = detcoef(C,L,[1,2,3,4,5,6]);
%
% Wavelet energy spectrum entropy function for levels-1,4,5,&6 decomp. for
% phase C
WEEc1 = wavent1(d1)
WEEc2 = wavent2(d2)
WEEc3 = wavent3(d3)
WEEc4 = wavent4(d4)
WEEc5 = wavent5(d5)
WEEc6 = wavent6(d6)
%
%
% Wavelet Decomposition of Phase I0
[C,L] = wavedec(currentI0,6,'db8');
[d1,d2,d3,d4,d5,d6] = detcoef(C,L,[1,2,3,4,5,6]);
%
% Wavelet energy spectrum entropy function for levels-1,4,5,&6 decomp. for
phase I0
WEEI01 = wavent1(d1)
WEEI02 = wavent2(d2)
WEEI03 = wavent3(d3)
WEEI04 = wavent4(d4)
WEEI05 = wavent5(d5)
WEEI06 = wavent6(d6)
% end;
```

C.4: Input Menu to the HFDD Method:

MATLAB Script-*splash.m*

```
% Splash screen for the HFDD method
% Presents a menu with options for different file formats as input to the
% HFDD method
close all
clear all
clc
%
x = menu('HFDD METHOD FOR DS','HFDD-ASCII','Fault Detection-XLS','Fault
Classification-XLS','RTDS Comtrade File','SEL Comtrade File');
%
switch (x)

% Read data file/compute level-1 wavelet decomposition/compute level-5
% wavelet decomposition
% Read COMTRADE files from RSCAD runtime and event files from the IED
        case 1
            tdfread;
        case 2
            wavelet_main1;
        case 3
            wavelet_main;
        case 4
            read_comtrade_rtdswave;
        case 5
            read_comtrade_sel451events;
% end;
```

C.5: Fault Detection Algorithm:

MATLAB Script-*faultmain.m*

```
% Rule-Based Algorithm for Fault Detection
% Begin initialization-do not edit
% clear
% clc
global ra rb rc rI0
%ra==rb==rc==rI0==0;
%
% Phase A threshold  = 1.6
% Phase B threshold  = 2.2
% Phase C threshold  = 2.2
% Phase I0 threshold = 2.0
%
x = menu ('Fault Detection Algorithm?','Fault Detection');
disp ('    ');
disp (' STARTING FAULT DETECTION ALGORITHM  ');
disp ('    ');
%
% Wavelet Energy Entropy for Level-1
ra = WEEa1;
rb = WEEb1;
rc = WEEc1;
rI0 = WEEI01;
```

```
% Wavelet Energy Entropy for Level-5
ra5 = WEEa5;
rb5 = WEEb5;
rc5 = WEEc5;
rI05 = WEEI05;
%
% Rule-Based Algorithm for Fault Detection
if((ra<=1.6)&&(rb<=2.2)&&(rc<=2.2)&&(rI0<=2.000));
    disp ('    ');
    disp(' NO-FAULT')
else
    disp ('    ');
    disp(' RESULT = FAULT DETECTED')
disp ('    ');
disp (' STARTING FAULT TYPE & FAULTED PHASE CLASSIFICATION ALGORITHM  ');
disp ('    ');
%
%  call SLG function
faulty(ra5,rb5,rc5,rI05)
%
%  A.C. Adewole, Cape Peninsula University of Technology.
%  28 Jan., 2012
%  end;
```

C.6: Fault Classification Algorithm:

MATLAB Script-*faulty.m*

