

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **Case Study of Power System State Estimation Based On AI**

SESI PENGAJIAN: 2010/2011

Saya LIANG KAI FENG (870505-02-6089)
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☒

TIDAK TERHAD

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

**23A, TAMAN GANDING,
JALAN LANGGAR
05300, ALOR STAR
KEDAH**

DR. AHMED N. ABD ALLA
(Nama Penyelia)

Tarikh: **29 NOVEMBER 2010**

Tarikh: : **29 NOVEMBER 2010**

- CATATAN:
- * Potong yang tidak berkenaan.
 - ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
 - ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

CASE STUDY OF POWER SYSTEM STATE ESTIMATION
BY USING ARTIFICIAL NEURAL NETWORK

LIANG KAI FENG

This thesis is submitted as partial fulfillment of the requirements for the award of the
Bachelor of Electrical Engineering (Hons.) (Power System)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER 2010

“I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)”

Signature : _____

Name : DR.AHMED N ABD ALLA

Date : 29 NOVEMBER 2010

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : _____

Author : LIANG KAI FENG

Date : 29 NOVEMBER 2010

DEDICATION

*Specially dedicated to
My beloved family, and those who have guided and inspired me
Throughout my journey of learning*

ACKNOWLEDGEMENT

Throughout the development of this project I have gained chances to learn new skills and knowledge. I wish to express my sincere appreciation and gratitude to my supervisor, Dr. Ahmed N. Abd Alla for his continuous guidance, concern, encouragement and advices which gave inspiration in accomplishing my final year project.

Special thanks to University Malaysia Pahang for supporting and providing equipment and information sources that assisted my studies and projects.

My sincere appreciation to the lecturers of Faculty of Electrical and Electronics Engineering who have put in effort to the lectures and always nurture and guide us with precious advices. Thank you for sharing those experiences.

To all my lovely current and ex roommates and friends who always willingly assist and support me throughout my journey of education, you all deserve my wholehearted appreciation. Many thanks.

ABSTRACT

This is a study that mains in Artificial Neural Network technique which introduces approach towards the problem of errors that arise due to the practical equipment and actual measurements in distribution systems. Real time data or the state variables measured in power system are often incorporated with error. This project outputs a software program that performs power system state estimation using artificial intelligence optimization. It was developed using Artificial Neural Network in MATLAB software. This method considers nonlinear characteristics of the practical equipment and actual measurements in distribution systems. It can estimate bus voltage and load angle values at each node by minimizing difference between measured and calculated state variables. This is accomplished by the utilization of load flow analysis program which acts as computerized conventional solution that calculates mathematically the exact target outputs in accordance to the inputs applied. The significant functions of the developed software program also include the accurate estimation of power system state with insufficient input data applied. This project has successfully built a power system state estimation software program that perform accurate state estimation achieving desired outputs even when provided with insufficient input data magnitudes. It helps identify the current operating state of the system on which, security assessment functions and hence contingencies can be analyzed leading to the required corrective actions.

ABSTRAK

Projek ini mengkaji teknik Artificial Neural Network di mana ia menyelesaikan masalah yang disebabkan oleh ralat pengukuran dalam kemudahan di system rangkaian pengagihan kuasa. Angka parameter yang diukur dalam system kuasa sebenar biasanya mengandungi ralat. Project ini bertujuan menghasilkan program perisian yang berfungsi menganggar parameter dalam system kuasa dengan menggunakan teknik kepintaran artifak. Program perisian ini ditulis dalam perisian MATLAB dengan menggunakan teknik Artificial Neural Network. Teknik ini mempertimbangkan aspek- aspek ketidakselarasan kemudahan dalam system rangkaian pengagihan kuasa. Program ini mampu membuat penganggaran nilai voltan bas dan sudut beban pada setiap nod dengan meminimakan perbezaan antara nilai pengukuran dan pengiraan secara theory. Kebolehan ini dicapai melalui penglibatan pelaksanaan program yang menganalisa nilai- nilai pengaliran beban menggunakan cara pengiraan lama dengan menggunakan computer di mana ia memberi keputusan nilai yang tepat berdasarkan teori. Program ini juga berkebolehan untuk membuat penganggaran dan memberi nilai keputusan yang tepat tanpa memerlukan bekalan data yang sempurna. Secara kesuluruhannya, projek ini telah berjaya menghasilkan program perisian penganggaran nilai parameter semasa sistem kuasa yang berkesan mencapai ketepatan nilai keputusan yang tinggi walaupun tanpa dibekalkan data yang memadai. Program perisian ini membantu mengesan nilai semasa system operasi di mana penilaian fungsi keselamatan dalam operasi system kuasa dan justeru kesan- kesan awal kegagalan dapat dianalisis supaya tindakan pembetulan atau pembaikan dapat dilaksanakan.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE
1	INTRODUCTION	1
1.1	Power System State Estimation	2
1.2	Problem Statement	3
1.2.1	Load Flow Analysis	3
1.3	Objectives	4
1.4	Scope Of The Project	5
2	LITERATURE REVIEW	6
2.1	The Utilization of Database in State Determination	8
2.2	Artificial Neural Networks (ANNs)	9
2.2.1	Advantages and Disadvantages of Artificial Neural Networks (ANNs)	11
2.2.2	Mathematical Modeling of ANNs from Biological Model	11
2.2.3	Neural Network Topologies	14
2.3	Neural Network	14
2.3.1	Back-Propagation	14
2.3.2	Radial Basis Function (RBF) Network	15
2.4	Training of ANNs	17

3	METHODOLOGY	20
3.1	Phases	21
3.1.1	Data Collecting Phase	21
3.1.2	Training Phase	21
3.1.3	Testing Phase	22
3.2	The NNTool	23
3.2.1	Data Collecting Phase	23
3.2.2	Training Phase	24
3.3.3	Testing Phase	25
4	RESULT AND DISCUSSION	26
4.1	Graphic User Interface	27
4.2	Data Collection Phase	28
4.2.1	Collection of Single Set Data	28
4.2.2	Varying Single Set of Data	31
4.2.3	Running Load Flow Analysis	33
4.3	Training Phase	41
4.4	Testing Phase	41
4.4.1	First Category	42
4.4.2	Second Category	44
4.4.3	Third Category	47
4.5	Program Flow	51
4.6	State Estimation with Another Neural Network	60

5	CONCLUSION	63
5.1	Conclusion	63
5.2	Recommendation	65
5.3	Costing and Comercialization	66

LIST OF TABLES

TABLE NO.	TITLE	PAGE
4.1	IEEE 30-Bus System	29
4.2	IEEE 30-Bus Line Data	30
4.3	Results by complete sets of input	43
4.4	Results by incomplete sets of input	46
4.5	Results by anonymous sets of input varying 10 times	48
4.6	Results by anonymous sets of input varying 20 times	49
4.7	Results by anonymous sets of input varying 50 times	50

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	The style of Neural Computation	10
2.2	Biological Model (Neuron)	12
2.3	Formula of Biological Model	12
2.4	Mathematical Models(ANN)	13
2.5	A Generalized Network	17
2.6	The Structure of A Neuron	17
3.1	Three Phases of The Project	20
3.2	NNTool	23
3.3	Creating New Network	24
4.1	Power System State Estimator	27
4.2	Overall flow of Software Program	28
4.3	Flow of Varying Data	32
4.4	One Line Diagram for 3 Bus Power System	33
4.5	Program of Newton-Raphson Method	34
4.6	Result of Newton-Raphson Method	35
4.7	Program of Gauss-Seidel Method	36
4.8	Result of Gauss-Seidel Method	37
4.9	Program of Fast-Decouple Method	38
4.10	Result of Fast-Decouple Method	39
4.11	Details of Neural Network Formed	40
4.12	The row 1 till row 29 of resulting incomplete matrix pp with zeros	45
4.13	Flow of data30bus.m	53
4.14	Flow of varydata10.m	53

4.15	Flow of lfaNR.m	54
4.16	Flow of formNN.m	55
4.17	Flow of test.m	55
4.18	Flow of TestEmpty.m	56
4.19	Flow of TestAnonymous1.m	57
4.20	Flow of TestAnonymous2.m	58
4.21	Flow of TestAnonymous3.m	59
4.22a	BackPropagation Iteration On Complete Set of Data	60
4.22b	Radial Basis Network Iteration On Complete Set of Data	60
4.23a	BackPropagation Iteration On Incomplete Set of Data	60
4.23b	Radial Basis Network Iteration On Incomplete Set of Data	61
4.24a	BackPropagation Iteration On Anonymous Set of Data	61
4.24b	Radial Basis Network Iteration On Anonymous Set of Data	61

LIST OF APPENDICES

APPENDIX	TITLE
1	databus30.m
2	varydata10.m
3	lfaNR.m
4	formNN.m(BackPropagation)
5	formRBFNN.m(Radial Basis Network)
6	test.m
7	TestEmpty.m
8	TestAnonymous1.m
9	TestAnonymous2.m
10	TestAnonymous3.m

CHAPTER 1

INTRODUCTION

The advancement in computer and communication technologies has resulted in wide application of the supervisory control and data acquisition (SCADA) system in the modern control centers. SCADA is highly capable and flexible that it deals with large information flows coming from many protective and control devices placed in the bulky electric power systems.

By processing the real- time redundant measures and network parameters available in the SCADA database, the state estimation obtains current states of system. Therefore, the performance of state estimation relies on the accuracy of the measured data as well as the parameters of the networks model.

1.1 Power System State Estimation

Power System State Estimation is a calculation to estimate the power system state by using EMS(Energy Management System). The aim of the state estimation is to get the best estimate of the current system states processing a set of real-time redundant measures and network parameters available in the database. The performance of state estimation, therefore, depends on the accuracy of the measured data as well as the parameters of the network model. The measured data are subject to noise or errors in the metering system and the communication process. Large errors in the analog measurements, so-called bad data, may happen in practice[6].

Network parameters such as impedances of transmission lines may be incorrect as a result of inaccurate manufacturing provided data, error in calibration, etc[2]. In addition, due to the lack of field information and possible errors in calculations, transformer tap positions may be erroneous. The purpose of a state estimator is to filter all these errors to achieve the best possible estimate of the state of the system[8].

Generally, WLS (Weight-Least-Square) estimator or non-Gaussian estimator is used to determine the state of the system. Few estimation is use for example Maximum Likelihood Estimation, General State Estimation is use to minimize the bias of the power system state[2].

1.2 Problem Statement

Despite the convenience provided by the SCADA system, however, there are errors that arise due to the practical equipment and actual measurements in distribution systems. For instance, noise in metering system and communication process, large error in analog measurements that may happen in practice, erroneous transformer tap positions due to the lack of field information and possible errors in calculations, inaccurate manufacturing provided data, error in calibration and so on.

1.2.1 Load Flow Analysis

The conventional way, i.e. Load Flow Analysis requires complete set of data or input and takes time to mathematically perform. Though programs created to replace hand calculation are available nowadays and they successfully save a lot of time, but still, they need complete set of input data in order to run and achieve the desired outputs[12].

Therefore, this study proposes a power system state estimation method using an AI optimization, for example Artificial Neural Network (ANN) which can estimate bus voltage and load angle values at each node by minimizing difference between measured and calculated state variables. This method aims to filter the errors mentioned earlier so that the best possible estimate of the system state is achieved[15].

1.3 Objectives

This project aims to produce a software program that performs power system state estimation with the application of Artificial Neural Network. The software program should output results instantly after the inputs are given or in other words it takes shorter time to perform as compared to the hand calculation method for Load Flow Analysis.

Without requiring complete input parameters data it has to perform state estimation and achieve desired output. It should estimate bus voltage and load angle values at each node by minimizing difference between measured and calculated state variables that it aims to filter the errors at the same time considering the nonlinear characteristics of the practical equipment and actual measurements in distribution systems so that the best possible estimation of the system state is achieved.

1.4 Scope Of The Project

The related scopes of this project are Artificial Intelligence (AI), Artificial Neural Networks (ANNs) and MATLAB software. It involves data collection, training and testing phases. The training phase utilizes supervised learning technique and the weights or strengths of connections in the artificial neural network are automatically adjusted according to some modification rules.

MATLAB Software is utilized where .m file as the location to write program and form linkages between main program and sub programs, also, as the platform where ANN program is trained to be accurate, efficient and user friendly.

Power System Analysis, the Load Flow Analysis that performs to gain information of the power and voltage flow in the buses of the power system network in order to evaluate the performance of power system network as well as to analyze any planning for power system improvement under steady state conditions. It is necessary for planning, operation, economic scheduling and exchange of power between different utility.

CHAPTER 2

LITERATURE REVIEW

Electrical power system consists of complex networks that need continual and comprehensive analysis for the planning, design, and operation in order to assist future plant expansion[8]. In power system analysis, the power flow and voltage flow in power system network can be calculated by using three mathematical techniques, the Newton-Raphson method, Gauss-Seidel method and Fast-decouple method[11].

Newton- Raphson method is more practical and efficient for large power system since the number of iterations is independent of the system size but more functional evaluations are required at each iterations[12]. Fast-decouple method makes use of an approximate version of the Newton-Raphson procedure as an alternative strategy for improving computational when solving large power transmission systems[3].

However, hand computational work is almost impossible to perform analysis on large and complex power system network. Therefore, software designed to carry

out the mathematical calculation, or in other words, take over the hand calculation work, which outputs in short time as compared to the conventional method, is available nowadays[1].

2.1 The Utilization of Database in State Determination

The advancement in computer and communication technologies has resulted in the wide usage of the supervisory control and data acquisition (SCADA) system in the modern control centers.[5] SCADA is highly capable and flexible that it deals with large information flows coming from many protective and control devices placed in the bulky electric power systems. [7]

The information is very useful during events that cause outage. It helps the operator in control centers to identify defective part of the system and to start the restoration process. By processing a set of real time redundant measures and network parameters available in the database, the state estimation gets the best estimate of the current system states. [7]

Therefore, the performance of state estimation relies on the accuracy of the measured data as well as the parameters of the networks model. However, there are errors that arise due to the noise or errors in the metering system and the communication process, large error in analog measurements, also known as bad data that may happen in practice, network parameters, impedances of transmission lines for instance, that may be incorrect data in accordance to the inaccurate manufacturing provided data, error in calibration, etc and the lack of field information and possible errors in calculations that transformer tap positions may be erroneous. [9]

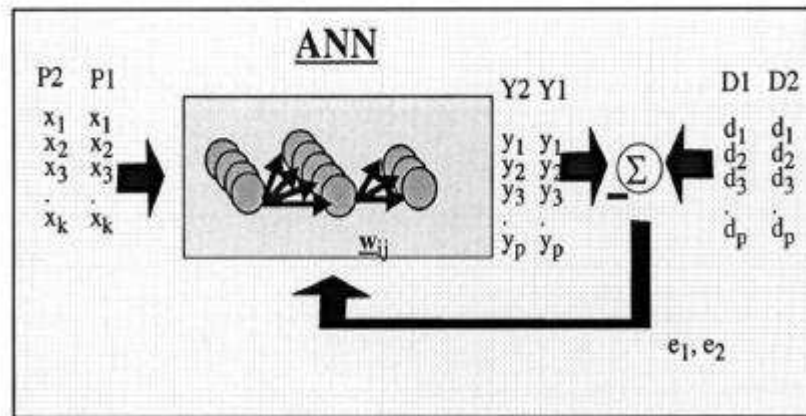
2.2 Artificial Neural Networks (ANNs)

An ANN is most often a nonlinear system that learns to perform a function (an input/output map) from data. It is adaptive, where the system parameters are changed during operation, normally called the training phase. [10] It is built with step-by-step procedure systematically to optimize a performance criterion or to follow some implicit internal constraint, commonly referred to as the learning rule.

The input/output training data are fundamental in neural network technology. They convey the necessary information to discover the optimal operating point. After the training phase the ANN parameters are fixed and the system is deployed to solve the problem at hand. This is called the testing phase. [7]

The nonlinearity of the neural network processing elements (PEs) provides flexibility to the system to achieve practically any desired input/output map. Hence it is said that some Artificial Neural Networks are universal mappers.[13]

For the case of supervised method, an input is given to the neural network while a corresponding target response set is given at the output. Then an error will be composed from the difference between the desired output or response and the system output which is next fed back to the system where it adjusts the parameters of the system systematically according to the learning rule.[7] This process is repeated until the performance is acceptable. The style of computation is shown in Figure 2.1.[4]



The style of neural computation.

Figure 2.1 The style of Neural Computation

The performance of the neural network hinges heavily on the data. Therefore, neural network technology is not a suitable solution for cases where data is insufficient to cover significant portion of the operating conditions or they are noisy. Conversely, it is a good solution to derive an approximate model for conditions where a plenty of data exist but with the problem poorly understood. [13]

Instead of conducting traditional engineering design that exhaustive subsystem specifications and intercommunication protocols are necessary,[4] in artificial neural networks, the designer chooses the network topology, the performance function, the learning rule and the criterion to stop the training phase, but the system adjusts the parameters automatically.

Though it is hard to bring a priori information into the design and it is difficult to incrementally refine the solution when the system does not work in proper way, ANN-based solutions are very time efficient in terms of development and resources. Besides, in many tough problems, it provides performance that is difficult to match with other technologies. Hence, ANNs are emerging as the choice for applications like pattern recognition, prediction, system identification and control.[7]

2.2.1 Advantages and Disadvantages of Artificial Neural Networks (ANNs)

ANNs is a system that takes the operation of biological neural networks as conceptual basis, i.e. it is an emulation of biological neural system. Despite the disadvantages that it is made with, it performs certain tasks that a program made for a common microprocessor is unable to perform. In other words, a neural network can perform tasks that a linear program cannot. [14]

When an element of the neural network fails, its parallel nature enables it to continue without any problem. Besides, it learns and does not need to be reprogrammed. Thus, it can be implemented in any application without any problem. However, the neural network needs training prior to its operation. Its architecture is different from that of a microprocessor; therefore, it needs to be emulated. [15]

In addition, high processing time is required for large neural networks. Artificial neural networks can have different architectures that consequently require different types of algorithms, but it is relatively simpler than to be a complex system.

2.2.2 Mathematical Modeling of ANNs from Biological Model

A biological nervous system consists of neurons as the basic signaling units where each neuron is a discrete cell whose several processes arise from its cell body. The ANNs emerged as circuits that could perform computational tasks with biological neurons as basic conceptual components.

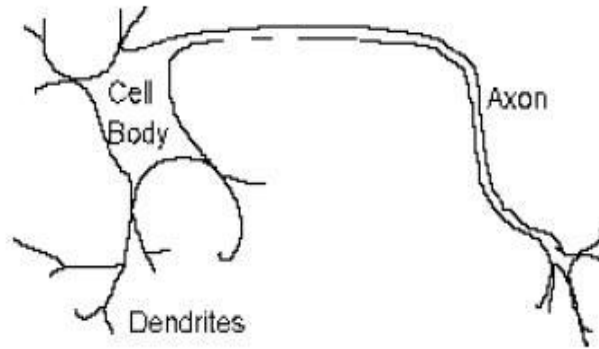


Figure 2.2 Biological model (Neuron)

[Source: <http://www.learnartificialneuralnetworks.com/>]

The neurons or cells as shown in Figure 2 are modeled as processing units where the area of contact between two physically non-touching neurons is called synapse where in this synaptic cleft electric signals are sent through chemical interactions. In a functional model, the synapses are modeled as weights and their values note the connection strength between an input and a neuron.

