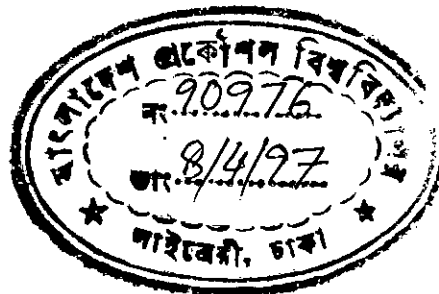


DESIGN AND IMPLEMENTATION OF A NEURAL NETWORK BASED REACTIVE POWER CONTROLLER

A THESIS
SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL AND ELECTRONIC ENGINEERING, BUET,
IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF SCIENCE IN ENGINEERING

BY
MD. ZIAUR RAHMAN



DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

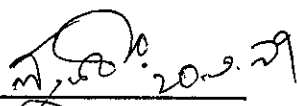
MARCH, 1997



**DEDICATED
TO MY PARENTS
THEY GAVE AWAY THEIR TODAYS FOR MY TOMORROWS**

The thesis titled, "Design and Implementation of a Neural Network Based Reactive Power Controller", submitted by Md. Ziaur Rahman, Roll No. 930613P, Registration No. 88157 of M. Sc. in Engineering has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of science in Electrical Engineering.

Board of Examiners:

1. 

Dr. S M Lutful Kabir

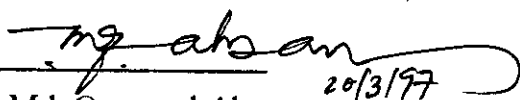
Professor

Department of Electrical & Electronic Engg.

BUET, Dhaka-1000

Chairman

(Supervisor)

2. 

Dr. Md. Quamrul Ahsan

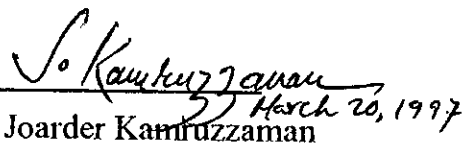
Professor and Head

Department of Electrical & Electronic Engg.

BUET, Dhaka-1000

Member

(Ex-Officio)

3. 

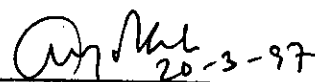
Dr. Joarder Kamruzzaman

Assistant Professor

Department of Electrical & Electronic Engg.

BUET, Dhaka-1000

Member

4. 

Dr. Abdur Rahim Mollah

Professor

Department of Electrical & Electronic Engg.

BIT, Khulna

Member

(External)

CONTENTS

Acknowledgments	i
Abstract	ii
List of figures	iv
List of Abbreviations	vi

CHAPTER 1: INTRODUCTION

1.1	Introduction	1
1.2	Fundamentals on power factor	
1.2.1	Power factor	4
1.2.2	Consumption of reactive energy	7
1.2.3	Necessity of better power factor	7
1.3	Conventional PFI plant	9
1.4	Thesis objective and layout	9

CHAPTER 2: DESIGN AND TRAINING OF THE NEURAL NETWORK

2.1	Introduction	11
2.2	Back propagation algorithm	
2.2.1	BPN operation	11
2.2.2	Mathematical analysis on BPN	14
2.2.3	BPN features	21
2.3	BPN for reactive power controller	
2.3.1	Input variables of the network	24
2.3.2	Output variable of the network	25

2.3.3	Hidden units of the network	27
2.4	Convergence of the network	
2.4.1	Hidden units	28
2.5	BPN with piece-wise linear activation	
2.5.1	Segmenting the sigmoid function	32
2.5.2	Training of the network	34
2.5.3	Convergence of the network	36
2.6	Sensitivity analysis of the network	40

CHAPTER 3: IMPLEMENTATION OF THE NEURAL NETWORK

3.1	Introduction	42
3.2	Control algorithm	42
3.3	Circuit implementation	
3.3.1	Input signal conditioning circuit	
3.3.1.1	Current sensing unit	44
3.3.1.2	Power factor sensing unit	46
3.3.2	Simulation of activation function	
3.3.2.1	Mathematical Analysis	48
3.3.2.2	Implementation	50
3.3.3	KVAR control circuit	
3.3.3.1	Function of the IC's	54
3.3.3.2	Elimination of low KVAR demand	58
3.3.4	Capacitor switching circuit	61