```
% Rule-Based Algorithm for Fault Classification (Single Line-to-Ground
Faults)
%
function [rules] =faulty(ra5,rb5,rc5,rI05)
%
% Compute Wavelet Energy Entropy Per Unit Criteria for Phases A, B, and C
la = ra5/(ra5+rb5+rc5);
lb = rb5/(ra5+rb5+rc5);
lc = rc5/(ra5+rb5+rc5);
%
% Read Results from Command Window into Workspace
assignin('base', 'la', la);
assignin('base', 'lb', lb);
assignin('base', 'lc', lc);
assignin('base', 'rI05',rI05);
%
% Assign Threshold Values for Phases A, B, and C
tha = 0.3390;
thb = 0.3250;
thc = 0.3370;
%
% Rule Based Algorithm for Fault Type and Faulted Phase(s) Classification
if ((la<tha)&&(lb>thb)&&(lc>thc));
        disp ('    ');
        disp(' RESULT = FAULT TYPE IS SINGLE PHASE-TO-GROUND')
        disp ('    ');
        disp(' RESULT = FAULTED PHASE IS PHASE A')
%
```

```
   elseif ((la>tha)&&(lb<thb)&&(lc>thc));
        disp ('    ');
        disp(' RESULT = FAULT TYPE IS SINGLE PHASE-TO-GROUND')
        disp ('    ');
        disp(' RESULT = FAULTED PHASE IS PHASE B')
%
       elseif ((la>tha)&&(lb>thb)&&(lc<thc));
        disp ('    ');
        disp(' RESULT = FAULT TYPE IS SINGLE PHASE-TO-GROUND')
        disp ('    ');
        disp(' RESULT = FAULTED PHASE IS PHASE C')
            else
                disp ('    ');
   disp (' RESULT = FAULT TYPE NOT SLG, EXECUTE FUNCTION FOR 2PH. FAULT');
%
%  call 2PH. function
faulty2ph(la,lb,lc,rI05)
%
%  A.C. Adewole, Cape Peninsula University of Technology.
%  28 Jan., 2012
%  end;
```

C.7: Fault Classification Algorithm:

MATLAB Script-*faulty2ph.m*

```
% Rule-Based Algorithm for Fault Classification (Two Phase Faults)
function [rules] =faulty2ph(la,lb,lc,rI0)
%
% classifies fault type and faulted phase(s) in an electric power system
% classification is based on entropy calc. of
% decomposed 3ph. current and zero seq. current
%
% inputs: ra,rb,rc,I0 are threshold values for individual phases obtained
from % extensive simulations
%
% Rules for 2 phase faults
%
tha = 0.3390;
thb = 0.3250;
thc = 0.3370;
if((la<tha)&&(lb<thb)&&(lc>thc)&&(rI0>3.85));
          disp ('    ');
          disp(' RESULT = FAULT TYPE IS TWO PHASE')
          disp ('    ');
          disp(' RESULT = FAULTED PHASES -> PHASE A-B')

   elseif ((la>tha)&&(lb<thb)&&(lc<thc)&&(rI0>3.3));
          disp ('    ');
          disp(' RESULT = FAULT TYPE IS TWO PHASE')
          disp ('    ');
          disp(' RESULT = FAULTED PHASES -> PHASE B-C')
%
  elseif ((la<tha)&&(lb>thb)&&(lc<thc)&&(rI0>3.5));
          disp ('    ');
          disp(' RESULT = FAULT TYPE IS TWO PHASE')
          disp ('    ');
          disp(' RESULT = FAULTED PHASES -> PHASE C-A')

%
```

```
      else
         disp ('    ');
         disp (' FAULT TYPE NOT 2PH., EXECUTE FUNCTION FOR 2PH.-G FAULT');
%
         disp ('    ');

%
%  call 2PH-G function
faulty2phg(la,lb,lc,rI0)
%

end;
%
%  A.C. Adewole, Cape Peninsula University of Technology.
%  28 Jan., 2012
%  end
```

C.8: **Fault Classification Algorithm:**

MATLAB Script-*faulty2phg.m*