The inputs are modified by their respective weights before linear combination takes place whereby they are summed up by an adder. Then, an activation function will control the amplitude of the neuron output to a range between 0 and 1, or, -1 and 1. This is mathematically described in the Figure 2.4 below according to the formula shown in Figure 2.3.

$$v_k = \sum_{j=1}^P w_{kj} x_j$$

Figure 2.3 Formula

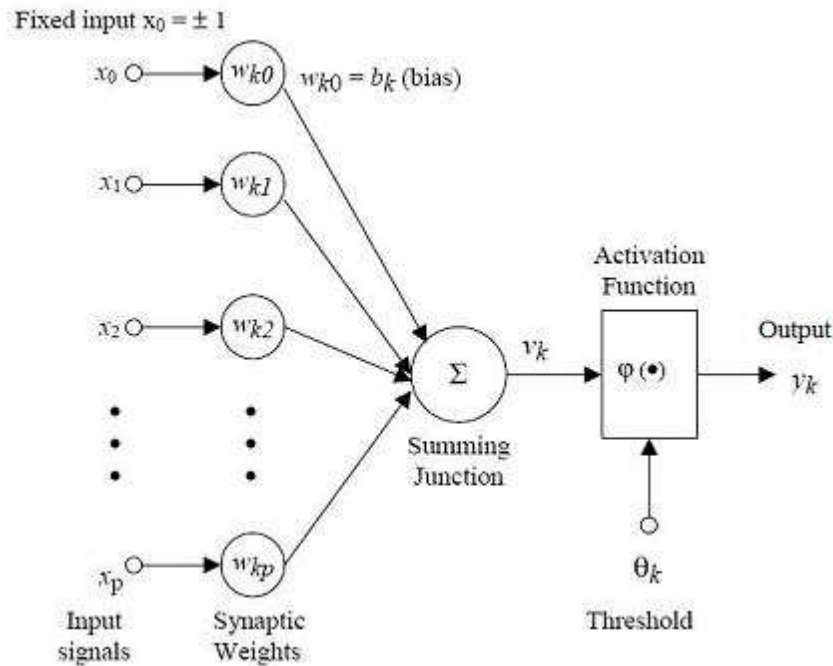


Figure 2.4 Mathematical model (ANNs)

[Source: <http://www.learnartificialneuralnetworks.com/>]

The neuron output, y_k , is the outcome of some activation function on the value of v_k . In short, an artificial neural network is a pool of simple processing units that communicate by sending signals to each other over a large number of weighted connections. Apart from adjusting the weight, each processing units receive input from neighbours or external sources to compute an output signal which is propagated to other units.

There are three types of units in neural systems: input units which receive data from outside the neural network, output units which send data out of the network and hidden units whose input and output signals remain within the network. The system is parallel that computations by many units can be carried out simultaneously.

At the same time during operation, units can be updated either synchronously whereby all units simultaneously update their activation, or, asynchronously whereby

each unit has a probability, which is usually fixed, of updating its activation at a time, t , and only one unit perform at a time.

2.2.3 Neural Network Topologies

There are many pattern types of connections between units and the propagation of data, i.e. radial basis function (RBF) network, feed-forward neural networks, Kohonen self-organizing network, recurrent neural networks and etc [4].

2.3 Neural Network

2.3.1 Back-Propagation

Back-Propagation is a simple neural network which uses multi-layered neural network to be constructed. Back-Propagation is a supervised learning neural network which means it needs a teacher who knows or can calculate, the desired or given input.

If we consider the human brain to be the 'ultimate' neural network, then ideally we would like to build a device which imitates the brain's functions. However, because of limits in our technology, we must settle for a much simpler design. The obvious approach is to design a small electronic device which has a transfer function similar to a biological neuron, and then connect each neuron to many other neurons, using RLC networks to imitate the dendrites, axons, and synapses. This type of

electronic model is still rather complex to implement, and we may have difficulty 'teaching' the network to do anything useful.

Further constraints are needed to make the design more manageable. First, we change the connectivity between the neurons so that they are in distinct layers, such that each neuron in one layer is connected to every neuron in the next layer. Further, we define that signals flow only in one direction across the network, and we simplify the neuron and synapse design to behave as analog comparators being driven by the other neurons through simple resistors. We now have a feed-forward neural network model that may actually be practical to build and use.

Referring to figures 2.5 and 2.6, the network functions as follows: Each neuron receives a signal from the neurons in the previous layer, and each of those signals is multiplied by a separate weight value. The weighted inputs are summed, and passed through a limiting function which scales the output to a fixed range of values. The output of the limiter is then broadcast to all of the neurons in the next layer. So, to use the network to solve a problem, we apply the input values to the inputs of the first layer, allow the signals to propagate through the network, and read the output values.[17]

2.3.2 Radial Basis Function (RBF) Network

As stated in Wikipedia, Radial Basis Function (RBF) Network is a technique used for multidimensional space [16]. It is built into a distance criterion with respect to a center and may be applied in neural network as replacement for sigmoid hidden layer transfer characteristic in Multi- Layer Perceptrons.

RBF networks have two layers of processing [16]. Input is firstly mapped onto each RBF in the hidden layer. The output layer is a linear combination of hidden layer values representing the mean predicted output. The interpretation of this output layer value is like a regression model in statistics.

The output layer is a typical sigmoid function of a linear combination of hidden layer values which represents a posterior probability. Performance can be improved by shrinkage techniques, i.e. ridge regression in classical statistics. It is believed that small parameter values will smooth the output functions in a Bayesian framework.

RBF networks do not suffer from local minima in the same way as Multi-Layer Perceptrons since the only parameters that are adjusted in the learning process are the linear mapping from hidden layer to output layer. The linearity ensures the error surface to be quadratic. Therefore, a minimum is easily found. This can be found in one matrix operation in regression problems. The fixed non-linearity introduced by the sigmoid output function in classification problems is most efficiently dealt with using iteratively re-weighted least squares.[16]

RBF networks require good coverage of the input space by radial basis functions. Centers of RBF are determined with reference to the distribution of the input data without referencing to the prediction task. Hence, representational resources that are irrelevant to the learning task may be wasted on areas of the input space. One of the common solutions is to associate each data point with its own centre. This can make the linear system to be solved in the final layer rather large, and needs shrinkage techniques to avoid over fitting.

The association of each input datum with an RBF introduces naturally to kernel methods such as Support Vector Machines and Gaussian Processes whereby the RBF is the kernel function. These three approaches utilize a non-linear kernel

function to project the input data into a space where the learning problem can be solved using a linear model. [16]

Gaussian Processes, and unlike SVMs, RBF networks are usually trained by maximizing the probability, i.e. minimizing the error of the data under the model in a Maximum Likelihood framework. The SVMs avoid over fitting by maximizing instead a margin. In most classification applications by SVMs, the RBF networks are outperformed. They can be competitive in regression applications when the dimensionality of the input space is relatively small.

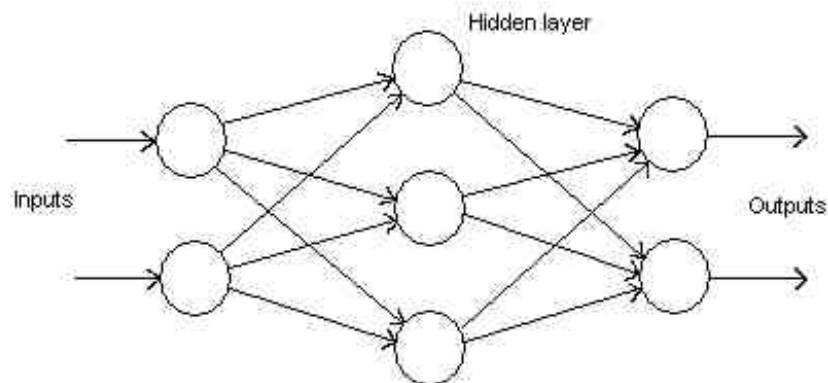


Figure 2.5 A Generalized Network

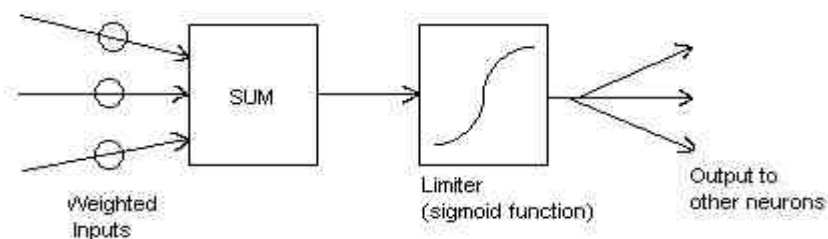


Figure 2.6 The Structure Of A Neuron

2.4 Training of ANNs

When training an ANN with a set of input and output data, we wish to adjust the weights in the ANN, to make the ANN give the same outputs as seen in the training data.[15] On the other hand, we do not want to make the ANN too specific, making it give precise results for the training data, but incorrect results for all other data. When this happens, we say that the ANN has been over-fitted.

Training phase will be the process to configure the neural network on the inputs of data whilst matching the desired output. Different methods can be use to train the ANN which is feeding the ANN with some teaching patterns and let it to change its weight by some learning rules.[13] The learning situations are categorized as supervised learning or Associative learning, Unsupervised learning or Self-organization and Reinforcement learning.[14]

Supervised learning is done to provide the input and output pairs by an external teacher and it was used because we know the correct input data. Whereas for unsupervised learning or self- organization, an output unit is trained to respond to clusters of pattern within the input. The system is supposed to discover statistically salient features of the input population. [13]

For reinforcement learning, an intermediate form of the above two types of learning. For this, the learning machine does some action on the environment and gets a feedback response in return. It then grades its action good, i.e. rewarding, or bad, i.e. punishable based on the environmental response and adjusts its parameters accordingly.[4]

CHAPTER 3

METHODOLOGY

In order to proceed with my project, a powerful computation tool is used to build and train the Artificial Intelligence. MATLAB is an essential tool throughout my whole project to develop the software program. The program was written in the M-file which are shown below.

This project will briefly working in 3 phases, which is data collection phase, training phase and testing phase. The flow chart was show below.

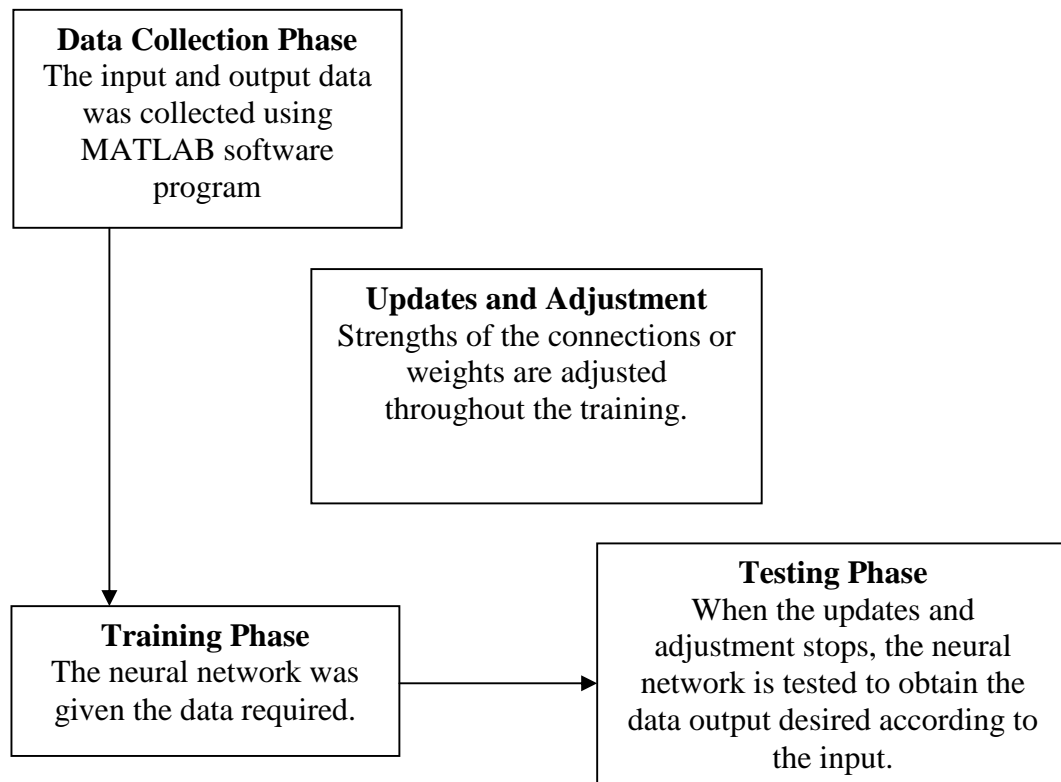


Figure 3.1: Three phases of the project.

3.1 Phases

3.1.1 Data Collecting Phase

Basically data collection phase is where we use the software program to get the required input and output data which are needed to be configured into the neural network. The data is collected from MATLAB by writing programs that randomly change the values of load power and is linked to another program that calculates the outputs in accordance to the varied particular inputs.

There are few software programs which are able to calculate the output works similarly with the conventional way, the Load Flow Analysis. This could be handy as the calculation period was shorten and accurate data can be obtained. These data will be used to train the artificial intelligence to produce the desired outputs.

3.1.2 Training Phase

The neural network is configured such that the desired output would be produced by giving the correct input. Training phase is where the neural network is 'teach' how to configure out the desired output by giving certain inputs. The strengths of the connections within the network can be set by setting the weights explicitly using a priori knowledge (e.g formulas) or by "training" the neural network, i.e. feeding it teaching patterns and letting it changes its weights according to some learning rules.

Supervisory learning method will be use in this project. Neural network was train to get the matching output pattern by giving the required input. During the training, there are adjustments and updates that take place automatically on the weight of ANN each time a new set of input data is given until when it obtains the optimal strengths of connections or weight in the ANN, that it comes to an equilibrium state where it meets the criterion in which the parameters do not change anymore so that in the end the program is able to estimate and give the desired or accurate output when an input is given.

3.1.3 Testing Phase

For the last phase, the accuracy and efficiency of the neural network will be tested. The neural network will be given a set of data with particular dimension corresponding to that input in the training phase. Using the Load Flow Analysis method, we will simulate a set of outputs which will be compared to the neural network output to check for the accuracy and efficiency.

The outputs data here is referred to as target outputs. This means similar sets of inputs which are used to configure the neural network are applied and the expected output of estimation should be accurately near to the target outputs.

The inputs are entered to the program so that it performs state estimation and gives the relevant output values. These outputs will be compared to the output values obtained by the Load Flow Analysis program to evaluate its functionality and results.

3.2 The NNTool

There are another way to configure the neural network, using the NNTool. NNTool can be obtain by running the MATLAB software and the tool is categorized under the toolboxes in MATLAB software.

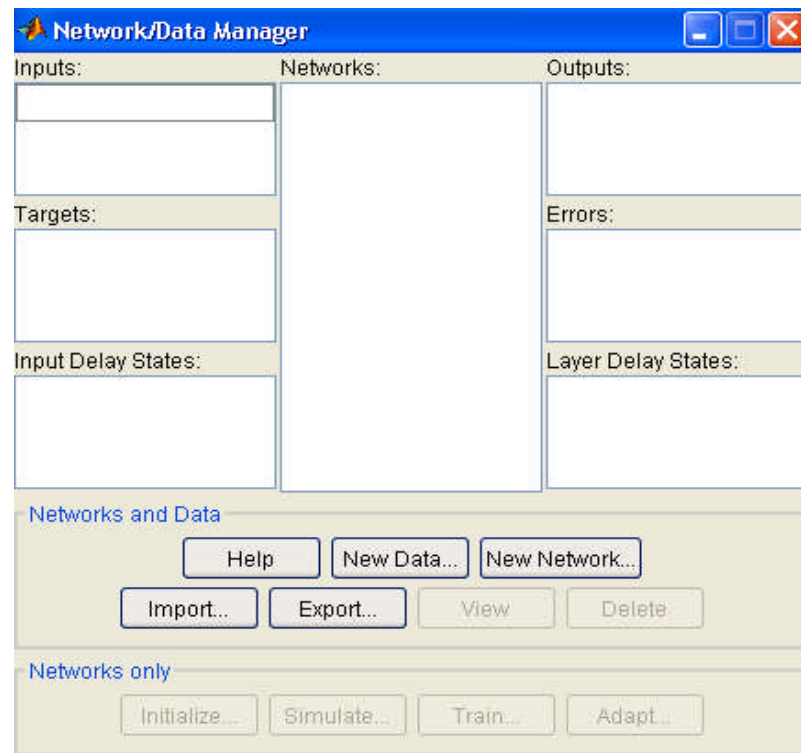


Figure 3.2: NNTool

3.2.1 Data Collecting Phase

The data is defined in the workspace is save so that it can be imported to the NNTool and fill the blank space. The input will be obtain from the Load Flow Analysis input and the target will be the output of it.

3.2.2 Training Phase

In Network/ Data Manager, the neural network is formed by just clicking the New Network button after entering the relevant inputs and targets. Eventually a window as shown in Figure 3.4 will appear with options to select the Network Type and other properties of neural network that are to be formed. After the selection, Create button will lead to the formation of the neural network.

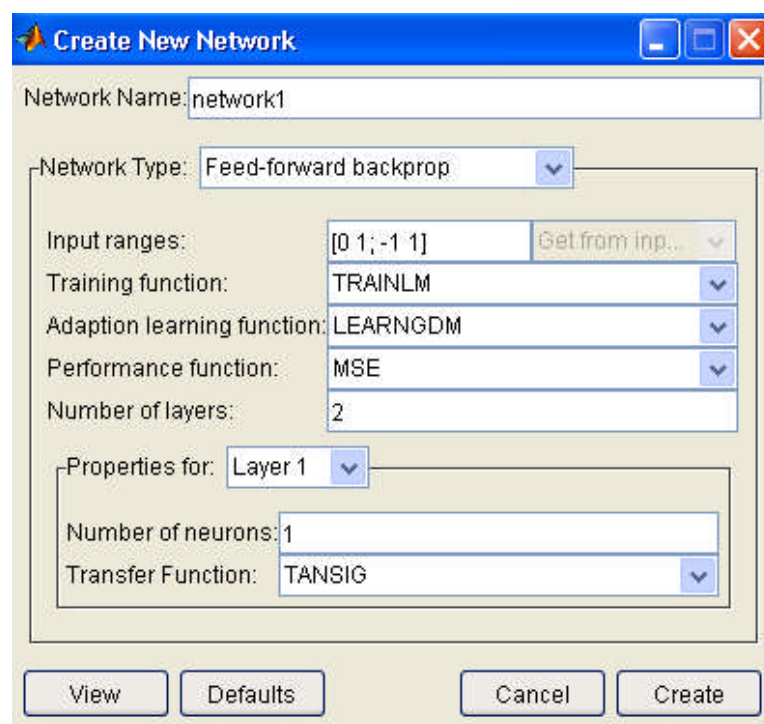


Figure 3.3: Creating New Network

The new neural network that is formed will appear in the Networks blank with the user determined name. Clicking the neural network and View button will lead to another window that appears with options to view the illustrated connections of related layers formed in the neural network.

3.2.3 Testing Phase

Other functions such as Simulate and Weights are also provided to do simulation, i.e. testing, and also to view the weights parameters of the neural network. To perform simulation, the inputs data has to be obtained by running relevant M-File similarly as mentioned so as to appear in the selection box.

After selecting inputs, Simulate button will lead to the result of estimation done by neural network. The outputs can be seen in the Outputs blank. All data used in this project will appear in matrix form.

CHAPTER 4

RESULT AND DISCUSSION

The output data of neural network simulation and estimation will be in the correct range of value output as the original one. By inserting or giving neural network the correct output data, we will do a training for neural network and letting it know that the output data is the desired output. With the correct output as a reference, neural network will be able to simulate a set of output data which will be having the same range of values of data as the original output.