CHAPTER 4: RESULTS

4.1	Introduction	63
4.2	Performance testing on NN modules	
4.2.1	Testing on input signal conditioning unit	63
4.2.2	Testing on hidden nodes	65
4.2.3	Testing on neural network	70
4.2.4	Testing on KVAR control and capacitor switching circuit	70
4.3	Performance test on NN based RPC	
4.3.1	Experimental setup	73
4.3.2	Experimental data	76
4.4	Results	77
4.5	Cost analysis	80

CHAPTER 5: CONCLUSION

5.1	Conclusion	81
5.2	Further works	82

REFERENCES 84

APPENDICES

Appendix-1	A-1
Appendix-2	A-10
Appendix-3	A-15
Appendix-4	A-17
Appendix-5	A-19
Appendix-6	A-26
Appendix-7	A-29

ACKNOWLEDGMENT

The author reaffirms his indebtedness to Professor S M Lutful Kabir for his continuous supervision and inspiration to carry through this research. Cooperation from all the members of the electrical department is also appreciated.

ABSTRACT

Neural networks (NN) and their applications are creating immense interest among Electrical Engineers in every fields. Recent researches in this line are concentrated basically within two areas; implementation of NN and successful application of NN in various fields. This research work deals with both of these aspects of NN.

Trends on implementing NN, now a days, are dependent on VLSI technology utilizing the nonlinear operating regions of transistors within very small signal inputs. This research investigates an alternate way to implement analog NN with the simplest of electronic tools like resistors, diodes, buffers and analog adders where VLSI technology can be by-passed. Applications of NN in Power System Engineering has introduced a new dimension. Compared to other electrical fields, power system deals with larger signal sensing and interfacing; also unwanted signal spikes and harmonics in the power line may be vulnerable to small signal sensitive VLSI developed NN. Considering this, an analog NN based Reactive Power Controller (RPC) is designed and implemented in this project. In practice the microprocessor controlled RPC's are widely used. The main function of an RPC is to sense the amount of reactive power required by the system to improve its power factor. An analog NN based RPC has been locally designed and implemented having two input, six hidden and one output node capable of performing nonlinear function mapping; $X_1 \cdot \sin\theta$ in this particular case, where X_1 and θ are the two inputs. The sigmoid function is approximated to piece-wise linear (PWL) and off line training using Back propagation algorithm was used. A unique method has

been developed to implement the weights together with the PWL function with few resistance, diodes, buffers and an analog adder. A technic has been introduced to by-pass the need of amplifiers for constructing the weights. The performance of the developed RPC has been compared with an equivalent microprocessor based RPC. And the developed RPC has been found to be superior than the conventional one. Moreover, the developed RPC based on the implemented NN module approximately reduces the production cost by ten times in comparison to the microprocessor controlled RPC.

LIST OF FIGURES

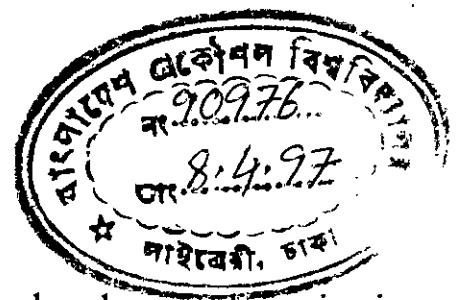
1.1	A graphical representation of power factor	5
2.1	The general backpropagation network architecture	12
2.2	The BPN network with suffix	15
2.3	Hypothetical error surface showing global and local minimum	23
2.4	A 3-D mapping of the inputs with the output	26
2.5	The correlation between output and target values of the test patterns	29
2.6	Response of node (1-4)	30
2.7	BPN module for the reactive power controller	31
2.8	Approximating sigmoid to piece-wise linear function	33
2.9	The BPN architecture of the neural network module with PWL	35
2.10	The correlation between output and target values of the test patterns	37
2.11	Response of node (1-6)	38
3.1	Line diagram of the neural network based RPC	43
3.2	The rectifier-filter circuit for the current	45
3.3	Rectifier-filter circuit for power factor sensing	47
3.4.a	j-th node of the hidden layer	49
3.4.b	PWL function of the hidden layer	49
3.5	A simple voltage divider circuit	51
3.6	Internal representation of IC LM3914	55
3.7	The pin diagram of 74LS148	57
3.8	The logic block prohibiting the KVAR control circuit to operate at very low KVAR demand	59
3.9	The block diagram of KVAR control circuit	60
3.10	Capacitor switching circuit	62