```
% Rule-Based Algorithm for Fault Classification (Two Phase-to-Ground and
Three % Phase Faults)
function [rules] = faulty2phg(la,lb,lc,rI0)
%
%
tha = 0.3390;
thb = 0.3250;
thc = 0.3370;
% Rules for 2Ph.-G
if((la<tha)&&(lb<thb)&&(lc>thc)&&(rI0<3.85));
                 disp ('    ');
        disp(' FAULT TYPE IS TWO PHASE-TO-GROUND')
        disp ('    ');
        disp(' FAULTED PHASES -> A-B-G')
%
%
      elseif ((la>tha)&&(lb<thb)&&(lc<thc)&&(rI0<3.3));
               disp ('    ');
        disp(' FAULT TYPE IS TWO PHASE-TO-GROUND')
        disp ('    ');
        disp(' FAULTED PHASES -> B-C-G')
%
     elseif ((la<tha)&&(lb>thb)&&(lc<thc)&&(rI0<3.52));
         disp ('    ');
         disp(' FAULT TYPE IS TWO PHASE-TO-GROUND')
         disp ('    ');
         disp(' FAULTED PHASES -> C-A-G')
%
       else
        disp ('    ');
        disp (' FAULT TYPE NOT 2PH.-G');
        disp ('    ');
        disp (' FAULT TYPE IS 3PH.');
%
       end
%
%  A.C. Adewole, Cape Peninsula University of Technology.
%  28 Jan., 2012
%  end
```

C.9: **Neural Network Training for Fault Section Identification: MATLAB Script-*nnfsitraining.m***

```
% This routine is for neural network training for fault section
identification
% Supervised learning is used. Hence, both inputs and targets are
% presented to the network in batch mode
% Method used is the early-stopping method
% The networks are trained for pattern recognition
%
clear all
close all
clc
%
% tic
disp('Neural Network Classifier for Fault Section Identification')
disp('      ')
disp('by A.C. Adewole')
disp('      ')
disp('Initialising ...')
disp('      ')
disp('Loading the training and target dataset "p" & "t"...')
%
% Load training dataset using excel file
[num] =xlsread('C:\Users\Charles\My
Documents\CPUT\Thesis\Sim\nndata\nndata230512.xls');
%
% Assign inputs from excel columns to vectors
a = num(:,1);
b = num(:,2);
c = num(:,3);
d = num(:,4);
e = num(:,6);
f = num(:,7);
g = num(:,8);
h = num(:,9);
%
p = [a b c d];
t = [e f g h];
%
p = transpose(p);
t = transpose(t);
%tic
%
inputs = p;
targets = t;
%
% Create a Pattern Recognition Network
%
hiddenLayerSize = 25;
net = patternnet(hiddenLayerSize);
%
% Choose Input and Output Pre/Post-Processing Functions
%
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'}; % Normalize
to +1 and -1 & remove matrix rows with constant values.
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};% Normalize
to +1 and -1 & remove matrix rows with constant values.


% Setup Division of Data for Training, Validation, Testing
net.divideFcn = 'dividerand';           % Divide data randomly
net.divideMode = 'sample';              % Divide up every sample
net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
%
net.trainFcn = 'trainlm';  % SCG = Scaled Conjugate Gradient; LM =
Levenberg-Marquardt

% Define training parameters
net.performFcn = 'mse';                  % Mean squared error
net.trainParam.goal = 0.001;              % Performance goal
net.trainParam.show = 15;                % Epochs between displays
net.trainParam.epochs = 1000;            % Maximum number of epochs to train
net.trainParam.max_fail = 5              % Maximum validation failures (used
to
                                         % avoid over-fitting)
net.trainParam.min_grad = 1e-10          % Minimum performance gradient
net.trainParam.mu = 0.001                % Initial Mu (used to control the
                                         % weight updating process)
net.trainParam.mu_dec = 0.1              % Mu decrease factor
net.trainParam.mu_inc = 10               % Mu increase factor
net.trainParam.mu_max = 1e10             %Maximum Mu
%
% Choose Plot Functions
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression','plotroc', 'plotfit'};

% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% Calculate Absolute Percentage Error
% APE =(abs(t-a)/(t)*100)
% View the Network
% view(net)

test = [targets;outputs]
% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
figure, plotconfusion(targets,outputs)
% figure, ploterrhist(errors)
figure, plotroc(t,outputs)