There are three phases to make sure the neural network is trained properly without any mistakes. Data collection phase is where neural network is fed by a correct and accurate output data. While training phase will train the neural network with the correct data output and records it in its memory. Testing phase is where neural network is used to estimate the output data even with data losses or bad datas.

4.1 Graphic User Interface

A graphic user interface or GUI is a type of user interface that allows users to interact with programs in more ways than typing such as computers. A GUI is create in order to ease users to use the program instead of plain codes. Figure 4.1 shows how my State Estimator program looks like.



Figure 4.1 Power System State Estimator

This GUI will control all the functions which will lead to different results. In the image the GUI shows 2 different AI, Feed Forward Back-Propagation and also Radial Basis Network which both of the results will be compared.

4.2 Data Collection Phase

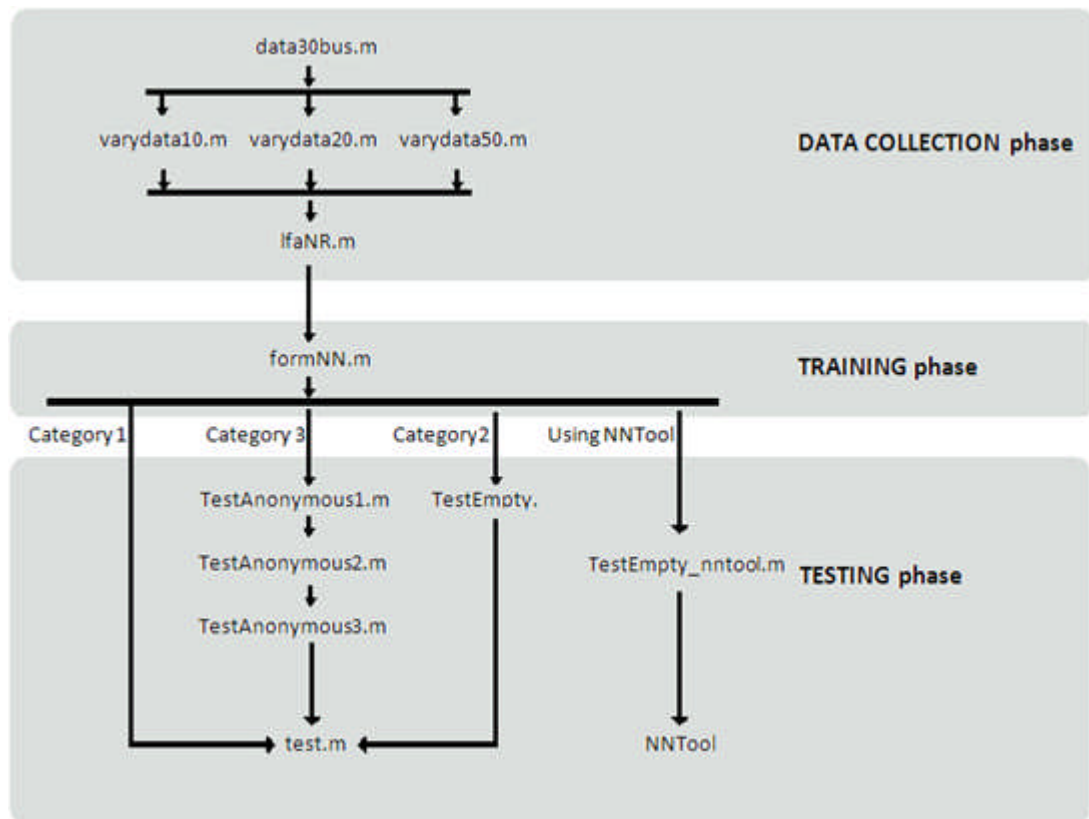


Figure 4.2 Overall flow of software program

For the first phase, i.e. data collection, the sets of input and corresponding output data need to be firstly obtained in order to proceed to the next phase, i.e. training phase.

4.2.1 Collection of Single Set Data

The first step of this project is to collect data from a power system. Below are the data for IEEE 30-Bus System and IEEE Line System which the data will be a reference for neural network training and the output of Load Flow Analysis will be the correct data for neural network to estimate.

				Load		Generator				
Bus No	Bus Code	Voltage Mag	Angle (Degree)	MW	Mvar	MW	Mvar	Qmin	Qmax	Inject Mvar
1	1	1.06	0	0.0	0.0	0.0	0.0	0	0	0
2	2	1.043	0	21.70	12.7	40.0	0.0	-40	50	0
3	0	1.0	0	2.4	1.2	0.0	0.0	0	0	0
4	0	1.06	0	7.6	1.6	0.0	0.0	0	0	0
5	2	1.01	0	94.2	19.0	0.0	0.0	-40	40	0
6	0	1.0	0	0.0	0.0	0.0	0.0	0	0	0
7	0	1.0	0	22.8	10.9	0.0	0.0	0	0	0
8	2	1.01	0	30.0	30.0	0.0	0.0	-10	40	0
9	0	1.0	0	0.0	0.0	0.0	0.0	0	0	0
10	0	1.0	0	5.8	2.0	0.0	0.0	0	0	19
11	2	1.082	0	0.0	0.0	0.0	0.0	-6	24	0
12	0	1.0	0	11.2	7.5	0.0	0.0	0	0	0
13	2	1.071	0	0.0	0.0	0	0	-6	24	0
14	0	1.0	0	6.2	1.6	0	0	0	0	0
15	0	1.0	0	8.2	2.5	0	0	-6	24	0
16	0	1.0	0	3.5	1.8	0	0	0	0	0
17	0	1.0	0	9.0	5.8	0	0	0	0	0
18	0	1.0	0	3.2	0.9	0	0	0	0	0
19	0	1.0	0	9.5	3.4	0	0	0	0	0
20	0	1.0	0	2.2	0.7	0	0	0	0	0
21	0	1.0	0	17.5	11.2	0	0	0	0	0
22	0	1.0	0	0.0	0.0	0	0	0	0	0
23	0	1.0	0	3.2	1.6	0	0	0	0	0
24	0	1.0	0	8.7	6.7	0	0	0	0	4.3
25	0	1.0	0	0.0	0.0	0	0	0	0	0
26	0	1.0	0	3.5	2.3	0	0	0	0	0
27	0	1.0	0	0.0	0.0	0	0	0	0	0
28	0	1.0	0	0.0	0.0	0	0	0	0	0
29	0	1.0	0	2.4	0.9	0	0	0	0	0
30	0	1.0	0	10.6	1.9	0	0	0	0	0

Table 4.1 IEEE 30-Bus System

Bus (n1)	Bus (nr)	R (pu)	X (pu)	$\frac{1}{2} B$ (pu)	Tap Setting
1	2	0.0192	0.0575	0.02640	1
1	3	0.0452	0.1852	0.02040	1
2	4	0.0570	0.1737	0.01840	1
3	4	0.0132	0.0379	0.00420	1
2	5	0.0472	0.1983	0.02090	1
2	6	0.0581	0.1763	0.01870	1
4	6	0.0119	0.0414	0.00450	1
5	7	0.0460	0.1160	0.01020	1
6	7	0.0267	0.0820	0.00850	1
6	8	0.0120	0.0420	0.00450	1
6	9	0.0	0.2080	0.0	0.978
6	10	0.0	0.5560	0.0	0.969
9	11	0.0	0.2080	0.0	1
9	10	0.0	0.1100	0.0	1
4	12	0.0	0.2560	0.0	0.932
12	13	0.0	0.1400	0.0	1
12	14	0.1231	0.2559	0.0	1
12	15	0.0662	0.1304	0.0	1
12	16	0.0945	0.1987	0.0	1
14	15	0.2210	0.1997	0.0	1
16	17	0.0824	0.1923	0.0	1
15	18	0.1073	0.2185	0.0	1
18	19	0.0639	0.1292	0.0	1
19	20	0.0340	0.0680	0.0	1
10	20	0.0936	0.2090	0.0	1
10	17	0.0324	0.0845	0.0	1
10	21	0.0348	0.0749	0.0	1
10	22	0.0727	0.1499	0.0	1
21	22	0.0116	0.0236	0.0	1
15	23	0.1000	0.2020	0.0	1
22	24	0.1150	0.1790	0.0	1
23	24	0.1320	0.2700	0.0	1
24	25	0.1885	0.3292	0.0	1
25	26	0.2544	0.3800	0.0	1
25	27	0.1093	0.2087	0.0	1
28	27	0.0000	0.3960	0.0	0.968
27	20	0.2198	0.4153	0.0	1
27	30	0.3202	0.6027	0.0	1
29	30	0.2399	0.4533	0.0	1
8	28	0.0636	0.2000	0.0214	1
6	28	0.0169	0.0599	0.065	1

Table 4.2 IEEE 30-Bus Line Data

Bus codes of 0, 1 and 2 specify generator bus, reference or slack bus and load bus respectively. However, this data provides only one set values of bus and line parameters of a 30 buses system at particular moment. In order to configure or train a neural network, it needs definitely more than a set of data.

4.2.2 Varying Single Set Data

We will use another MatLab code to vary the values of parameters in the Bus and Line data above into number denoted by 10. The codes will try to vary each data in the Bus and Line data up to 10 times to increase the accuracy of the artificial intelligent, the BackPropagation neural network.

Varying data set up to 10 times to feed the neural network with more inputs and different output so that it will have a set of range of data that is acceptable in theory calculations and explanation. With different data input and output, neural network will be able to get a range of accurate and correct data to perform the estimation.

The variation or deviation of data is done by mathematical operations and function “rand”. This function produces uniformly distributed random numbers. The syntax “rand(n,m)” gives an n-by-m matrix with random entries ranging from 0 to 1. The sequence of the numbers generated is determined by the state of generator.

The state will reset at each start up of MATLAB, therefore the sequence of numbers generated will be the same unless the state is changed. In other words, several continuous runs of a same program without start up taking place in between will output varying values. The flow of the program at each time it runs is illustrated

in the Figure 4.3 showing the formation of final matrices, which are needed as input for the program of load flow analysis at the next stage.

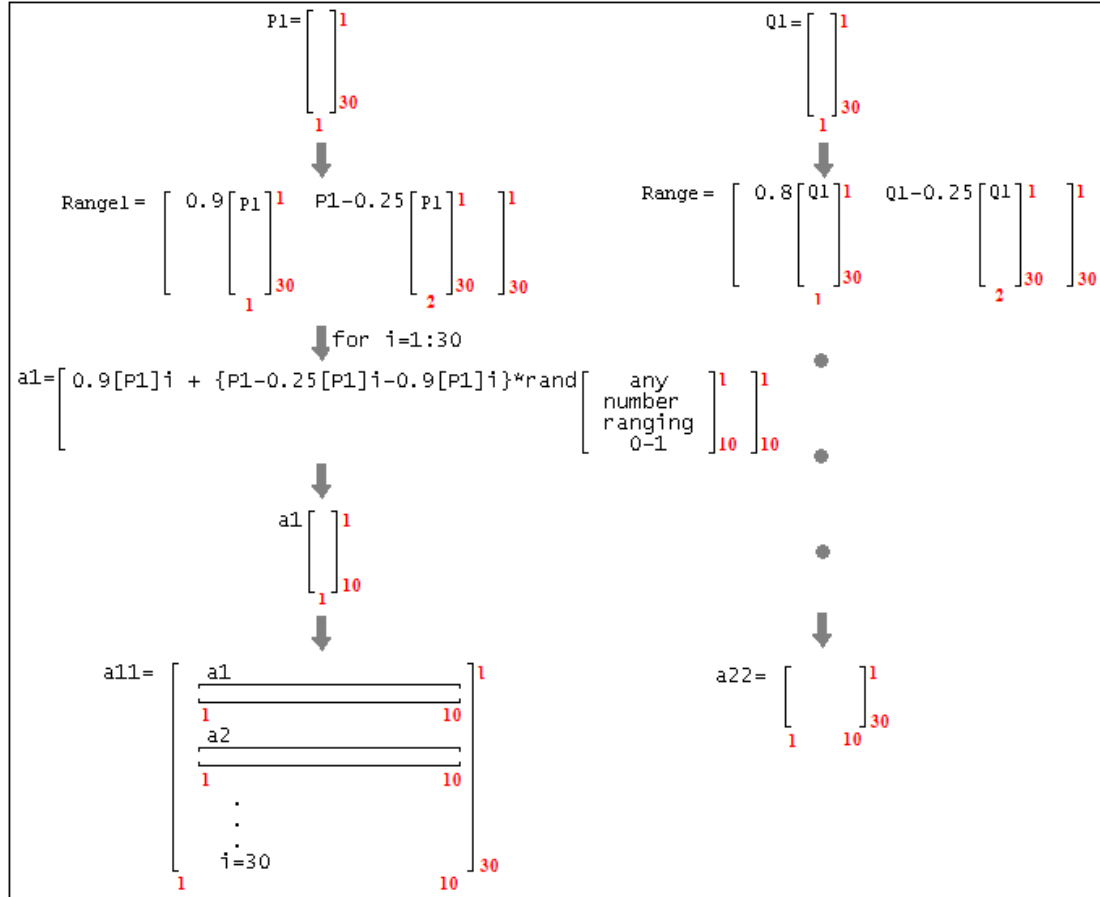


Figure 4.3 Flow of Varying Data

During variation of the real power, a 10 by 1 matrix named as $a1$ is formed for each bus, which is 30 in total, and be transformed into a row collected by matrix $a11$ which turns into a 30 by 10 matrix in the end.

4.2.3 Running Load Flow Analysis

Before proceeding to the build of neural network, load flow analysis is performed to obtain the data of power system due to the varied real and reactive load power. There are three methods for load flow analysis, i.e. Newton-Raphson method, Gauss-Seidel method and Fast-decouple method, provided by Hadi Saadat, 2004, which are developed to run on MATLAB for power system analysis purpose [4].

In order to choose a program to be used for data collection phase, as well as to perform a checking and comparison procedure on the accuracy of these three programs prior to the beginning of configuration and training of neural network, trial has been carried out on all the three mentioned programs based on a 3 buses power system example as shown in Figure 4.4 below.

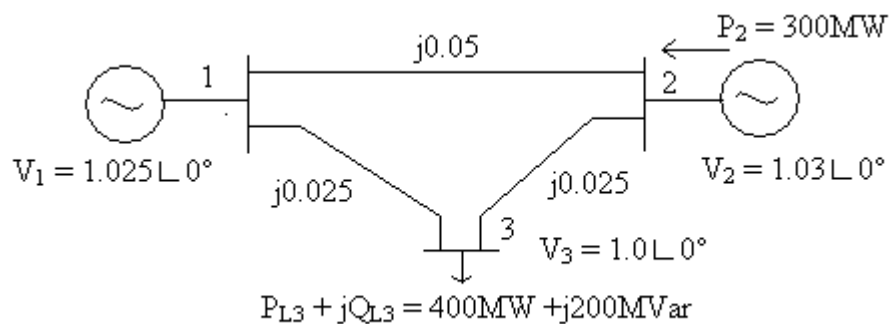


Figure 4.4 One line diagram for 3 bus power system

The program for Newton-Raphson Method and its result are shown in Figure 4.5, and Figure 4.6 respectively.

```

1 basemva = 100; accuracy = 0.001; accel = 1.8; maxiter = 100;
2
3 %      2-BUS TEST SYSTEM
4 %      Bus Bus Voltage Angle ---Load--- -----Generator----- Static Mvar
5 %      No code Mag. Degree MW Mvar MW Mvar Qmin Qmax +Qc/-Ql
6 busdata=[1 1 1.025 0.0 0 0 0.0 0.0 0 0 0
7           2 0 1.030 0.0 0 0 300 0.0 0 0 0
8           3 2 1.000 0.0 400 200 0.0 0.0 0 0 0];
9
10 %
11 %      Bus bus R X 1/2 B Line code
12 %      nl nr p.u. p.u. p.u. = 1 for lines
13 %      > 1 or < 1 tr. tap at bus nl
14 linedata=[1 2 0.0 0.05 0.0 1
15            1 3 0.0 0.025 0.0 1
16            2 3 0.0 0.025 0.0 1];
17
18 lfybus % form the bus admittance matrix
19 lfnewton % Load flow solution by Gauss-Seidel method
20 busout % Prints the power flow solution on the screen
21 lineflow % Computes and displays the line flow and losses

```

Figure 4.5 Program of Newton-Raphson Method

```

>>                                     Power Flow Solution by Newton-Raphson Method
                                     Maximum Power Mismatch = 0.000231479
                                     No. of Iterations = 3

Bus  Voltage  Angle  -----Load-----  ---Generation---  Injected
No.  Mag.      Degree    MW      Mvar      MW      Mvar      Mvar

  1   1.025    0.000     0.000     0.000    100.004    142.914     0.000
  2   1.007    1.425     0.000     0.000    300.000     0.000     0.000
  3   1.000   -2.115    400.000    200.000     0.000    82.638     0.000

Total                                400.000    200.000    400.004    225.551     0.000


                                     Line Flow and Losses

--Line--  Power at bus & line flow  --Line loss--  Transformer
from to    MW      Mvar      MVA      MW      Mvar      tap

  1         100.004  142.914  174.428
    2   -51.320   37.628   63.637    0.000    1.927
    3   151.320  105.293  184.349    0.000    8.087

  2         300.000    0.000  300.000
    1    51.320  -35.701   62.517    0.000    1.927
    3   248.680   35.701  251.229    0.000   15.562

  3        -400.000 -117.362  416.862
    1  -151.320  -97.207  179.853    0.000    8.087
    2  -248.680  -20.139  249.494    0.000   15.562

Total loss                                0.000   25.576
>>

```

Figure 4.6 Results of Newton-Raphson Method

The program for Gauss- Seidel Method and its result are shown in Figure 4.7, and Figure 4.8 respectively.

```

1 basemva = 100; accuracy = 0.001; accel = 1.8; maxiter = 100;
2
3 %      3-BUS TEST SYSTEM
4 %      Bus Bus  Voltage Angle  ---Load---  -----Generator----- Static Mvar
5 %      No  code Mag.    Degree MW    Mvar MW    Mvar Qmin Qmax  +Qc/-Ql
6 busdata=[1  1  1.025  0.0  0  0  0.0  0.0  0  0  0
7           2  0  1.030  0.0  0  0  300  0.0  0  0  0
8           3  2  1.000  0.0  400  200  0.0  0.0  0  0  0];
9 %
10 %      Bus bus  R      X      1/2 B  = 1 for lines
11 %      nl  nr  p.u.   p.u.   p.u.    > 1 or < 1 tr. tap at bus nl
12 linedata=[1  2  0.0  0.05  0.0      1
13            1  3  0.0  0.025  0.0      1
14            2  3  0.0  0.025  0.0      1];
15 lfybus % form the bus admittance matrix
16 lfgauss % Load flow solution by Gauss-Seidel method
17 busout % Prints the power flow solution on the screen
18 lineflow % Computes and displays the line flow and losses
19