% tInd = tr.testInd;
% tstOutputs = net(inputs(tInd));
% tstPerform = perform(net,targets(tInd),tstOutputs)
% toc
```

```
% Uncomment these lines to view the weights and biases.
% net.IW{1,1}%to view hidden layer weights
% net.LW{2,1}%to view output layer weights
% net.b{1,1}%to view hidden layer bias
% net.b{2,1}%to view output layer bias
end
```

C.10: Neural Network Training for Fault Section Identification: MATLAB Script-*nnfsitraining_reg.m*

```
% This routine is for neural network training for fault section
identification
% Supervised learning is used. Hence, both inputs and targets are
% presented to the network in batch mode
% This script file is used for the hold-out and regularization methods
% respectively
% The networks are trained for pattern recognition
%
clear all
close all
clc
%
disp('Neural Network Approximator for Fault Location')
disp('       ')
disp('by A.C. Adewole')
disp('       ')
disp('Initialising ...')
disp('       ')
disp('Loading the training and target dataset "p" & "t"...')
%
[num] =xlsread('C:\Users\Charles\My
Documents\CPUT\Thesis\Sim\nndata\nndata230512.xls');
%
a = num(:,1);
b = num(:,2);
c = num(:,3);
d = num(:,4);
e = num(:,6);
f = num(:,7);
g = num(:,8);
h = num(:,9);
%
P = [a b c d];
t = [e f g h];
%
p = transpose(p);
t = transpose(t);
% tic
%
% Normalize the input dataset to -1 to +1 range
[pn,minp,maxp,tn,mint,maxt] = premnmx(p,t);
%
% rand('state',0)%1916041
rand('state');          % Returns current state of the generator
Ntrials = 10            % where Ntrials is the number of trials/network
j=0
for h = 5:5:55          % h is the nr. of neurons in the hidden layer
     j = j+1
     H = h
    for i =1:Ntrials
```

```
Set the Network Architecture
net = init(net);        % Initialize the network
net = newff(minmax(pn),[5,5,4],{'tansig','tansig','tansig'},'trainlm');
%
% Set the network parameters
net.performFcn = 'mse';                 % Mean squared error
net.trainParam.goal =0.001;             % Performance goal
net.trainParam.show = 15;               % Epochs between displays
net.trainParam.epochs = 1000;           % Maximum number of epochs to train
net.trainParam.max_fail = 5             % Maximum validation failures (used
to
                                        % avoid over-fitting)
net.trainParam.min_grad = 1e-10         % Minimum performance gradient
net.trainParam.mu = 0.001               % Initial Mu (used to control the
                                        % weight updating process)
net.trainParam.mu_dec = 0.1             % Mu decrease factor
net.trainParam.mu_inc = 10              % Mu increase factor
net.trainParam.mu_max = 1e10            %Maximum Mu
%
%
% Train the network
[net,tr] = train(net,pn,tn);

%[net,tr]=train(net,pn,tn);
outputs = sim(net,pn);
errors = gsubtract(tn,outputs);
performance = perform(net,tn,outputs)
figure, plotconfusion(tn,outputs)
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotroc','plotfit'};
figure, plotroc(t,outputs)
% toc
%
% Post processing of the trained network
an = sim(net,pn);
postmnmx(an,mint,maxt);
%
[t;a];
% end
```

C.11: Neural Network Training for Fault Location: MATLAB Script-*nnfltraining.m*

```
% This routine is for neural network training for fault location
% Supervised learning is used. Hence, both inputs and targets are
% presented to the network in batch mode
% This script file is used for the hold-out and regularization methods
% respectively
% The networks are trained as function approximators/regressors
clear all
close all
clc
%
disp('Neural Network Approximator for Fault Location')
disp('      ')
disp('by A.C. Adewole')
disp('      ')
disp('Initialising ...')
disp('      ')
```

```
disp('Loading the training and target dataset "p" & "t"...')
%
%[num] = xlsread('C:\Users\Charles\My
Documents\CPUT\Thesis\Sim\nndata\FLdata\Final_NNtraining data.xls');
[num] = xlsread
('C:\Users\Charles\MyDocuments\CPUT\Thesis\Sim\nndata\nndata230512.xls');
a = num(:,1);
b = num(:,2);
c = num(:,3);
d = num(:,4);
e = num(:,5);
%
p = [a b c d];
t = [e];
p = transpose(p);
t = transpose(t);
%
% tic
%
inputs = p;
targets = t;
%
% Create a Fitting Network
hiddenLayerSize = 21;
net = fitnet(hiddenLayerSize);
%
% Choose Input and Output Pre/Post-Processing Functions
%
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};
%
% Setup Division of Data for Training, Validation, Testing
%
net.divideFcn = 'dividerand';           % Divide data randomly
net.divideMode = 'sample';              % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
%
%
net.trainFcn = 'trainlm';               % Levenberg-Marquardt

% Set the network parameters
net.performFcn = 'mse';                 % Mean squared error
net.trainParam.goal = 0.0001;           % Performance goal
net.trainParam.show = 15;               % Epochs between displays
net.trainParam.epochs = 1000;           % Maximum number of epochs to train
net.trainParam.max_fail = 5             % Maximum validation failures (used
to avoid over-fitting)
net.trainParam.min_grad = 1e-10         % Minimum performance gradient
net.trainParam.mu = 0.001               % Initial Mu (used to control the
                                        % weight updating process)
net.trainParam.mu_dec = 0.1             % Mu decrease factor
net.trainParam.mu_inc = 10              % Mu increase factor
net.trainParam.mu_max = 1e10            % Maximum Mu

% Train the Network
[net,tr] = train(net,inputs,targets);

net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
```

```
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)
%
test = [targets;outputs]
% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotfit(net,inputs,targets)
% figure, plotregression(targets,outputs)
% figure, ploterrhist(errors)
% toc
% end
```

C.12: Neural Network Training for Fault Location: MATLAB Script-*nnfltraining_reg.m*

```
% This routine is for neural network training for fault location
% Supervised learning is used. Hence, both inputs and targets are
% presented to the network in batch mode
% This script file is used for the hold-out and regularization methods
% respectively
% The networks are trained as function approximators/regressors
%
clear all
close all
clc
%
disp('Neural Network Classifier for Fault Section Identification')
disp('      ')
disp('by A.C. Adewole')
disp('      ')
disp('Initialising ...')
disp('      ')
disp('Loading the training and target dataset "p" & "t"...')
%
[num] = xlsread('C:\Users\Charles\My
Documents\CPUT\Thesis\Sim\nndata\nndata230512.xls');

a = num(:,1);
b = num(:,2);
c = num(:,3);
d = num(:,4);
e = num(:,6);
%
p = [a b c d];
t = [e];
%
p = transpose(p);
```

```
t = transpose(t);
%
% tic
%
% Normalize the input dataset to -1 to +1 range

[pn,minp,maxp,tn,mint,maxt] = premnmx(p,t);
%
%rand('state',0)1916041
rand('state');
Ntrials = 10
j=0
for h = 8:4:24
     j = j+1
     H = h
     for i =1:Ntrials
% Set the Network Architecture
%
net = newff(minmax(pn),[H,1],{'tansig','purelin'},'trainlm');
%
% Set the network parameters
net.performFcn = 'mse';                  % Mean squared error
net.trainParam.goal =0.001;              % Performance goal
net.trainParam.show = 15;                % Epochs between displays
net.trainParam.epochs = 1000;            % Maximum number of epochs to train
net.trainParam.max_fail = 5              % Maximum validation failures (used
to avoid over-fitting)
net.trainParam.min_grad = 1e-10          % Minimum performance gradient
net.trainParam.mu = 0.001                % Initial Mu (used to control the
weight updating process)
net.trainParam.mu_dec = 0.1              % Mu decrease factor
net.trainParam.mu_inc = 10               % Mu increase factor
net.trainParam.mu_max = 1e10             % Maximum Mu
%
%
%net = init(net);
%
% Train the network
[net,tr] = train(net,pn,tn);
%
% Post-processing of trained network
outputs = sim(net,pn);
a = postmnmx(outputs,mint,maxt);
errors = gsubtract(tn,outputs);
performance = perform(net,tn,outputs)
%
% figure, plotconfusion(tn,outputs)
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotfit'};
%
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};
figure, ploterrhist(errors)