```

Figure 4.7 Program of Gauss- Seidel Method

```

>>
Power Flow Solution by Gauss-Seidel Method
Maximum Power Mismatch = 0.000514858
No. of Iterations = 42

Bus Voltage Angle -----Load----- ---Generation--- Injected
No. Mag. Degree MW Mvar MW Mvar Mvar

1 1.025 0.000 0.000 0.000 100.034 142.928 0.000
2 1.007 1.424 0.000 0.000 300.000 0.000 0.000
3 1.000 -2.115 400.000 200.000 0.000 82.652 0.000

Total 400.000 200.000 400.034 225.579 0.000

Line Flow and Losses

--Line-- Power at bus & line flow --Line loss-- Transformer
from to MW Mvar MVA MW Mvar tap

1 100.034 142.928 174.456
2 -51.294 37.626 63.614 0.000 1.926
3 151.336 105.294 184.362 -0.000 8.088

2 300.000 0.000 300.000
1 51.294 -35.700 62.495 0.000 1.926
3 248.643 35.702 251.194 0.000 15.557

3 -400.000 -117.348 416.858
1 -151.336 -97.206 179.866 -0.000 8.088
2 -248.643 -20.145 249.458 0.000 15.557

Total loss -0.000 25.571
>>

```

Figure 4.8 Results of Gauss-Seidel Method

The program for Fast- Decouple Method and its result are shown in Figure 4.9, and Figure 4.10 respectively.

```

1 basemva = 100; accuracy = 0.001; accel = 1.8; maxiter = 100;
2
3 %      3-BUS TEST SYSTEM
4 %      Bus Bus Voltage Angle ---Load--- -----Generator----- Static Mvar
5 %      No code Mag. Degree MW Mvar MW Mvar Qmin Qmax +Qc/-Ql
6 busdata=[1 1 1.025 0.0 0 0 0.0 0.0 0 0 0
7           2 0 1.030 0.0 0 0 300 0.0 0 0 0
8           3 2 1.000 0.0 400 200 0.0 0.0 0 0 0];
9 %
10 %      Bus bus R X 1/2 B = 1 for lines
11 %      nl nr p.u. p.u. p.u. > 1 or < 1 tr. tap at bus nl
12 linedata=[1 2 0.0 0.05 0.0 1
13            1 3 0.0 0.025 0.0 1
14            2 3 0.0 0.025 0.0 1];
15 lfybus % form the bus admittance matrix
16 lfgauss % Load flow solution by Gauss-Seidel method
17 busout % Prints the power flow solution on the screen
18 lineflow % Computes and displays the line flow and losses
19

```

Figure 4.9 Program of Fast- Decouple Method

```

>>                                     Power Flow Solution by Fast Decoupled Method
                                     Maximum Power Mismatch = 0.000176286
                                     No. of Iterations = 4

Bus  Voltage  Angle  -----Load-----  ---Generation---  Injected
No.  Mag.      Degree    MW      Mvar      MW      Mvar      Mvar

  1   1.025    0.000      0.000     0.000    100.008   142.916    0.000
  2   1.007    1.425      0.000     0.000    300.000     0.000    0.000
  3   1.000   -2.115     400.000   200.000     0.000    82.642    0.000

Total                                400.000   200.000   400.008   225.558    0.000

                                     Line Flow and Losses

--Line--  Power at bus & line flow  --Line loss--  Transformer
from to    MW      Mvar      MVA      MW      Mvar      tap

  1         100.008  142.916  174.432
    2   -51.320   37.628   63.637    0.000    1.927
    3   151.320  105.293  184.349    0.000    8.087

  2         300.000    0.000  300.000
    1    51.320  -35.701   62.517    0.000    1.927
    3   248.679   35.701  251.229    0.000   15.562

  3        -400.000 -117.358  416.861
    1  -151.320  -97.207   179.852    0.000    8.087
    2  -248.679  -20.140   249.493    0.000   15.562

Total loss                                0.000   25.576
>>

```

Figure 4.10 Results of Fast- Decouple Method

Conclusion derived from the comparison of the results is that all three methods give similar answers or outputs for the same presented input data, hence any one of these programs is suitable. In this project, program of Newton- Raphson method is chosen to be use as reference in MATLAB.

The Figure 4.11 shows the details of the neural network formed after running the software program. This is obtained by double clicking on the resulting “net”

from the workspace window. After training phase, the neural network should be ready for testing phase.

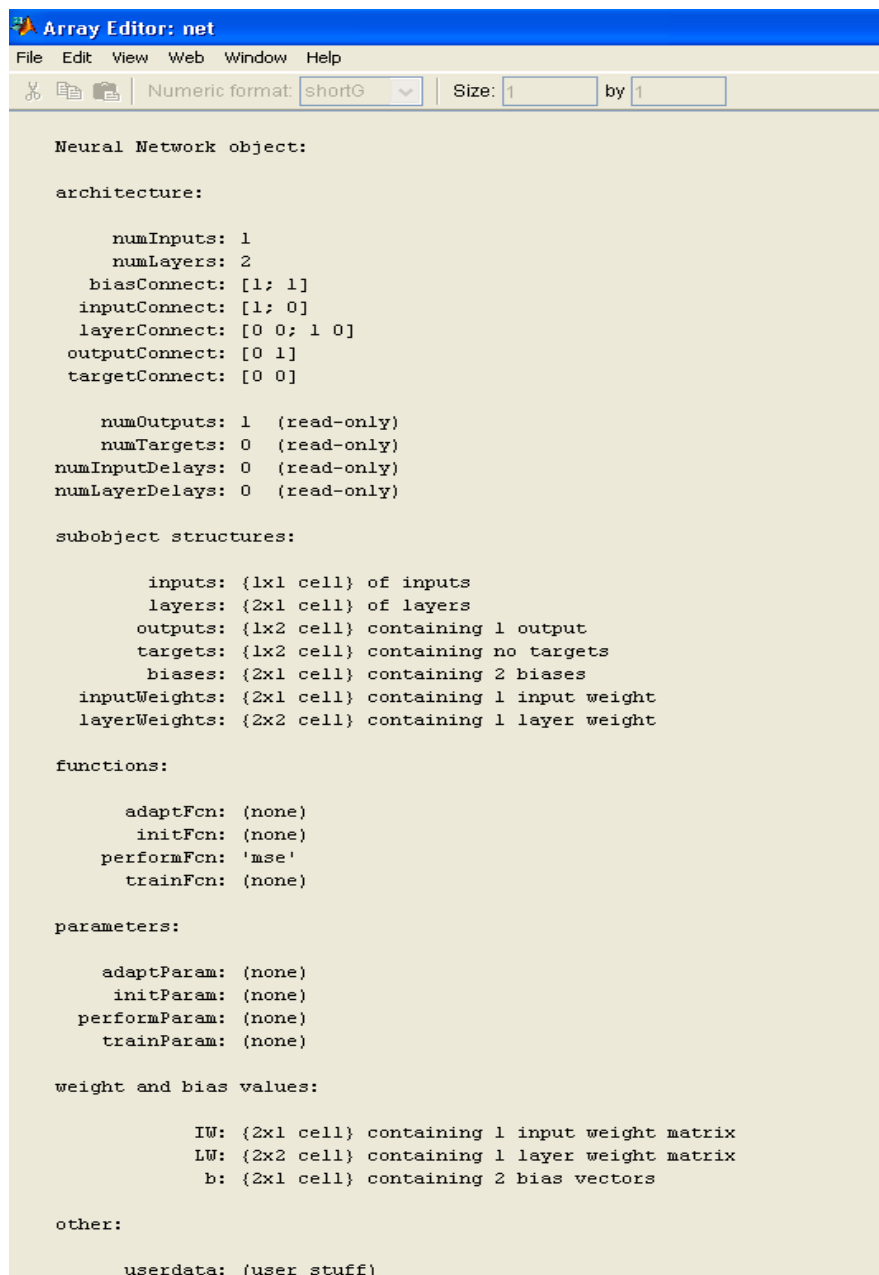


Figure 4.11 Details of neural network formed

4.3 Training Phase

A neural network needs to be “taught” before it knows how to calculate or estimate any values. By referring to Appendix 8, formNN.m, we can see that we will create a new network for BackPropagation neural network and fed the neural network with the correct data collected in the previous phase. “newff” in formNN.m file represents the new network created for Feed-Forwarded BackPropagation neural network, with 1 hidden layer to increase the accuracy of the estimated output.

In forming the neural network, the input data, the first input matrix has to match with the target data, the second input matrix by columns which means their number of columns have to be same. Therefore, both of the input matrices have to be transposed before anything.

4.4 Testing phase

Testing phase aims to test the effectiveness of the neural network in terms of accuracy of the outputs as compared to the target outputs or real outputs. The testing phase tests the accuracy of the outputs generated by the neural network when input data is given, whether complete or incomplete.

This phase generally consists of three categories. The first category is to test whether the software program runs successfully without error and functions to give outputs of estimation when complete set of input is applied. The second category tests whether the software program runs successfully without error and functions to give outputs of estimation when incomplete set of input is applied.

The third category tests whether the software program runs successfully without error and functions to give outputs of estimation when new or anonymous set of input is applied with the increase in number of variation, nc of input set used for training phase to study the change in accuracy of outputs.

4.4.1 First Category

The first category is to test whether the software program runs successfully without error and functions to give outputs of estimation when complete set of input is applied. By referring to Appendix 6, test.m, with an example of input, X, which is the first set or column of similar input as given to train the neural network in the previous phase.

Y denotes the output simulated by the neural network in accordance to the input, X. The number of columns of input, X has to be same as in the training phase where any missing data of an incomplete input set should be replaced by zero. Note that all the programs mentioned earlier run in the according sequence continuously after each command is given, therefore in this example input, X, shown remains and readily appears as the transposed version.

Input: 1 st set of pp			Input: 5 th set of pp			Input: 10 th set of pp		
Target	Result	Accuracy	Target	Result	Accuracy	Target	Result	Accuracy
1.06	1.06	100.000%	1.06	1.06	100.00%	1.06	1.06	100.00%
1.043	1.043	100.000%	1.043	1.043	100.00%	1.043	1.043	100.00%
1.026	1.0262	100.019%	1.026	1.0261	100.01%	1.0261	1.0262	100.01%
1.0177	1.018	100.029%	1.0178	1.018	100.02%	1.0179	1.018	100.01%
1.01	1.01	100.000%	1.01	1.01	100.00%	1.01	1.01	100.00%
1.0157	1.0159	100.020%	1.0158	1.0159	100.01%	1.016	1.0159	99.99%
1.0073	1.0076	100.030%	1.0078	1.0076	99.98%	1.0077	1.0076	99.99%
1.01	1.01	100.000%	1.01	1.01	100.00%	1.01	1.01	100.00%
1.0572	1.0573	100.009%	1.0573	1.0573	100.00%	1.0574	1.0573	99.99%
1.0547	1.0549	100.019%	1.0549	1.0549	100.00%	1.0551	1.0549	99.98%
1.082	1.082	100.000%	1.082	1.082	100.00%	1.082	1.082	100.00%
1.064	1.0643	100.028%	1.0642	1.0643	100.01%	1.064	1.0643	100.03%
1.071	1.071	100.000%	1.071	1.071	100.00%	1.071	1.071	100.00%
1.052	1.0526	100.057%	1.0522	1.0525	100.03%	1.0521	1.0525	100.04%
1.0486	1.049	100.038%	1.0487	1.0488	100.01%	1.0486	1.0489	100.03%
1.0544	1.0547	100.028%	1.0545	1.0546	100.01%	1.0546	1.0546	100.00%
1.0504	1.0507	100.029%	1.0506	1.0505	99.99%	1.0507	1.0506	99.99%
1.0408	1.041	100.019%	1.0407	1.0407	100.00%	1.0407	1.0408	100.01%
1.0386	1.0388	100.019%	1.0384	1.0384	100.00%	1.0386	1.0385	99.99%
1.0415	1.0418	100.029%	1.0416	1.0416	100.00%	1.0418	1.0416	99.98%
1.045	1.0452	100.019%	1.0453	1.0454	100.01%	1.0456	1.0453	99.97%
1.0454	1.0457	100.029%	1.0458	1.0459	100.01%	1.0461	1.0458	99.97%
1.0405	1.0411	100.058%	1.0411	1.0412	100.01%	1.0409	1.0411	100.02%
1.0366	1.0372	100.058%	1.0375	1.0376	100.01%	1.0372	1.0374	100.02%
1.0332	1.0341	100.087%	1.0341	1.0345	100.04%	1.0346	1.0344	99.98%
1.0186	1.0202	100.157%	1.0196	1.0207	100.11%	1.021	1.0205	99.95%
1.0382	1.0389	100.067%	1.039	1.0393	100.03%	1.0395	1.0392	99.97%
1.0149	1.0151	100.020%	1.0151	1.0152	100.01%	1.0153	1.0152	99.99%
0.97867	0.98099	100.237%	0.9819	0.98326	100.14%	0.98418	0.98288	99.87%
0.98656	0.9889	100.237%	0.99003	0.99146	100.14%	0.99238	0.99092	99.85%
0	0	100.000%	0	0	100.00%	0	0	100.00%
-0.073056	-0.072048	98.620%	-0.077053	-0.073552	95.46%	-0.072532	-0.072722	100.26%
-0.11061	-0.10946	98.960%	-0.11469	-0.11094	96.73%	-0.10991	-0.11014	100.21%
-0.13328	-0.13183	98.912%	-0.13825	-0.13363	96.66%	-0.13233	-0.13264	100.23%
-0.19431	-0.19344	99.552%	-0.20831	-0.1965	94.33%	-0.19419	-0.19433	100.07%
-0.15724	-0.15563	98.976%	-0.1636	-0.15797	96.56%	-0.15612	-0.15673	100.39%
-0.18029	-0.17864	99.085%	-0.18925	-0.18164	95.98%	-0.17951	-0.17993	100.23%
-0.16616	-0.16464	99.085%	-0.1734	-0.16746	96.57%	-0.16509	-0.16607	100.59%
-0.20115	-0.19878	98.822%	-0.20694	-0.20091	97.09%	-0.19904	-0.19977	100.37%
-0.21398	-0.21112	98.663%	-0.21857	-0.21264	97.29%	-0.21207	-0.21176	99.85%
-0.21398	-0.21112	98.663%	-0.21857	-0.21264	97.29%	-0.21207	-0.21176	99.85%
-0.22684	-0.22354	98.545%	-0.23167	-0.22534	97.27%	-0.22467	-0.22438	99.87%
-0.22769	-0.2247	98.687%	-0.23284	-0.22661	97.32%	-0.22566	-0.22559	99.97%
-0.22216	-0.21926	98.695%	-0.2274	-0.22128	97.31%	-0.22005	-0.22023	100.08%
-0.22627	-0.22349	98.771%	-0.23183	-0.22591	97.45%	-0.22431	-0.22469	100.17%
-0.23482	-0.23207	98.829%	-0.24037	-0.2343	97.47%	-0.2327	-0.2331	100.17%
-0.23641	-0.23363	98.824%	-0.24213	-0.23612	97.52%	-0.23416	-0.23481	100.28%
-0.23341	-0.23056	98.779%	-0.2389	-0.23277	97.43%	-0.23078	-0.23156	100.34%
-0.23105	-0.22818	98.758%	-0.23636	-0.22985	97.25%	-0.22794	-0.22892	100.43%
-0.23092	-0.22803	98.748%	-0.23618	-0.22969	97.25%	-0.22787	-0.22876	100.39%
-0.23425	-0.2311	98.655%	-0.23894	-0.23256	97.33%	-0.23173	-0.2317	99.99%
-0.23778	-0.23468	98.696%	-0.2424	-0.23575	97.26%	-0.23467	-0.23506	100.17%
-0.2336	-0.23014	98.519%	-0.23824	-0.23121	97.05%	-0.22926	-0.23041	100.50%
-0.24057	-0.23634	98.242%	-0.24529	-0.23716	96.69%	-0.23465	-0.23648	100.78%
-0.22681	-0.22353	98.554%	-0.23143	-0.22474	97.11%	-0.22253	-0.2238	100.57%
-0.16736	-0.16561	98.954%	-0.17373	-0.16793	96.66%	-0.16594	-0.16669	100.45%
-0.29256	-0.28687	98.055%	-0.29366	-0.286	97.39%	-0.28218	-0.28511	101.04%
-0.28543	-0.2797	97.993%	-0.286	-0.2781	97.24%	-0.27444	-0.2776	101.15%
Average accuracy :		99.409%	Average accuracy :		98.53%	Average accuracy :		100.15%
General accuracy :		99.362%						

Table 4.3 Results by complete sets of input

4.4.2 Second Category

The main objective of using the neural network is to estimate the outputs when the input given is incomplete. This category tests the neural network by using the same input as first category but made incomplete prior to the testing. Program TestEmpty.m as shown in Appendix 9 is run before the test.m after the formNN.m. The TestEmpty.m is created to make some values in the input data set, matrix pp to become zero hence providing incomplete input for testing. Parts of the resulting pp matrix are shown in Figure 4.11.

	1	2	3	4	5	6	7	8	9	10
1	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06
2	1.043	1.043	1.043	1.043	1.043	1.043	1.043	1.043	1.043	1.043
3	1.026	1.0262	1.0263	1.0263	1.026	1.0262	1.0261	1.026	1.0261	1.0261
4	1.0177	1.0181	1.0182	1.0182	1.0178	1.018	1.018	1.0178	1.0179	1.0179
5	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
6	1.0157	1.0159	1.0161	1.016	1.0158	1.016	1.016	1.0159	1.0159	1.016
7	1.0073	1.0075	1.0079	1.0075	1.0078	1.0078	1.008	1.0077	1.0075	1.0077
8	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
9	1.0572	1.0572	1.0574	1.0575	1.0573	1.0573	1.0575	1.0574	1.0574	1.0574
10	0	0	0	0	0	0	0	0	0	0
11	1.082	1.082	1.082	1.082	1.082	1.082	1.082	1.082	1.082	1.082
12	1.064	1.0644	1.0643	1.0644	1.0642	1.0644	1.0644	1.0643	1.0644	1.064
13	1.071	1.071	1.071	1.071	1.071	1.071	1.071	1.071	1.071	1.071
14	1.052	1.053	1.0526	1.0527	1.0522	1.0527	1.053	1.0525	1.0529	1.0521
15	1.0486	1.049	1.0491	1.0491	1.0487	1.049	1.0492	1.0492	1.0493	1.0486
16	1.0544	1.0546	1.0547	1.0547	1.0545	1.0546	1.055	1.0548	1.0549	1.0546
17	1.0504	1.0504	1.0509	1.0506	1.0506	1.0504	1.0511	1.0508	1.0509	1.0507
18	1.0408	1.0408	1.0411	1.0409	1.0407	1.0407	1.0407	1.0412	1.0412	1.0407
19	1.0386	1.0384	1.0389	1.0385	1.0384	1.0385	1.0384	1.039	1.039	1.0386
20	0	0	0	0	0	0	0	0	0	0
21	1.045	1.045	1.0454	1.0457	1.0453	1.0452	1.0455	1.0457	1.0452	1.0456
22	1.0454	1.0455	1.0459	1.0462	1.0458	1.0457	1.046	1.0462	1.0457	1.0461
23	1.0405	1.0412	1.0412	1.0416	1.0411	1.0411	1.0413	1.0415	1.0416	1.0409
24	1.0366	1.0371	1.0373	1.038	1.0375	1.0373	1.0375	1.0379	1.0376	1.0372
25	1.0332	1.034	1.0343	1.0348	1.0341	1.0348	1.034	1.0341	1.0344	1.0346
26	1.0186	1.02	1.021	1.0214	1.0196	1.0214	1.0204	1.0202	1.0203	1.021
27	1.0382	1.0389	1.039	1.0392	1.039	1.0396	1.0384	1.0386	1.0392	1.0395
28	1.0149	1.0152	1.0153	1.0153	1.0151	1.0153	1.0151	1.015	1.0152	1.0153
29	0.97867	.98183	.98114	0.982	0.9819	.98425	0.9774	.97786	.98228	0.98418

Figure 4.12 The row 1 till row 29 of resulting incomplete matrix pp with zeros

Then, the test.m is run to estimate the outputs. The results are tabulated as below in Table 4.3. The target is the set of values of matrix tt corresponding to the input set of matrix pp. The result columns are the output values estimated by software program shown by matrix Y.