% View the Network
view(net)
% toc
```

```
% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotfit(net,inputs,targets)
% figure, plotregression(targets,outputs)
% figure, ploterrhist(errors)
% end
```

C.13: Neural Network Training for Fault Location: MATLAB Script-*nntesting_fsi.m*

```
% Neural network testing for fault section identification and fault
location
%
[num] =xlsread ('C:\Users\Charles\My
Documents\CPUT\Thesis\Sim\nndata\nndata230512.xls');
a1 = num(:,1);
b1 = num(:,2);
c1 = num(:,3);
d1 = num(:,4);
f1 = num(:,6);
g1 = num(:,7);
h1 = num(:,8);
i1 = num(:,9);
%
pnew = [a1 b1 c1 d1];
tnew = [f1 g1 h1 i1]; % target vectors for fault section ident. Testing
%
% Uncomment for fault location testing
% tnew = [e1]; % target vectors for fault section ident. testing
%
pnew = transpose(pnew);
tnew = transpose(tnew);
pnewn = tramnmx(pnew,minp,maxp);
anewn = sim(net,pnewn);
anew = postmnmx(anewn,mint,maxt);
tic
test = [tnew;anew]
toc
% end
```

C.14: Neural Network Algorithm for Fault Section Identification: MATLAB Script-*fsi1ph.m*

```
% Algorithm for fault section identification based on neural network
% Load Trained Neural Network for SLG Fault Section Identification
load fsi1ph
%
% Uncomment when necessary
% load fsi2ph          % Trained neural network for two phase faults
% load fsi2phg         % Trained neural network for two phase-to-grd.
faults
% load fsi3ph          % Trained neural network for three phase faults
%
assignin('base', 'la', la);
assignin('base', 'lb', lb);
```

```
assignin('base', 'lc', lc);
assignin('base', 'rI05',rI05);
pnew = [la lb lc rI05];
%
pnew = transpose(pnew);
%
net = evalin('base','net');
tic
FSI = sim(net,pnew)
toc
% end
```

C.15: Neural Network Algorithm for Fault Location:

MATLAB Script-*fl1ph.m*

```
% Algorithm for fault location based on neural network
% Load Trained Neural Network for Single Phase-Ground Fault Location
load fl1ph
%
% Uncomment when necessary
% load fl2ph            % Trained neural network for two phase faults
% load fl2phg           % Trained neural network for two phase-to-grd.
faults
% load fl3ph            % Trained neural network for three phase faults
%
% Write Parameters to Workspace
assignin('base', 'la', la);
assignin('base', 'lb', lb);
assignin('base', 'lc', lc);
assignin('base', 'rI05',rI05);
%
pnew = [la lb lc rI05];
%
pnew = transpose(pnew);
%
net = evalin('base','net');
tic
FL = sim(net,pnew)
toc
% end
```

C.16: Visualization Plot of WEE Per Unit:

MATLAB Script-*wee_visualplot.m*

```
% Visualization Plot for Wavelet Energy Spectrum Entropy Per Unit
%
[num]=xlsread('C:\Users\Charles\MyDocuments\CPUT\Thesis\Sim\3dplot_thesis_2
ph_dg.xls');
a = num(:,1);
b = num(:,2);
c = num(:,3);
d = num(:,4);
e = num(:,5);
f = num(:,6);
g = num(:,7);
h = num(:,8);
i = num(:,9);
```

```
plot3(a,b,c,'b*')
grid
hold on
plot3(d,e,f,'go')
plot3(g,h,i,'r+')
grid
legend('2Ph.-AB','2Ph.-BC','2Ph.-CA','Location','NorthEastOutside')
title ('Distribution Plot of 2Ph. Faults (DG2)')
xlabel('lamda-ab')
ylabel('lamda-bc')
zlabel('lamda-ca')
grid on
% end
```

C.17: Plotting of Mother Wavelets:

MATLAB Script-*waveletfamilyplotting.m*

```
% Plotting of various mother wavelets
%
% Plotting of Morlet Mother Wavelet
[psi,xval] = wavefun('morl',10);
plot(xval,psi); title('Morlet Wavelet');
%
%
% Plotting of Daubechies-4 Scaling and Wavelet Function
[phi,psi,xval] = wavefun('db4',10);
subplot(211);
plot(xval,phi); title('db4 Scaling Function');
subplot(212);
plot(xval,psi); title('db4 Wavelet Function');
%
%
% Plotting of Daubechies-4 Scaling and Wavelet Function
[psi,xval] = wavefun('mexh',10);
plot(xval,psi); title('Mexican Hat Wavelet');
% end
```

C.18: Plotting of Activation Functions:

MATLAB Script-*activation_functions.m*

```
% Plots various activation functions used in neural network training
%
% Plotting Hardlim Activation Function
n = -5:0.1:5;
plot(n,hardlim(n),'c+:');
x = linspace(-5,5,501);
y = hardlim(x);
plot(x,y)
title('hardlim')
xlabel('v')
ylabel('a')
%
%
% Plotting Hardlims Activation Function
x = linspace(-5,5,501);
y = hardlims(x);
```

```
plot(x,y)
title('Hardlims')
xlabel('v')
ylabel('a')
%
%
% Plotting Logsig Activation Function
x = linspace(-5,5,501);
y = logsig(x);
plot(x,y)
title('Logsig')
xlabel('v')
ylabel('a')
%
%
% Plotting Poslin Activation Function
x = (-5:0.1:5);
y = poslin(x);
plot(x,y)
title('Poslin')
xlabel('v')
ylabel('a')
%
%
% Plotting Purelin Activation Function
x = linspace(-5,5,501);
y = purelin(x);
plot(x,y)
title('Purelin')
xlabel('v')
ylabel('a')
grid on
%
%
% Plotting Radbas Activation Function
x = (-5:0.1:5);
y = radbas(x);
plot(x,y)
title('radbas')
xlabel('v')
ylabel('a')
% Plotting Satlin Activation Function
x = (-5:0.1:5);
y = satlin(x);
plot(x,y)
title('satlin')
xlabel('v')
ylabel('a')
%
%
% Plotting Satlins Activation Function
x = (-5:0.1:5);
y = satlins(x);
plot(x,y)
title('satlins')
xlabel('v')
ylabel('a')
%
%
% Plotting Tansig Activation Function
x = linspace(-5,5,501);
y = tansig(x);
plot(x,y)
title('tansig')
```

```
xlabel('v')
ylabel('a')
%
%
% Plotting Tribas Activation Function
x = (-5:0.1:5);
y = tribas(x);
plot(x,y)
title('tribas')
xlabel('v')
ylabel('a')
% end
```

C.19: **Plotting of Short Circuit Current for the various Case Studies:**

MATLAB Script-*shortcircuitbar_plot.m*