Input: 1 st set of pp			Input: 5 th set of pp			Input: 10 th set of pp		
Target	Result	Accuracy	Target	Result	Accuracy	Target	Result	Accuracy
1.06	1.06	100.000%	1.06	1.06	100.00%	1.06	1.06	100.00%
1.043	1.043	100.000%	1.043	1.043	100.00%	1.043	1.043	100.00%
1.026	1.0262	100.019%	1.026	1.0262	100.02%	1.0261	1.0262	100.01%
1.0177	1.018	100.029%	1.0178	1.018	100.02%	1.0179	1.018	100.01%
1.01	1.01	100.000%	1.01	1.01	100.00%	1.01	1.01	100.00%
1.0157	1.0159	100.020%	1.0158	1.0159	100.01%	1.016	1.0159	99.99%
1.0073	1.0076	100.030%	1.0078	1.0076	99.98%	1.0077	1.0076	99.99%
1.01	1.01	100.000%	1.01	1.01	100.00%	1.01	1.01	100.00%
1.0572	1.0573	100.009%	1.0573	1.0573	100.00%	1.0574	1.0573	99.99%
1.0547	1.0548	100.009%	1.0549	1.0548	99.99%	1.0551	1.0548	99.97%
1.082	1.082	100.000%	1.082	1.082	100.00%	1.082	1.082	100.00%
1.064	1.0642	100.019%	1.0642	1.0642	100.00%	1.064	1.0642	100.02%
1.071	1.071	100.000%	1.071	1.071	100.00%	1.071	1.071	100.00%
1.052	1.0526	100.057%	1.0522	1.0526	100.04%	1.0521	1.0526	100.05%
1.0486	1.0492	100.057%	1.0487	1.0492	100.05%	1.0486	1.0492	100.06%
1.0544	1.0548	100.038%	1.0545	1.0548	100.03%	1.0546	1.0548	100.02%
1.0504	1.0509	100.048%	1.0506	1.0509	100.03%	1.0507	1.0509	100.02%
1.0408	1.0413	100.048%	1.0407	1.0413	100.06%	1.0407	1.0413	100.06%
1.0386	1.0391	100.048%	1.0384	1.0391	100.07%	1.0386	1.0391	100.05%
1.0415	1.0419	100.038%	1.0416	1.0419	100.03%	1.0418	1.0419	100.01%
1.045	1.045	100.000%	1.0453	1.045	99.97%	1.0456	1.045	99.94%
1.0454	1.0455	100.010%	1.0458	1.0455	99.97%	1.0461	1.0455	99.94%
1.0405	1.041	100.048%	1.0411	1.041	99.99%	1.0409	1.041	100.01%
1.0366	1.0369	100.029%	1.0375	1.0369	99.94%	1.0372	1.0369	99.97%
1.0332	1.0337	100.048%	1.0341	1.0337	99.96%	1.0346	1.0337	99.91%
1.0186	1.0197	100.108%	1.0196	1.0197	100.01%	1.021	1.0197	99.87%
1.0382	1.0385	100.029%	1.039	1.0385	99.95%	1.0395	1.0385	99.90%
1.0149	1.015	100.010%	1.0151	1.015	99.99%	1.0153	1.015	99.97%
0.97867	0.97896	100.030%	0.9819	0.97896	99.70%	0.98418	0.97896	99.47%
0.98656	0.98658	100.002%	0.99003	0.98658	99.65%	0.99238	0.98658	99.42%
0	0	100.000%	0	0	100.00%	0	0	100.00%
-0.073056	-0.070486	96.482%	-0.077053	-0.070486	91.48%	-0.072532	-0.070486	97.18%
-0.11061	-0.10793	97.577%	-0.11469	-0.10793	94.11%	-0.10991	-0.10793	98.20%
-0.13328	-0.12997	97.517%	-0.13825	-0.12997	94.01%	-0.13233	-0.12997	98.22%
-0.19431	-0.19008	97.823%	-0.20831	-0.19008	91.25%	-0.19419	-0.19008	97.88%
-0.15724	-0.15323	97.450%	-0.1636	-0.15323	93.66%	-0.15612	-0.15323	98.15%
-0.18029	-0.1755	97.343%	-0.18925	-0.1755	92.73%	-0.17951	-0.1755	97.77%
-0.16616	-0.16178	97.364%	-0.1734	-0.16178	93.30%	-0.16509	-0.16178	98.00%
-0.20115	-0.19658	97.728%	-0.20694	-0.19658	94.99%	-0.19904	-0.19658	98.76%
-0.22402	-0.21915	97.826%	-0.2295	-0.21915	95.49%	-0.22139	-0.21915	98.99%
-0.20115	-0.19658	97.728%	-0.20694	-0.19658	94.99%	-0.19904	-0.19658	98.76%
-0.21398	-0.20953	97.920%	-0.21857	-0.20953	95.86%	-0.21207	-0.20953	98.80%
-0.21398	-0.20953	97.920%	-0.21857	-0.20953	95.86%	-0.21207	-0.20953	98.80%
-0.22684	-0.22169	97.730%	-0.23167	-0.22169	95.69%	-0.22467	-0.22169	98.67%
-0.22769	-0.22272	97.817%	-0.23284	-0.22272	95.65%	-0.22566	-0.22272	98.70%
-0.22216	-0.2172	97.767%	-0.2274	-0.2172	95.51%	-0.22005	-0.2172	98.70%
-0.22627	-0.22102	97.680%	-0.23183	-0.22102	95.34%	-0.22431	-0.22102	98.53%
-0.23482	-0.22977	97.849%	-0.24037	-0.22977	95.59%	-0.2327	-0.22977	98.74%
-0.23641	-0.23108	97.745%	-0.24213	-0.23108	95.44%	-0.23416	-0.23108	98.68%
-0.23341	-0.22827	97.798%	-0.2389	-0.22827	95.55%	-0.23078	-0.22827	98.91%
-0.23105	-0.22644	98.005%	-0.23636	-0.22644	95.80%	-0.22794	-0.22644	99.34%
-0.23092	-0.2263	97.999%	-0.23618	-0.2263	95.82%	-0.22787	-0.2263	99.31%
-0.23425	-0.22956	97.998%	-0.23894	-0.22956	96.07%	-0.23173	-0.22956	99.06%
-0.23778	-0.23353	98.213%	-0.2424	-0.23353	96.34%	-0.23467	-0.23353	99.51%
-0.2336	-0.22893	98.001%	-0.23824	-0.22893	96.09%	-0.22926	-0.22893	99.86%
-0.24057	-0.2354	97.851%	-0.24529	-0.2354	95.97%	-0.23465	-0.2354	100.32%
-0.22681	-0.22216	97.950%	-0.23143	-0.22216	95.99%	-0.22253	-0.22216	99.83%
-0.16736	-0.16321	97.520%	-0.17373	-0.16321	93.94%	-0.16594	-0.16321	98.35%
-0.29256	-0.28725	98.185%	-0.29366	-0.28725	97.82%	-0.28218	-0.28725	101.80%
-0.28543	-0.28083	98.388%	-0.286	-0.28083	98.19%	-0.27444	-0.28083	102.33%
Average accuracy :		98.933%	Average accuracy :		97.63%	Average accuracy :		99.48%
General accuracy :		98.682%						

Table 4.4 Results by incomplete sets of input

As shown in the above table, the general accuracy obtained is 98.682%. This means that the software program perform highly accurate state estimation even when the input given is incomplete. The main objective of this project is achieved.

4.4.3 Third Category

The accuracy of the values output by neural network depends on the number of input sets used for training. Therefore, by increasing the value of varying data, it will show the difference or expected improvement in output accuracy. For this category, the inputs used are firstly varied so as to try running the software program to do estimation on different or anonymous set of input.

In this case, 3 anonymous test will be undergo to check the accuracy of the neural network. First, it will run a test with different data from bus and line. The input data are some values that are out of range which will give bad output if using normal Load Flow Analysis calculation. Note that the bus data and line data are varied by multiplication of 1.25 as shown in Appendix. Then the program flow is as usual or similar to that of the described data collection phase.

The 2nd test will vary the values of bus data and line data up to 10 times to check the accuracy of the output. Eventually, 3rd test runs the Newton Raphson load flow analysis to simulate the target output, which will determine the accuracy and the efficiency of the neural network.

The result of doing 3 sets of anonymous input data varying for 10, 20 and 50 times is shown as follow.

Input: 1 st set of M			Input: 5 th set of M			Input: 10 th set of M		
Target	Result	Accuracy	Target	Result	Accuracy	Target	Result	Accuracy
1.325	1.06	80.000%	1.325	1.06	80.00%	1.325	1.06	80.00%
1.3038	1.043	79.997%	1.3038	1.043	80.00%	1.3038	1.043	80.00%
1.2855	1.0262	79.829%	1.2849	1.0262	79.87%	1.2847	1.0262	79.88%
1.2752	1.018	79.831%	1.2745	1.018	79.87%	1.2742	1.018	79.89%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.2726	1.0159	79.829%	1.2722	1.0159	79.85%	1.2719	1.0159	79.87%
1.2621	1.0076	79.835%	1.2613	1.0076	79.89%	1.2611	1.0076	79.90%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.3209	1.0573	80.044%	1.3207	1.0573	80.06%	1.3203	1.0573	80.08%
1.3156	1.0548	80.176%	1.3153	1.0548	80.19%	1.3148	1.0548	80.23%
1.3525	1.082	80.000%	1.3525	1.082	80.00%	1.3525	1.082	80.00%
1.3304	1.0642	79.991%	1.3299	1.0642	80.02%	1.3295	1.0642	80.05%
1.3387	1.071	80.003%	1.3387	1.071	80.00%	1.3387	1.071	80.00%
1.316	1.0526	79.985%	1.3149	1.0526	80.05%	1.314	1.0526	80.11%
1.311	1.0492	80.031%	1.31	1.0492	80.09%	1.3094	1.0492	80.13%
1.3168	1.0548	80.103%	1.3167	1.0548	80.11%	1.3161	1.0548	80.15%
1.3106	1.0509	80.185%	1.3105	1.0509	80.19%	1.3097	1.0509	80.24%
1.3005	1.0413	80.069%	1.2995	1.0413	80.13%	1.2985	1.0413	80.19%
1.2975	1.0391	80.085%	1.2962	1.0391	80.17%	1.2953	1.0391	80.22%
1.3011	1.0419	80.078%	1.3001	1.0419	80.14%	1.2992	1.0419	80.20%
1.3036	1.045	80.163%	1.3036	1.045	80.16%	1.3029	1.045	80.21%
1.3043	1.0455	80.158%	1.3042	1.0455	80.16%	1.3034	1.0455	80.21%
1.3005	1.041	80.046%	1.2994	1.041	80.11%	1.2986	1.041	80.16%
1.2945	1.0369	80.100%	1.2939	1.0369	80.14%	1.2925	1.0369	80.22%
1.2928	1.0337	79.958%	1.2923	1.0337	79.99%	1.2906	1.0337	80.09%
1.2758	1.0197	79.926%	1.2754	1.0197	79.95%	1.2724	1.0197	80.14%
1.3	1.0385	79.885%	1.2994	1.0385	79.92%	1.2982	1.0385	80.00%
1.2736	1.015	79.695%	1.2732	1.015	79.72%	1.2728	1.015	79.75%
1.2306	0.97896	79.551%	1.23	0.97896	79.59%	1.2264	0.97896	79.82%
1.2411	0.98658	79.492%	1.2401	0.98658	79.56%	1.2366	0.98658	79.78%
0	0	100.000%	0	0	100.00%	0	0	100.00%
-0.07079	-0.070486	99.571%	-0.07726	-0.070488	91.23%	-0.075889	-0.070486	92.88%
-0.1081	-0.10793	99.843%	-0.11506	-0.10793	93.80%	-0.11455	-0.10793	94.22%
-0.13	-0.12997	99.977%	-0.13851	-0.12998	93.84%	-0.13788	-0.12997	94.26%
-0.18933	-0.19008	100.396%	-0.21023	-0.19008	90.42%	-0.20567	-0.19008	92.42%
-0.15327	-0.15323	99.974%	-0.16332	-0.15323	93.82%	-0.16263	-0.15323	94.22%
-0.17532	-0.1755	100.103%	-0.19109	-0.17551	91.85%	-0.18866	-0.1755	93.02%
-0.16162	-0.16178	100.099%	-0.17137	-0.16178	94.40%	-0.17101	-0.16178	94.60%
-0.19572	-0.19658	100.439%	-0.20537	-0.19658	95.72%	-0.20596	-0.19658	95.45%
-0.21792	-0.21915	100.564%	-0.22736	-0.21915	96.39%	-0.22862	-0.21915	95.86%
-0.19572	-0.19658	100.439%	-0.20537	-0.19658	95.72%	-0.20596	-0.19658	95.45%
-0.20704	-0.20953	101.203%	-0.21885	-0.20953	95.74%	-0.21922	-0.20953	95.58%
-0.20704	-0.20953	101.203%	-0.21885	-0.20953	95.74%	-0.21922	-0.20953	95.58%
-0.21888	-0.22169	101.284%	-0.23122	-0.22169	95.88%	-0.23217	-0.22169	95.49%
-0.22012	-0.22272	101.181%	-0.23229	-0.22272	95.88%	-0.23297	-0.22272	95.60%
-0.21597	-0.2172	100.570%	-0.22634	-0.2172	95.96%	-0.22734	-0.2172	95.54%
-0.22058	-0.22102	100.199%	-0.23007	-0.22103	96.07%	-0.23144	-0.22102	95.50%
-0.22781	-0.22977	100.860%	-0.23927	-0.22977	96.03%	-0.24078	-0.22977	95.43%
-0.22938	-0.23108	100.741%	-0.24083	-0.23108	95.95%	-0.2425	-0.23108	95.29%
-0.22652	-0.22827	100.773%	-0.23739	-0.22827	96.16%	-0.23894	-0.22827	95.53%
-0.22479	-0.22644	100.734%	-0.23356	-0.22644	96.95%	-0.23524	-0.22644	96.26%
-0.2245	-0.2263	100.802%	-0.23351	-0.22631	96.92%	-0.23518	-0.2263	96.22%
-0.22618	-0.22956	101.494%	-0.23805	-0.22956	96.43%	-0.23916	-0.22956	95.99%
-0.2298	-0.23353	101.623%	-0.24026	-0.23354	97.20%	-0.24226	-0.23353	96.40%
-0.22549	-0.22893	101.526%	-0.23594	-0.22894	97.03%	-0.23835	-0.22893	96.05%
-0.23082	-0.2354	101.984%	-0.24141	-0.2354	97.51%	-0.2451	-0.2354	96.04%
-0.21944	-0.22216	101.240%	-0.22985	-0.22216	96.65%	-0.2318	-0.22216	95.84%
-0.16343	-0.16321	99.865%	-0.17345	-0.16321	94.10%	-0.17309	-0.16321	94.29%
-0.2803	-0.28725	102.479%	-0.28957	-0.28726	99.20%	-0.29467	-0.28726	97.49%
-0.27204	-0.28083	103.231%	-0.28208	-0.28084	99.56%	-0.28683	-0.28083	97.91%
Average accuracy :		90.391%	Average accuracy :		87.87%	Average accuracy :		87.77%
General accuracy :		88.675%						

Table 4.5 Results by anonymous sets of input varying 10 times.

Input: 1 st set of M			Input: 5 th set of M			Input: 10 th set of M		
Target	Result	Accuracy	Target	Result	Accuracy	Target	Result	Accuracy
1.325	1.06	80.000%	1.325	1.06	80.00%	1.325	1.06	80.00%
1.3038	1.043	79.997%	1.3038	1.043	80.00%	1.3038	1.043	80.00%
1.285	1.026	79.844%	1.2851	1.026	79.84%	1.2853	1.026	79.83%
1.2746	1.0178	79.853%	1.2747	1.0178	79.85%	1.2749	1.0178	79.83%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.2723	1.0158	79.840%	1.2723	1.0158	79.84%	1.2722	1.0158	79.85%
1.2616	1.0074	79.851%	1.2618	1.0074	79.84%	1.2611	1.0074	79.88%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.3207	1.0571	80.041%	1.3206	1.0571	80.05%	1.3207	1.0571	80.04%
1.3153	1.0545	80.172%	1.3151	1.0545	80.18%	1.3153	1.0545	80.17%
1.3525	1.082	80.000%	1.3525	1.082	80.00%	1.3525	1.082	80.00%
1.3299	1.0642	80.021%	1.3302	1.0642	80.00%	1.3302	1.0642	80.00%
1.3387	1.071	80.003%	1.3387	1.071	80.00%	1.3387	1.071	80.00%
1.3144	1.0525	80.075%	1.3152	1.0525	80.03%	1.3153	1.0525	80.02%
1.3097	1.0488	80.079%	1.3101	1.0488	80.05%	1.31	1.0488	80.06%
1.3167	1.0544	80.079%	1.3167	1.0544	80.08%	1.3169	1.0544	80.07%
1.3104	1.0504	80.159%	1.3105	1.0504	80.15%	1.3108	1.0504	80.13%
1.2994	1.0408	80.099%	1.2998	1.0408	80.07%	1.2995	1.0408	80.09%
1.2965	1.0385	80.100%	1.2968	1.0385	80.08%	1.2962	1.0385	80.12%
1.3003	1.0415	80.097%	1.3004	1.0415	80.09%	1.2999	1.0415	80.12%
1.3036	1.0448	80.147%	1.3028	1.0448	80.20%	1.3034	1.0448	80.16%
1.3041	1.0453	80.155%	1.3034	1.0453	80.20%	1.304	1.0453	80.16%
1.2991	1.0409	80.125%	1.299	1.0409	80.13%	1.2994	1.0409	80.11%
1.2935	1.037	80.170%	1.2927	1.037	80.22%	1.2933	1.037	80.18%
1.2921	1.0341	80.033%	1.2911	1.0341	80.09%	1.291	1.0341	80.10%
1.275	1.0203	80.024%	1.2727	1.0203	80.17%	1.274	1.0203	80.09%
1.2994	1.0389	79.952%	1.2989	1.0389	79.98%	1.2979	1.0389	80.04%
1.2733	1.0151	79.722%	1.2732	1.0151	79.73%	1.2729	1.0151	79.75%
1.2305	0.98205	79.809%	1.2288	0.98205	79.92%	1.2217	0.98205	80.38%
1.2407	0.99056	79.839%	1.2392	0.99056	79.94%	1.2318	0.99056	80.42%
0	0	100.000%	0	0	100.00%	0	0	100.00%
-0.076344	-0.0743	97.323%	-0.074932	-0.0743	99.16%	-0.070287	-0.0743	105.71%
-0.11477	-0.11165	97.282%	-0.11279	-0.11165	98.99%	-0.10808	-0.11165	103.30%
-0.13818	-0.13458	97.395%	-0.13576	-0.13458	99.13%	-0.12996	-0.13458	103.55%
-0.2102	-0.20101	95.628%	-0.20605	-0.20101	97.55%	-0.18845	-0.20101	106.66%
-0.16338	-0.15893	97.276%	-0.16042	-0.15893	99.07%	-0.15343	-0.15893	103.58%
-0.19046	-0.18432	96.776%	-0.18625	-0.18432	98.96%	-0.1763	-0.18432	104.55%
-0.17237	-0.16777	97.331%	-0.16885	-0.16777	99.36%	-0.16142	-0.16777	103.93%
-0.20521	-0.20175	98.314%	-0.20303	-0.20175	99.37%	-0.19621	-0.20175	102.82%
-0.22708	-0.22405	98.666%	-0.22531	-0.22405	99.44%	-0.21858	-0.22405	102.50%
-0.20521	-0.20175	98.314%	-0.20303	-0.20175	99.37%	-0.19621	-0.20175	102.82%
-0.21786	-0.21319	97.856%	-0.21519	-0.21319	99.07%	-0.20865	-0.21319	102.18%
-0.21786	-0.21319	97.856%	-0.21519	-0.21319	99.07%	-0.20865	-0.21319	102.18%
-0.23084	-0.22565	97.752%	-0.22767	-0.22565	99.11%	-0.2212	-0.22565	102.01%
-0.23177	-0.22695	97.920%	-0.22895	-0.22695	99.13%	-0.22273	-0.22695	101.89%
-0.22555	-0.22175	98.315%	-0.22347	-0.22175	99.23%	-0.21669	-0.22175	102.34%
-0.22975	-0.22624	98.472%	-0.22752	-0.22624	99.44%	-0.22072	-0.22624	102.50%
-0.23873	-0.2343	98.144%	-0.23611	-0.2343	99.23%	-0.2303	-0.2343	101.74%
-0.24012	-0.23617	98.355%	-0.23754	-0.23617	99.42%	-0.23222	-0.23617	101.70%
-0.23686	-0.23309	98.408%	-0.23446	-0.23309	99.42%	-0.22892	-0.23309	101.82%
-0.23333	-0.23086	98.941%	-0.23253	-0.23086	99.28%	-0.22528	-0.23086	102.48%
-0.23328	-0.23067	98.881%	-0.23233	-0.23067	99.29%	-0.22519	-0.23067	102.43%
-0.23754	-0.2331	98.131%	-0.23535	-0.2331	99.04%	-0.22861	-0.2331	101.96%
-0.23999	-0.23674	98.646%	-0.23882	-0.23674	99.13%	-0.2321	-0.23674	102.00%
-0.23583	-0.23202	98.384%	-0.23473	-0.23202	98.85%	-0.22879	-0.23202	101.41%
-0.24155	-0.23775	98.427%	-0.24165	-0.23775	98.39%	-0.23459	-0.23775	101.35%
-0.22969	-0.22554	98.193%	-0.22799	-0.22554	98.93%	-0.22314	-0.22554	101.08%
-0.17367	-0.16873	97.156%	-0.17072	-0.16873	98.83%	-0.16385	-0.16873	102.98%
-0.28942	-0.28766	99.392%	-0.28892	-0.28766	99.56%	-0.29124	-0.28766	98.77%
-0.28167	-0.27948	99.222%	-0.28109	-0.27948	99.43%	-0.28382	-0.27948	98.47%
Average accuracy :		89.051%	Average accuracy :		89.58%	Average accuracy :		91.21%
General accuracy :		89.946%						

Table 4.6 Results by anonymous sets of input varying 20 times.