```
% Plotting of short circuit current bar chart for analysis
%
x = [800 808 816 824 854 832 858 834 836.1 836.2];
yMat = [944.87  990.1   990.26  992.07
636.76  705.52  706.23  711.95
266.43  351.04  352.3   362.04
258.59  343.21  344.47  354.38
224.17  314.39  315.82  327.62
631.49  883.7   887.66  923.53
138.77  237.97  239.96  256.33
143.58  183.72  236.72  186.35
141.296 181.42  231.72  183.39
141.292 181.42  231.71  183.38];
 figure; bar(yMat);
 a = text(0.5,890,'N-800')    % create text object in current axes
 b = text(1.9,730,'N-808')
 c = text(2.9,380,'N-816')
 d = text(3.9,380,'N-824')
 e = text(5.0,380,'N-854')
 f = text(5.9,890,'N-832')
 g = text(7.0,280,'N-858')
 h = text(8.0,280,'N-834')
 i = text(9.0,280,'N-836-1')
 j = text(10.0,280,'N-836-2')
set(a,'rotation',90)           % Set handle graphics object properties
set(b,'rotation',90)
set(c,'rotation',90)
set(d,'rotation',90)
set(e,'rotation',90)
set(f,'rotation',90)
set(g,'rotation',90)
set(h,'rotation',90)
set(i,'rotation',90)
set(j,'rotation',90)
gtext('N = Node')
title ('Short Circuit Current for Case Studies','FontSize',12)
xlabel ('Nodes','FontSize',12)
ylabel (' Max Short Circuit Current/A ','FontSize',12)
% end
```

APPENDIX D: SCRIPT FILE FOR BATCH MODE OPERATION WITH THE RTDS

```
/*****************************APPENDIX D************************** /
/* INVESTIGATION OF METHODOLOGIES FOR FAULT DETECTION AND DIAGNOSIS */
/*             IN ELECTRIC POWER SYSTEM PROTECTION                 */
/ *********************************************************** /
/ **********************M.TECH RESEARCH********************* /
/ *********************************************************** /
/ ***********************A.C. ADEWOLE*********************** /
/ *********************************************************** /
/ *********************STUDENT NR. 211224863***************** /
/ *********************************************************** /
/ *********CAPE PENINSULA UNIVERSITY OF TECHNOLOGY*********** /
/ *********************************************************** /
/ ***************************2012************************** /
/*********************************************************** /
/* Script file for batch mode operation for hardware-in-the-loop sim.*/
/* The hardware consist of the RTDS & a SEL-451 IED in closed loop*/
/*********************************************************** /
/* Variable Declaration */
int k,i,loop_counter;
float res[7];
int fault_typeA[3],fault_typeB[3],fault_typeC[3];
/***************************************************/
/* Initialization */
/* Define fault resistances */
res[0] = 0.1;
res[1] = 1.0;
res[2] = 5.0;
res[3] = 10.0;
res[4] = 20.0;
res[5] = 100.0;
/****************************************************/
/* Define fault type */
fault_typeA[0] = 1;
fault_typeA[1] = 0;
fault_typeA[2] = 0;
fault_typeB[0] = 0;
fault_typeB[1] = 2;
fault_typeB[2] = 0;
fault_typeC[0] = 0;
fault_typeC[1] = 0;
fault_typeC[2] = 4;
```

```
/****************************************************/
for (k=0;k<3;k++)
{
/* Set the fault type */
SetSwitch "Subsystem #1 : CTLs : Inputs : AG" = fault_typeA[k];
SetSwitch "Subsystem #1 : CTLs : Inputs : BG" = fault_typeB[k];
SetSwitch "Subsystem #1 : CTLs : Inputs : CG" = fault_typeC[k];
/****************************************************/
for (i=0;i<6;i++)
{
loop_counter++;
/****************************************************/
/* Set the fault resistance */
SetSlider "DraftVariables : Ron" = res[i];
fprintf(stdmsg,"Running Simulation Case Number %d\n",loop_counter);
/****************************************************/
/* Start simulation */
Start;
SUSPEND 0.16667;
PushButton "Subsystem #1 : CTLs : Inputs : Fault";
SUSPEND 0.058;
ReleaseButton "Subsystem #1 : CTLs : Inputs : Fault";
/****************************************************/
ComtradePlotSave"Subsystem#1|CTs","E:\RSCAD\FIRMWARE\Ch\IEEE13NODEwithGTAO\
comtradefile13node.cfg";
/****************************************************/
Stop;
}
}
```