Input: 1 st set of M			Input: 5 th set of M			Input: 10 th set of M		
Target	Result	Accuracy	Target	Result	Accuracy	Target	Result	Accuracy
1.325	1.06	80.000%	1.325	1.06	80.00%	1.325	1.06	80.00%
1.3038	1.043	79.997%	1.3038	1.043	80.00%	1.3038	1.043	80.00%
1.2849	1.026	79.851%	1.2848	1.026	79.86%	1.2851	1.026	79.84%
1.2744	1.0178	79.865%	1.2744	1.0178	79.87%	1.2747	1.0178	79.85%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.272	1.0158	79.858%	1.272	1.0158	79.86%	1.2723	1.0158	79.84%
1.2612	1.0076	79.892%	1.2616	1.0076	79.87%	1.2619	1.0076	79.85%
1.2625	1.01	80.000%	1.2625	1.01	80.00%	1.2625	1.01	80.00%
1.3202	1.0572	80.079%	1.3203	1.0572	80.07%	1.3206	1.0572	80.05%
1.3146	1.0548	80.237%	1.3146	1.0548	80.24%	1.3151	1.0548	80.21%
1.3525	1.082	80.000%	1.3525	1.082	80.00%	1.3525	1.082	80.00%
1.3299	1.0642	80.021%	1.3297	1.0642	80.03%	1.33	1.0642	80.02%
1.3387	1.071	80.003%	1.3387	1.071	80.00%	1.3387	1.071	80.00%
1.3144	1.0521	80.044%	1.314	1.0521	80.07%	1.3152	1.0521	80.00%
1.3098	1.0488	80.073%	1.3095	1.0488	80.09%	1.3098	1.0488	80.07%
1.3159	1.0545	80.135%	1.3162	1.0545	80.12%	1.3165	1.0545	80.10%
1.3095	1.0505	80.221%	1.3099	1.0505	80.20%	1.3102	1.0505	80.18%
1.299	1.0407	80.115%	1.299	1.0407	80.12%	1.2989	1.0407	80.12%
1.2955	1.0385	80.162%	1.296	1.0385	80.13%	1.2957	1.0385	80.15%
1.2993	1.0416	80.166%	1.2996	1.0416	80.15%	1.2997	1.0416	80.14%
1.3024	1.0451	80.244%	1.3023	1.0451	80.25%	1.3031	1.0451	80.20%
1.3031	1.0456	80.239%	1.3029	1.0456	80.25%	1.3038	1.0456	80.20%
1.2987	1.041	80.157%	1.2984	1.041	80.18%	1.2993	1.041	80.12%
1.2925	1.0372	80.248%	1.292	1.0372	80.28%	1.2936	1.0372	80.18%
1.2904	1.034	80.130%	1.29	1.034	80.16%	1.2915	1.034	80.06%
1.2734	1.0196	80.069%	1.2718	1.0196	80.17%	1.2734	1.0196	80.07%
1.2974	1.0389	80.076%	1.2975	1.0389	80.07%	1.2989	1.0389	79.98%
1.2726	1.0151	79.766%	1.2727	1.0151	79.76%	1.2731	1.0151	79.73%
1.2205	0.98133	80.404%	1.2224	0.98133	80.28%	1.2285	0.98133	79.88%
1.231	0.98992	80.416%	1.2323	0.98992	80.33%	1.2382	0.98992	79.95%
0	0	100.000%	0	0	100.00%	0	0	100.00%
-0.072111	-0.074688	103.574%	-0.075444	-0.074688	99.00%	-0.073546	-0.074688	101.55%
-0.11115	-0.11209	100.846%	-0.11363	-0.11209	98.64%	-0.11147	-0.11209	100.56%
-0.1337	-0.135	100.972%	-0.13673	-0.135	98.73%	-0.13411	-0.135	100.66%
-0.19285	-0.2022	104.848%	-0.20419	-0.2022	99.03%	-0.19672	-0.2022	102.79%
-0.15792	-0.15946	100.975%	-0.16141	-0.15946	98.79%	-0.15837	-0.15946	100.69%
-0.18013	-0.18468	102.526%	-0.18642	-0.18468	99.07%	-0.18124	-0.18468	101.90%
-0.16705	-0.16823	100.706%	-0.16987	-0.16823	99.03%	-0.16742	-0.16823	100.48%
-0.20186	-0.20295	100.540%	-0.20502	-0.20295	98.99%	-0.20153	-0.20295	100.70%
-0.22485	-0.2256	100.334%	-0.22783	-0.2256	99.02%	-0.2241	-0.2256	100.67%
-0.20186	-0.20295	100.540%	-0.20502	-0.20295	98.99%	-0.20153	-0.20295	100.70%
-0.2142	-0.215	100.373%	-0.21784	-0.215	98.70%	-0.21401	-0.215	100.46%
-0.2142	-0.215	100.373%	-0.21784	-0.215	98.70%	-0.21401	-0.215	100.46%
-0.22724	-0.22806	100.361%	-0.23099	-0.22806	98.73%	-0.22638	-0.22806	100.74%
-0.22803	-0.22893	100.395%	-0.23169	-0.22893	98.81%	-0.22794	-0.22893	100.43%
-0.22303	-0.22355	100.233%	-0.22603	-0.22355	98.90%	-0.2223	-0.22355	100.56%
-0.22754	-0.22809	100.242%	-0.2301	-0.22809	99.13%	-0.22669	-0.22809	100.62%
-0.2359	-0.23637	100.199%	-0.23886	-0.23637	98.96%	-0.23577	-0.23637	100.25%
-0.23813	-0.23785	99.882%	-0.24037	-0.23785	98.95%	-0.23754	-0.23785	100.13%
-0.23487	-0.23474	99.945%	-0.23739	-0.23474	98.88%	-0.23405	-0.23474	100.29%
-0.23189	-0.23254	100.280%	-0.23502	-0.23254	98.94%	-0.23105	-0.23254	100.64%
-0.23169	-0.23237	100.293%	-0.23486	-0.23237	98.94%	-0.23081	-0.23237	100.68%
-0.23449	-0.2353	100.345%	-0.23811	-0.2353	98.82%	-0.2336	-0.2353	100.73%
-0.23804	-0.23892	100.370%	-0.24173	-0.23892	98.84%	-0.23663	-0.23892	100.97%
-0.23444	-0.23418	99.889%	-0.2382	-0.23418	98.31%	-0.23291	-0.23418	100.55%
-0.23988	-0.24061	100.304%	-0.24483	-0.24061	98.28%	-0.2394	-0.2406	100.50%
-0.2288	-0.22733	99.358%	-0.23196	-0.22733	98.00%	-0.22662	-0.22733	100.31%
-0.16869	-0.16938	100.409%	-0.17201	-0.16938	98.47%	-0.16887	-0.16938	100.30%
-0.29839	-0.29068	97.416%	-0.29935	-0.29068	97.10%	-0.28725	-0.29068	101.19%
-0.29032	-0.28226	97.224%	-0.29236	-0.28226	96.55%	-0.28046	-0.28226	100.64%
Average accuracy :		90.270%	Average accuracy :		89.39%	Average accuracy :		90.37%
General accuracy :		90.010%						

Table 4.7 Results by anonymous sets of input varying 50 times

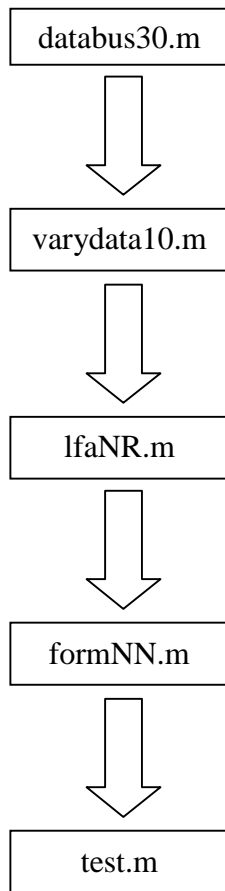
As observed from the results shown in Table 4.4, Table 4.5 and Table 4.6, the general accuracies are 88.675%, 89.946% and 90.010% accordingly which increases with the increase of the varying amount. This means that the accuracy of the estimation improves as more input sets are used to train and form the neural network. The more input data is provided for training phase, the better the adjustment of weights of the relationship between input and targeted output formed in the neural network therefore strengthen the accuracy of estimation performed.

The general accuracies of the results by anonymous sets of input for amount of 10, 20 and 50 as shown previously are however lower than that of the results shown in Table 4.1 and Table 4.2 which are achieved by similar set of input matrix pp that are used for training phase. This is expected as the range of the tested anonymous inputs is different from that of the input used to train the neural network earlier.

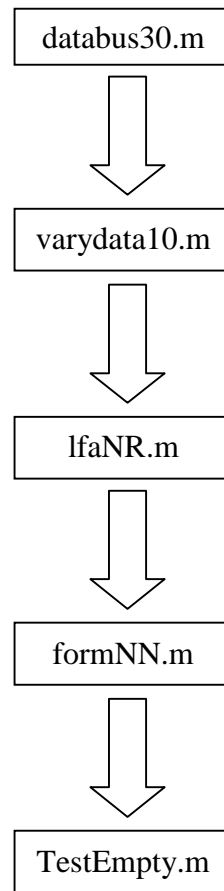
4.5 Program Flow

The Program will start by reading the 30-Bus system and the line system. And it will vary data up to 10 times for neural network to have a range or accuracy and values. Load flow analysis is executed after that to calculate the output of the total 30 bus system. It will then train the neural network for estimation purpose. Below are the flow chart of how a complete set of data, incomplete set of data and anonymous set of data execute.

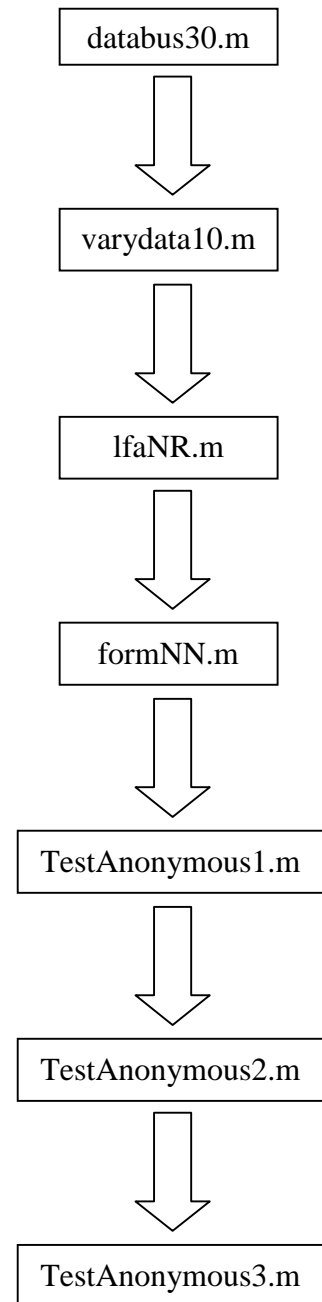
Complete Set of Data



Incomplete Set of Data



Anonymous Set of



Above are the program flow where when you click on the “Start” button on the GUI, it will follow the above flow and execute the program. The output will be show as a text in the command window in MATLAB. While the details of flows that will execute in the codes are stated next.

data30bus.m

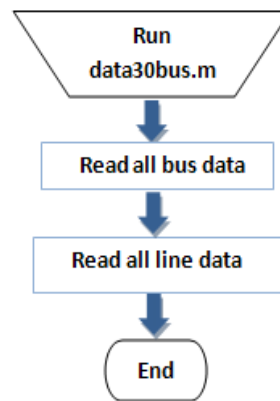


Figure 4.13 Flow of data30bus.m

varydata10.m

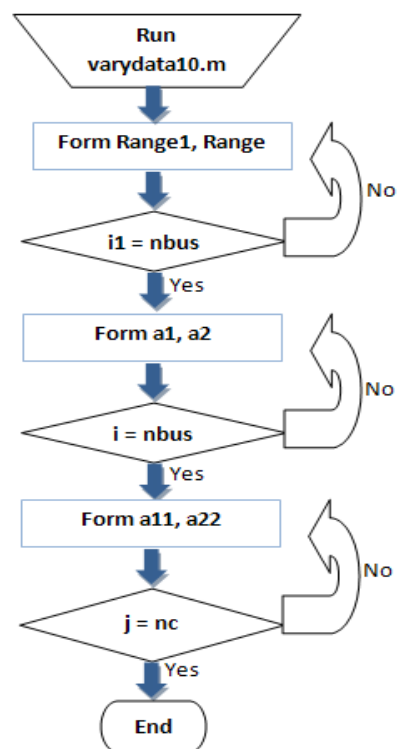


Figure 4.14 Flow of varydata10.m

IfaNR.m

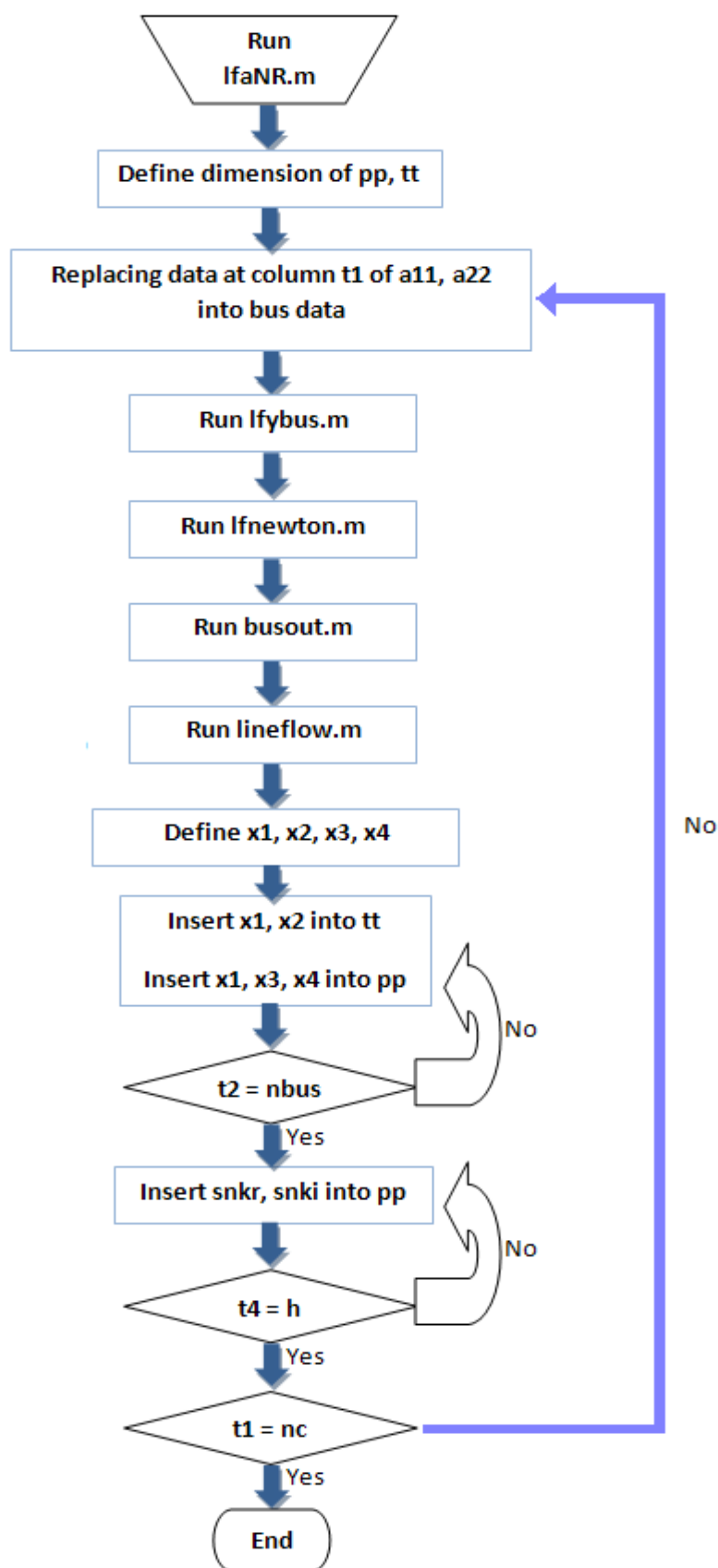


Figure 4.15 Flow of IfaNR.m

formNN.m

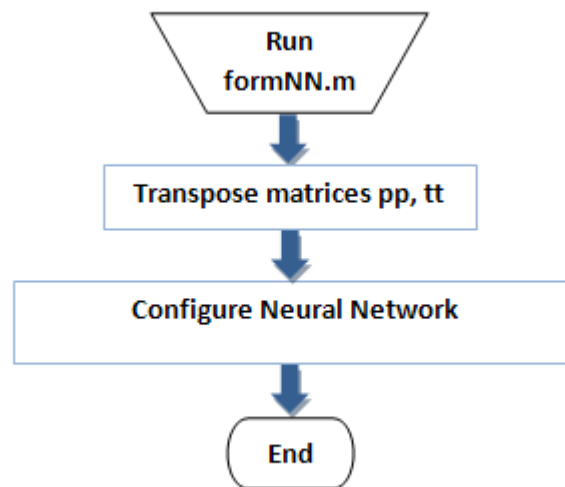


Figure 4.16 Flow of formNN.m

test.m

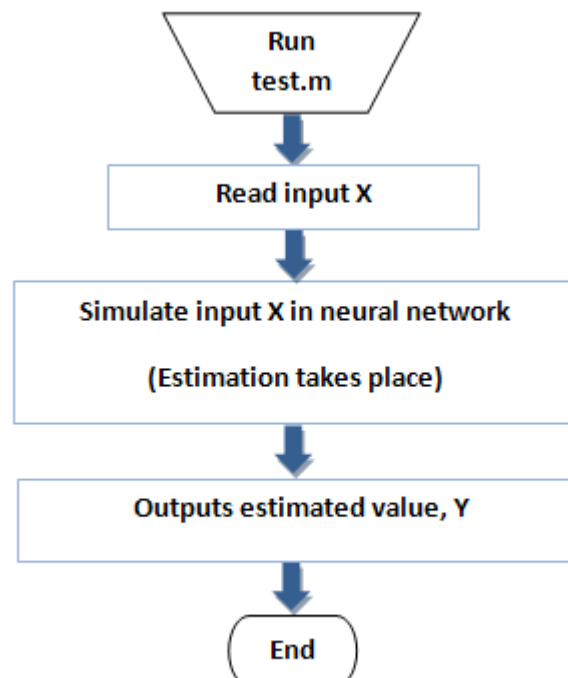
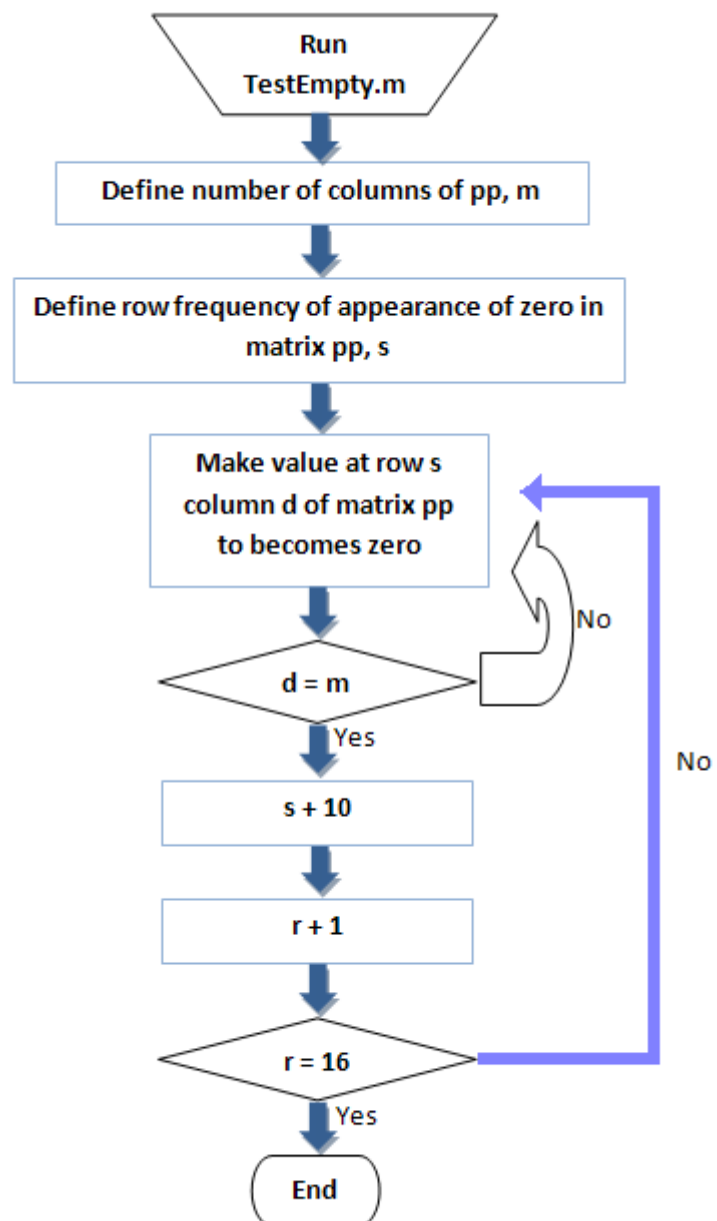
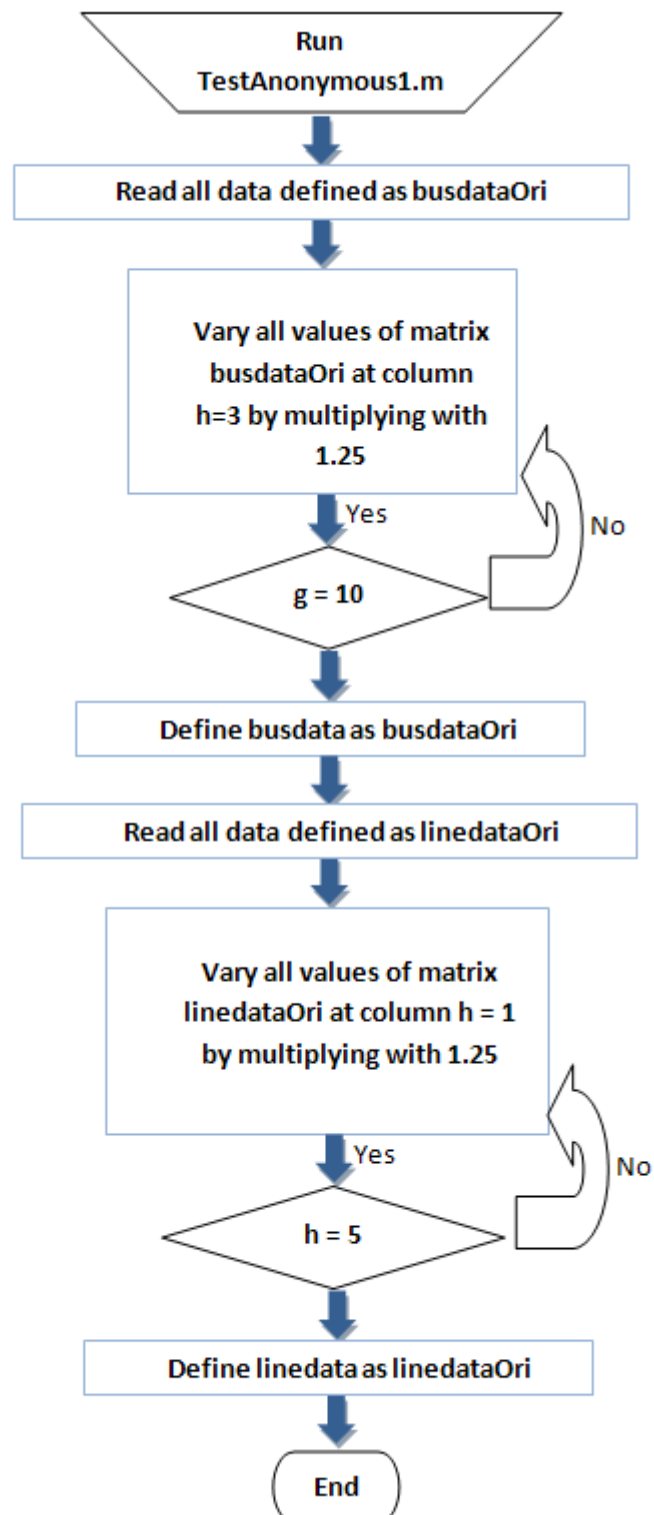
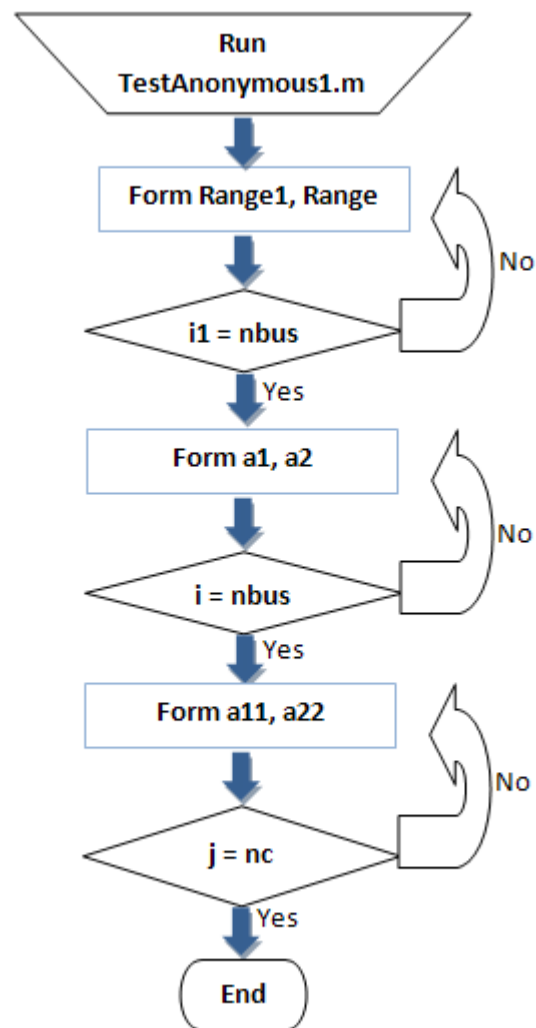


Figure 4.17 Flow of test.m

TestEmpty.m**Figure 4.18** Flow of TestEmpty.m

TestAnonymous1.m**Figure 4.19** Flow of Test Anonymous1.m

TestAnonymous2.m**Figure 4.20** Flow of TestAnonymous2.m

TestAnonymous3.m

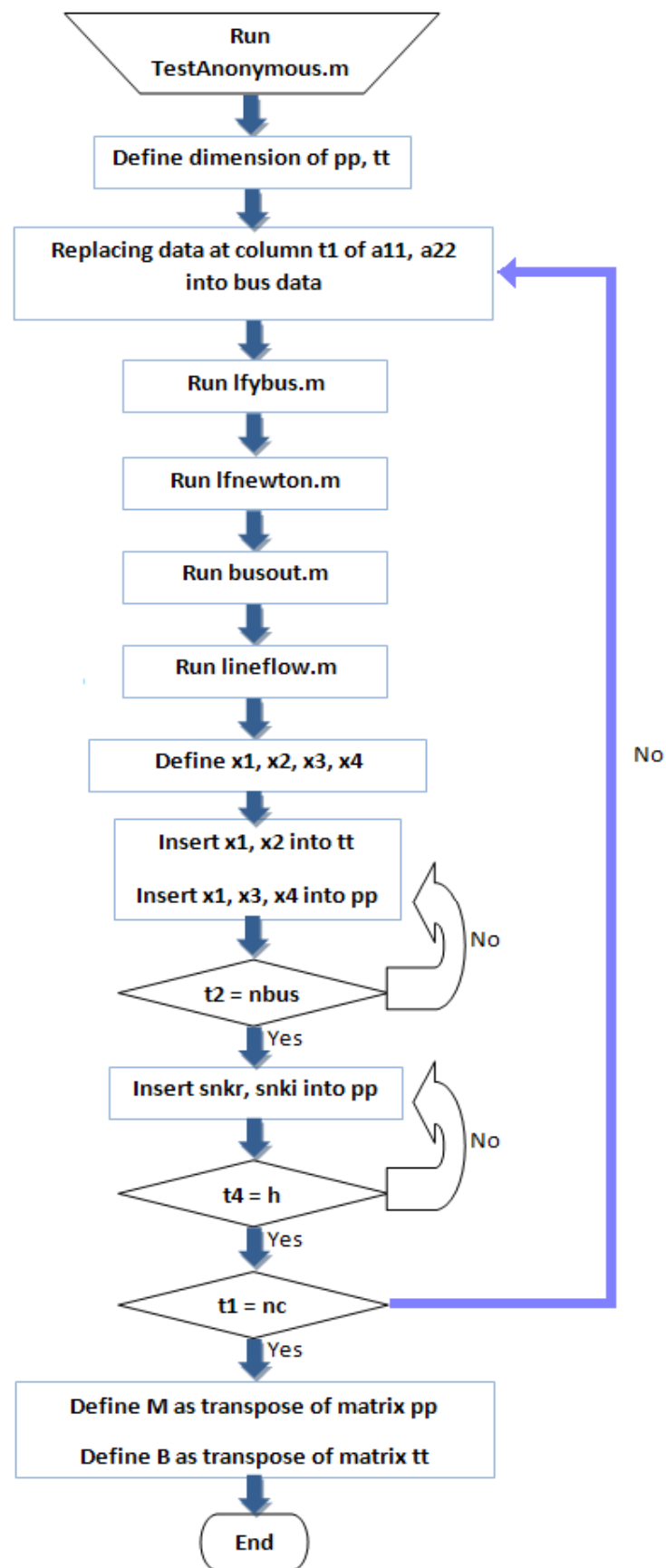


Figure 4.21 Flow of TestAnonymous3.m

4.6. State Estimation with Another Neural Network

After finish debugging and checking Back-Propagation network, I am looking into another neural network, the Radial Basis Neural Network. Basically, it does almost the same as Back-Propagation, but I did found that the Back-Propagation results in better output, which mean, the total loses estimated by Back-Propagation is lesser than Radial Basis Network. Figure 4.22a, Figure 4.22b, Figure 4.23a, Figure 4.23b, Figure 4.24a and Figure 4.24b shows the output of both neural network at the arrangement of complete set of data, incomplete set of data and anonymous set of data.

```

Total loss                                11.559   -1.718
The simulation of Data Collection phase is completed
Elapsed time is 1.319299 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.22a: BackPropagation Iteration On Complete Set of Data

```

Total loss                                11.160   -2.962
The simulation of Data Collection phase is completed
NEWRB, neurons = 0, MSE = 6.44144e-006
Elapsed time is 0.331569 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.22b: Radial Basis Network Iteration On Complete Set of Data

```

Total loss                                11.160   -2.962
The simulation of Data Collection phase is completed
Elapsed time is 0.092827 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.23a: BackPropagation Iteration On Incomplete Set of Data

```

Total loss                                11.160   -2.962
The simulation of Data Collection phase is completed
NEWRB, neurons = 0, MSE = 6.44144e-006
Elapsed time is 0.161038 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.23b: Radial Basis Network Iteration On Incomplete Set of Data

```

Total loss                                14.097  -36.880
Elapsed time is 0.082879 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.24a: BackPropagation Iteration On Anonymous Set of Data

```

Total loss                                15.221  -32.234
NEWRB, neurons = 0, MSE = 1.05594e-005
Elapsed time is 0.052721 seconds.
Training phase is completed. The Neural Network is formed
Testing has been carried out successfully

```

Figure 4.24b: Radial Basis Network Iteration On Anonymous Set of Data

From Figure 4.12a and Figure 4.12b, we can see that both gives output in range but slightly different value on each loses. This is cause by the difference of algorithm in both different neural networks. We can see that Figure 4.13a and Figure 4.13b are both having the same total loses. The accuracy of the estimation on both neural network is near 100%.

While the output of Figure 4.14a and Figure 4.14b are having both different values. The BackPropagation has slightly lower total loss compare to Radial Basis. Because the anonymous test has 3 different anonymous data given to the bus system, Radial Basis shows that it has some bias on the accuracy towards the output value. BackPropagation neural network is much better on higher iteration calculations.

Although the elapsed time of backpropagation is much longer compare to Radial Basis, it only consist of few milliseconds to 1 second time, which does not really obvious when both of the neural network is execute. As long as the output of the neural network does not bias too much, it would be a profit for them to estimate the power system state.

CHAPTER 5

CONCLUSION

5.1 Conclusion

In this project, the first and second category of testing phase results in high accuracy of about 99% whereas the third category outputs satisfactory accuracy of about 90%. This happens since neural network is built using a set of varied data and tested using another set of data with randomly manipulated range. Therefore, the data for testing can be just an assumption of real time power system values and the accuracy is hence considered high for this generalized neural network prototype.

This prototype, i.e. the developed program is in general form that allows the training phase to use other sets of data of any power system with any value range to create an individual neural network which is suitable and accurate to be used in state estimation for that particular power system of which data is used for training phase.

In real power system, data used for training phase is real time data and redundant. Therefore, it will create a specific neural network that performs effective

state estimation for that particular power system. In other words, the accuracy will be high.

This software program outputs instant results which are expected in accordance to the applied inputs which helps saving time as compared to the conventional method, i.e. hand calculation load flow analysis. It does not need complete input parameters data to run or perform and is able to give the desired or accurate outputs.

It should estimate bus voltage and load angle values at each node by minimizing difference between measured and calculated state variables that it aims to filter the errors at the same time considering the nonlinear characteristics of the practical equipment and actual measurements in distribution systems so that the best possible estimation of the system state is achieved.

The utilization of Load Flow Analysis, i.e. Newton Raphson method for this project, has successfully omitted the error that might exist in the raw measurement data of power system. As a result, the developed software program is able to estimate bus voltage and load angle values at each node with minimized difference between measured and calculated state variables whereby errors is filtered considering the nonlinear characteristics of the practical equipment and actual measurements in distribution systems. The best possible power system state estimation performance of this software programishenceachieved In conclusion, this project has successfully produced a software program that performs power system state estimation by utilizing the Artificial Neural Network technique through MATLAB software.

5.2 Recommendation

This power system state estimation software program is a prototype or general program that suits applications of power system with any size or number of buses. The data collection phase, i.e. data30bus.m can be edited with replacement of the real time redundant data obtained from the particular power system to be trained following the same method and program flow as shown in Chapter 5.

This software program gives highly accurate estimation results for a power system when its neural network is trained with the own data of that particular power system which best describes the characteristic and pattern of changes in reality.

In addition, the large capacity of real past time data of a power system will enable sufficient training phase where the neural network reaches equilibrium level and hence giving an expected result of accuracy that is near to the result of testing carried out in this project, i.e. 99%, under first category as explained in Chapter 4 section 4.4.1.

5.3 Costing and Commercialization

The development of this software program does not involve any hardware other than a computer or laptop that supports MATLAB software. Therefore there is cost incurred for the license of MATLAB.

This power system state estimation software program is a general program that can be edited to suit power system with any size or number of buses. Therefore, it can be applied at control center in power plant where the states of power system are processed and analyzed in computerized way.

The range of power system state values is not restricted for training phase of this software program, hence it can be commercialized locally and internationally in power system field.

REFERENCE

1. Shigenori Naka, Takamu Genji, Toshiki Yura, Yoshikazu Fukuyama, "Practical Distribution State Estimation Using Hybrid Particle Swarm Optimization" *Proc. of IEEE Power Engineering Society Winter Meeting*, Columbus, Ohio, USA, January 28 - February 1st, 2001.
2. Roberto Minguez, Antonio J. Conejo, Ali S. Hadi, "Non Gaussian State Estimation in Power System", *International Conference on Mathematical and Statistical Modeling in Honor of Enrique Castillo*. June 28-30, 2006.
3. Hadi Saadat.(2004) "*Power System Analysis*", 2nd Edition, McGraw Hill, 2004
4. Atabak Mashhadi Kashtiban, Majid Valizadeh, "Application of Neural Networks in Power System Security Assessment".
5. A. P. Sakis Meliopoulos, "State Estimation for Mega RTOs", *IEEE/PES Summer Meeting Chicago, IL – July 21-25, 2002*.
6. A. P. Sakis Meliopoulos, Bruce Fadarnesh, Shalom Zelingher, "Power System State Estimation: Modeling Error Effects and Impact on System Operation" .
7. A.H.M.A.Rahim, A.J.Al-Ramadhan, " Parameter Estimation of Power System Dynamic Equivalent Using ANN." *Proc Summer Computer Simulation Conference*, Chicago, July 1999.
8. A. Monticelli, "Electric Power System State Estimation", *Proceedings of the IEEE vol. 88, no. 2, February 2000*.
9. A. A. Hossam-Eldin, E. N. Abdallah, and M. S. El-Nozahy, "A Modified Genetic Based Technique for Solving the Power System State Estimation Problem", *World Academy of Science, Engineering and Technology* 55 2009.
10. Nabil H. Abbasy, Wael El-Hassawy "Power System State Estimation: ANN Application to Bad Data Detection and Identification".
11. L. Mili, Th. Van Cutsen, M. Ribbens-Pavella, "Bad Data Detection Methods in Power System State Estimation - A Comparative Study," *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS- 104, No.11, November 1985.

12. A. Panosyan, B. R. Oswald, "Modified Newton Raphson Load Flow Analysis For Integrated AC/DC Power System", *Institute of Electric Power Systems, University of Hannover, Germany*.
13. Robert Lukomski, Kazimierz Wilkosz, "Combining Theoretical Knowledge and Artificial Neural Networks For Power System Topology Verification.", *Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland*
14. D.M.Vinod Kumar, S.C. Srivastava, S.Shah, S.Mathur, "Topology processing and static state estimation using artificial neural networks", *IEE Proceedings online no. 19960050 Paper first received 16th January 1995 and in revised form 18th September 1995*.
15. A. K. Sinha J. K. Mondal, "Dynamic State Estimator Using ANN Based Bus Load Prediction", *IEEE Transactions on Power Systems, Vol. 14, No. 4, November 1999*.
16. http://en.wikipedia.org/wiki/Artificial_neural_network
17. Ehud D. Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks", *IEEE Transactions On Neural Networks Vol. 1 . No. 2. June 1990*

APPENDIX 1: databus30.m

```
% DATA COLLECTION phase
% 30-BUS SYSTEM
```

```
basemva=100; accuracy=0.001; accel=1.8; maxiter=100;
```

```
%      Bus Bus Voltage Angle      --Load--      ---Generator---Injected
%      No  Code Mag.   Degree    MW    Mvar    MW    Mvar Qmin  Qmax  Mvar
busdata=[ 1   1  1.06    0        0.0  0.0    0.0  0.0    0    0    0;
          2   2  1.043   0       21.70 12.7   40.0  0.0   -40   50    0;
          3   0  1.0     0        2.4  1.2    0.0  0.0    0    0    0;
          4   0  1.06    0        7.6  1.6    0.0  0.0    0    0    0;
          5   2  1.01    0       94.2 19.0    0.0  0.0   -40   40    0;
          6   0  1.0     0         0.0  0.0    0.0  0.0    0    0    0;
          7   0  1.0     0       22.8 10.9    0.0  0.0    0    0    0;
          8   2  1.01    0       30.0 30.0    0.0  0.0   -10   40    0;
          9   0  1.0     0         0.0  0.0    0.0  0.0    0    0    0;
         10   0  1.0     0         5.8  2.0    0.0  0.0    0    0   19;
         11   2  1.082   0         0.0  0.0    0.0  0.0    -6   24    0;
         12   0  1.0     0       11.2  7.5    0.0  0.0    0    0    0;
         13   2  1.071   0         0.0  0.0    0    0    -6   24    0;
         14   0  1.0     0         6.2  1.6    0    0    0    0    0;
         15   0  1.0     0         8.2  2.5    0    0    -6   24    0;
         16   0  1.0     0         3.5  1.8    0    0    0    0    0;
         17   0  1.0     0         9.0  5.8    0    0    0    0    0;
         18   0  1.0     0         3.2  0.9    0    0    0    0    0;
         19   0  1.0     0         9.5  3.4    0    0    0    0    0;
         20   0  1.0     0         2.2  0.7    0    0    0    0    0;
         21   0  1.0     0       17.5 11.2    0    0    0    0    0;
         22   0  1.0     0         0.0  0.0    0    0    0    0    0;
         23   0  1.0     0         3.2  1.6    0    0    0    0    0;
         24   0  1.0     0         8.7  6.7    0    0    0    0   4.3;
         25   0  1.0     0         0.0  0.0    0    0    0    0    0;
         26   0  1.0     0         3.5  2.3    0    0    0    0    0;
         27   0  1.0     0         0.0  0.0    0    0    0    0    0;
         28   0  1.0     0         0.0  0.0    0    0    0    0    0;
         29   0  1.0     0         2.4  0.9    0    0    0    0    0;
         30   0  1.0     0       10.6  1.9    0    0    0    0   0];
```

```
% Line Data
%      Bus Bus R      X      1/2B      for Line code or
%      nl  nr pu      pu      pu      tap setting value
linedata=[ 1   2  0.0192  0.0575  0.02640  1;
           1   3  0.0452  0.1852  0.02040  1;
           2   4  0.0570  0.1737  0.01840  1;
           3   4  0.0132  0.0379  0.00420  1;
           2   5  0.0472  0.1983  0.02090  1;
           2   6  0.0581  0.1763  0.01870  1;
           4   6  0.0119  0.0414  0.00450  1;
           5   7  0.0460  0.1160  0.01020  1;
           6   7  0.0267  0.0820  0.00850  1;
           6   8  0.0120  0.0420  0.00450  1;
           6   9  0.0     0.2080  0.0     0.978;
           6  10  0.0     0.5560  0.0     0.969;
```

9	11	0.0	0.2080	0.0	1;
9	10	0.0	0.1100	0.0	1;
4	12	0.0	0.2560	0.0	0.932;
12	13	0.0	0.1400	0.0	1;
12	14	0.1231	0.2559	0.0	1;
12	15	0.0662	0.1304	0.0	1;
12	16	0.0945	0.1987	0.0	1;
14	15	0.2210	0.1997	0.0	1;
16	17	0.0824	0.1923	0.0	1;
15	18	0.1073	0.2185	0.0	1;
18	19	0.0639	0.1292	0.0	1;
19	20	0.0340	0.0680	0.0	1;
10	20	0.0936	0.2090	0.0	1;
10	17	0.0324	0.0845	0.0	1;
10	21	0.0348	0.0749	0.0	1;
10	22	0.0727	0.1499	0.0	1;
21	22	0.0116	0.0236	0.0	1;
15	23	0.1000	0.2020	0.0	1;
22	24	0.1150	0.1790	0.0	1;
23	24	0.1320	0.2700	0.0	1;
24	25	0.1885	0.3292	0.0	1;
25	26	0.2544	0.3800	0.0	1;
25	27	0.1093	0.2087	0.0	1;
28	27	0.0000	0.3960	0.0	0.968;
27	20	0.2198	0.4153	0.0	1;
27	30	0.3202	0.6027	0.0	1;
29	30	0.2399	0.4533	0.0	1;
8	28	0.0636	0.2000	0.0214	1;
6	28	0.0169	0.0599	0.065	1];

APPENDIX 2: varydata10.m

```
% DATA COLLECTION phase
% to vary data up to 10 times (nc=10)

nbus = length(busdata(:,1));
nbr=length(linedata(:,1));
nc=10;
P1=busdata(:,5);
Q1=busdata(:,6);
for i1=1:nbus
    Range1(i1,1)=0.9*P1(i1);
    Range1(i1,2)=P1(i1)-0.25*P1(i1);
    Range(i1,1)=0.8*Q1(i1);
    Range(i1,2)=Q1(i1)-0.25*Q1(i1);
end
% load generation (active & Reactive)
for i=1:nbus
    a1=Range1(i,1)+(Range1(i,2)-Range1(i,1))*rand(nc,1);
    a2=Range(i,1)+(Range(i,2)-Range(i,1))*rand(nc,1);
    for j=1:nc
        a11(i,j)=a1(j);
        a22(i,j)=a2(j);
    end
end
end
```

APPENDIX 3: lfaNR.m

```
% DATA COLLECTION phase
% to run Newton Raphson load flow analysis

h=nbr;
h1=3*nbus;
h2=h1+h;
pp=zeros(nc,h1+2*h);
tt=zeros(nc,2*nbus);
for t1=1:nc
    busdata(:,5)=all(:,t1);
    busdata(:,6)=a22(:,t1);
    %[Ybus]=lfybus(linedata);
    %[Vm,delta,P,Q,S,VBc,a,nbr,nbus,nr,nl]=lfnewton(busdata,linedata);
    %[Snkr,Snki]=lineflow(busdata,linedata,Vm,delta,P,Q,S,V,Bc,nr,nl,basemv
a);
    lfybus;           % Forms the bus admittance matrix
    lfnewton;         %Power flow solution ny netween method
    busout;           %Print the power flow solution on the screen
    lineflow;         %Computes and displays the line flow and losses
    x1=Vm;
    x2=deltad;
    x3=P;
    x4=Q;
    % neural output
    for t2=1:nbus
        tt(t1,t2)=x1(t2);      % Vm
        tt(t1,nbus+t2)=x2(t2)*3.14/180; % deltad
    % neural input
        pp(t1,t2)=x1(t2);      % Vm
        pp(t1,nbus+t2)=x3(t2); % P
        pp(t1,2*nbus+t2)=x4(t2); % Q
    end
    for t4=1:h
        pp(t1,h1+t4)=snkr(t4); % Pij, Pji
        pp(t1,h2+t4)=snki(t4); % Qij, Qji
    end
end

clear t t1 t2 nbus x1 x2 x3 x4 Vm deltad P Q h h1 h2 tech nss ns nn ngs
snki
clear snkr linedata l ll lk lm m n nbr nc ng nl nr t4 y yload A Bc DC
DX J11
clear J22 J33 J44 L Pl Pd Pdt Pg Pgg Pgt Pk Ql Qd Qdt Qg Qgg Qgt Qk Ym
Z a al
clear all a2 a22 accel busprt deltad k kk kb i il iter j busdata
converge basemva
clear accuracy delta head maxerror maxiter Qmax Qmin Qsh Qsht R Range
Rangel S V X
```

APPENDIX 4: formNN.m(BackPropagation)

```
% TRAINING phase
% to train and form Neural Network
tic
warning off MATLAB:divideByZero
pp=pp';
tt=tt';

layer = 1;    % neural network layer
net = newff(pp,tt,layer);
toc
```


APPENDIX 5: formRBFNN.m(Radial Basis Network)

```
% TRAINING phase
% to train and form Neural Network
tic
warning off MATLAB:divideByZero
pp=pp';
tt=tt';

eg = 0.02; % sum-squared error goal
sc = 1;
net = newrb(pp,tt,eg,sc);
toc
```

APPENDIX 6: test.m

```
% TESTING phase
% to test the Neural Network

X= pp(:,1);      % X as input to the neural network
Y = sim(net,X);  % Y as output simulated by neural network
```

APPENDIX 7: TestEmpty.m

```
% TESTING phase
% to produce INCOMPLETE input data

m=length(pp(1,:));
s=10;
for r=1:16
    for d=1:m
        pp(s,d)=0*pp(s);
    end

    s=s+10; % data values are changed to zero at every 10 rows
    r=r+1;
end
```

APPENDIX 8: TestAnonymous1.m

```
% TESTING phase
% to change the values of 30-BUS SYSTEM
% 30-BUS SYSTEM

basemva=100; accuracy=0.001; accel=1.8; maxiter=100;

% Bus Data

%      Bus  Bus  Voltage Angle  --Load--      ---Generator---      Injected
%      No   Code   Mag.   Degree MW      Mvar  MW  Mvar  Qmin  Qmax  Mvar
Ori=[ 1    1    1.06    0      0.0    0.0    0.0  0.0    0    0    0;
      2    2    1.043   0     21.70  12.7   40.0  0.0   -40  50    0;
      3    0    1.0     0      2.4    1.2    0.0  0.0    0    0    0;
      4    0    1.06    0      7.6    1.6    0.0  0.0    0    0    0;
      5    2    1.01    0     94.2   19.0    0.0  0.0   -40  40    0;
      6    0    1.0     0      0.0    0.0    0.0  0.0    0    0    0;
      7    0    1.0     0     22.8   10.9    0.0  0.0    0    0    0;
      8    2    1.01    0     30.0   30.0    0.0  0.0   -10  40    0;
      9    0    1.0     0      0.0    0.0    0.0  0.0    0    0    0;
     10    0    1.0     0      5.8    2.0    0.0  0.0    0    0   19;
     11    2    1.082   0      0.0    0.0    0.0  0.0   -6   24    0;
     12    0    1.0     0     11.2   7.5    0.0  0.0    0    0    0;
     13    2    1.071   0      0.0    0.0    0    0   -6   24    0;
     14    0    1.0     0      6.2    1.6    0    0    0    0    0;
     15    0    1.0     0      8.2    2.5    0    0   -6   24    0;
     16    0    1.0     0      3.5    1.8    0    0    0    0    0;
     17    0    1.0     0      9.0    5.8    0    0    0    0    0;
     18    0    1.0     0      3.2    0.9    0    0    0    0    0;
     19    0    1.0     0      9.5    3.4    0    0    0    0    0;
     20    0    1.0     0      2.2    0.7    0    0    0    0    0;
     21    0    1.0     0     17.5  11.2    0    0    0    0    0;
     22    0    1.0     0      0.0    0.0    0    0    0    0    0;
     23    0    1.0     0      3.2    1.6    0    0    0    0    0;
     24    0    1.0     0      8.7    6.7    0    0    0    0   4.3;
     25    0    1.0     0      0.0    0.0    0    0    0    0    0;
     26    0    1.0     0      3.5    2.3    0    0    0    0    0;
     27    0    1.0     0      0.0    0.0    0    0    0    0    0;
     28    0    1.0     0      0.0    0.0    0    0    0    0    0;
     29    0    1.0     0      2.4    0.9    0    0    0    0    0;
     30    0    1.0     0     10.6    1.9    0    0    0    0    0];

for g=3:10
    busdataOri(:,g)=1.25*busdataOri(:,g);
end
busdata=busdataOri;

% Line Data

%      Bus  Bus  R      X      1/2B      for Line code or
%      n1   nr   pu      pu      pu      tap setting value
linedataOri= [1    2    0.0192  0.0575  0.02640  1;
              1    3    0.0452  0.1852  0.02040  1;
              2    4    0.0570  0.1737  0.01840  1;
```

3	4	0.0132	0.0379	0.00420	1;
2	5	0.0472	0.1983	0.02090	1;
2	6	0.0581	0.1763	0.01870	1;
4	6	0.0119	0.0414	0.00450	1;
5	7	0.0460	0.1160	0.01020	1;
6	7	0.0267	0.0820	0.00850	1;
6	8	0.0120	0.0420	0.00450	1;
6	9	0.0	0.2080	0.0	0.978;
6	10	0.0	0.5560	0.0	0.969;
9	11	0.0	0.2080	0.0	1;
9	10	0.0	0.1100	0.0	1;
4	12	0.0	0.2560	0.0	0.932;
12	13	0.0	0.1400	0.0	1;
12	14	0.1231	0.2559	0.0	1;
12	15	0.0662	0.1304	0.0	1;
12	16	0.0945	0.1987	0.0	1;
14	15	0.2210	0.1997	0.0	1;
16	17	0.0824	0.1923	0.0	1;
15	18	0.1073	0.2185	0.0	1;
18	19	0.0639	0.1292	0.0	1;
19	20	0.0340	0.0680	0.0	1;
10	20	0.0936	0.2090	0.0	1;
10	17	0.0324	0.0845	0.0	1;
10	21	0.0348	0.0749	0.0	1;
10	22	0.0727	0.1499	0.0	1;
21	22	0.0116	0.0236	0.0	1;
15	23	0.1000	0.2020	0.0	1;
22	24	0.1150	0.1790	0.0	1;
23	24	0.1320	0.2700	0.0	1;
24	25	0.1885	0.3292	0.0	1;
25	26	0.2544	0.3800	0.0	1;
25	27	0.1093	0.2087	0.0	1;
28	27	0.0000	0.3960	0.0	0.968;
27	20	0.2198	0.4153	0.0	1;
27	30	0.3202	0.6027	0.0	1;
29	30	0.2399	0.4533	0.0	1;
8	28	0.0636	0.2000	0.0214	1;
6	28	0.0169	0.0599	0.065	1];

```

for h=3:5
    linedataOri(:,h)=1.25*linedataOri(:,h);
end
linedata=linedataOri;

```

APPENDIX 9: TestAnonymous2.m

```
% TESTING phase
% to vary the changed values of 30-BUS SYSTEM up to 10 times (nc=10)

nbus = length(busdata(:,1));
nbr=length(linedata(:,1));
nc=10;
P1=busdata(:,5);
Q1=busdata(:,6);
for i1=1:nbus
    Range1(i1,1)=0.9*P1(i1);
    Range1(i1,2)=P1(i1)-0.25*P1(i1);
    Range(i1,1)=0.8*Q1(i1);
    Range(i1,2)=Q1(i1)-0.25*Q1(i1);
end
% load generation (active & Reactive)
for i=1:nbus
    a1=Range1(i,1)+(Range1(i,2)-Range1(i,1))*rand(nc,1);
    a2=Range(i,1)+(Range(i,2)-Range(i,1))*rand(nc,1);
    for j=1:nc
        a11(i,j)=a1(j);
        a22(i,j)=a2(j);
    end
end
end
```

APPENDIX 10: TestAnonymous3.m

```
% TESTING phase
% to run Newton Raphson load flow analysis to obtain a whole new set of
% inputs and target outputs for testing purpose

h=nbr;
h1=3*nbus;
h2=h1+h;
pp=zeros(nc,h1+2*h);
tt=zeros(nc,2*nbus);
for t1=1:nc
busdata(:,5)=a11(:,t1);
busdata(:,6)=a22(:,t1);

lfybus;           %Forms the bus admittance matrix (y bus)
lfnewton;          %Power flow solution Newton Raphson
busout;            %Print the power flow solution on the screen
lineflow;         %Computes and displays the line flow and losses

x1=Vm;
x2=deltad;
x3=P;
x4=Q;
% neural output
for t2=1:nbus
    tt(t1,t2)=x1(t2);      % Vm
    tt(t1,nbus+t2)=x2(t2)*3.14/180; % deltad
% neural input
    pp(t1,t2)=x1(t2);      % Vm
    pp(t1,nbus+t2)=x3(t2); % P
    pp(t1,2*nbus+t2)=x4(t2); % Q
end
for t4=1:h
    pp(t1,h1+t4)=snkr(t4); % Pij, Pji
    pp(t1,h2+t4)=snki(t4); % Qij, Qji
end
end

M=pp';
B=tt';

clear t t1 t2 nbus x1 x2 x3 x4 Vm deltad P Q h h1 h2 tech nss ns nn ngs
snki
clear snkr linedata l ll lk lm m n nbr nc ng nl nr t4 y yload A Bc DC
DX J11
clear J22 J33 J44 L P1 Pd Pdt Pg Pgg Pgt Pk Q1 Qd Qdt Qg Qgg Qgt Qk Ym
Z a al
clear a11 a2 a22 accel busprt deltad k kk kb i il iter j busdata
converge basemva
clear accuracy delta head maxerror maxiter Qmax Qmin Qsh Qsht R Range
Rangel S V X
```