4.1	Experimental setup for testing input signal conditioning unit	64
4.2.a	Response of current sensing unit	67
4.2.b	Response of power factor sensing unit	67
4.3	Test setup for hidden nodes	68
4.4a	Response of node 4 with input 2 grounded	69
4.4.b	Response of node 4 with input 1 grounded	69
4.5	Comparison between expected and actual output	72
4.6	Experimental setup for performance test of NN based RPC	75
4.7	The overall response of NN based RPC	79
A.1.1	Main features of RM 9606 type microprocessor based RPC controller	A-2
A.1.2	Circuit diagram of RM 9606 connected with the line bus	A-9

LIST OF ABBREVIATIONS

- BPN – Back Propagation Network
C.T. = Current Transformer
GDR = Generalized Delta Rule
KVAR = Kilo Volt Ampere Reactive
LED = Light Emitting Diode
NN = Neural Network
PFC = Power Factor Controller
PFI = Power Factor Improvement
PWL = Piece-Wise Linear
RPC = Reactive Power Controller
VLSI = Very Large Scale Integration
 μ P = MicroProcessor
 μ PC = MicroProcessor controller

CHAPTER ONE
INTRODUCTION



1.1 INTRODUCTION

Applications of artificial neural network have ushered a new dimension in the field of engineering, especially in industrial control systems. The neural networks try to mimic the nerve system in a mammalian brain into a mathematical model. The brain is a large-scale system connecting many neural cells called neuron. It has many excellent characteristics: parallel processing of information, learning function, self organizing capabilities and so forth [1],[2]. The brain can also provide an associative memory [2] and is good for information processing such as pattern recognition [3]. In artificial neural network, a model of the brain, connects many linear or nonlinear neuron models and process information in a parallel distributed manner [4]. In conventional single processor Von Neumann computers, the speed of computation is limited by the propagation delay of the transistors. Because of their massively parallel nature, neural network can perform computation at much higher speed [1]. In addition, the neural network has many interesting and attractive features. Neural networks have learning and self organization capabilities. Therefore, neural network can adapt to changes in data, learning the characteristics of the input signal. That is, neural network can learn a mapping between an input and output space and synthesize an associative memory that retrieves the appropriate output when presented with the input and generalizes when presented with new inputs [5]. Moreover, because of their nonlinear nature, neural network can perform functional approximation and signal filtering operations that are beyond optimal linear techniques [3]. Recently, many researchers have developed neural networks as new tools in many fields such as pattern recognition, information processing, design, planning, diagnosis, and control. This thesis work develops a hybrid system, a neural network based reactive power controller, as an example of a key technology in the future.

Most of the works done, so far, in industrial control systems are simulated by software programs or fabricated using VLSI technology [6]. The software simulation needs a microprocessor and usually takes a long period of time to execute the huge number of computations involved in the operation of the network. Several researches have adopted hardware implementations to realize such network [7]-[11]. During the past few years, various researchers have begun addressing analog and digital hardware implementations [12]-[21] of certain artificial neural network architecture encompassing a wide variety of applications. Between these two types, analog implementations of artificial neural networks have a number of unique advantages and problems when compared to digital realizations. The primary motivation for implementing a neural network algorithm with analog circuitry is its stand alone capability and capacity to operate on a real time fashion. Countering the above analog advantages is a more extensive list of difficulties and shortcomings. Typically, analog circuits are more complicated to design and more limited in application than digital circuitry [22]. On the other hand, the options that exist for speedy network solutions are digital serial processors known as “neural network coprocessors” [23]-[25]. Using specialized chips that are optimized for matrix multiplication and scaling primitives that underlie most neural models, these digital systems are generally circuit boards that plug onto a host computer bus. Although such systems are capable of simulating networks one or two orders of magnitude faster than the host computer, they remain serial simulators, with network simulation times that grow linearly with interconnection complexity. Hence, analog implementation of neural network seems to be the ultimate choice — at least for the time being.

In the proposed research, an analog neural network based reactive power controller will be designed and implemented. Reactive Power Controllers (RPC) are one of the most essential but costly electrical equipment for any industries or production factories. RPC is required in distribution system to improve power factor at a particular bus. It serves as a source of reactive power which is controlled by an automatic power factor controller (PFC) relay. The PFC's are microprocessor [26],[27] based control relay which automatically switches the capacitors in and out of the circuit. Depending on the switching stages of the RPC, its cost varies between one lac Taka for three step switching stages to five lac Taka for twelve step switching stages. This research project implements an RPC based on a feed forward neural network model.

Modeling biological systems presents many challenges to the analog circuit designer. Neural computation is often an emergent property of the system, derived from the way the component elements are organized, and may not be evident in any single element. It is often difficult to separate a neural structure into functional units [28]. Major areas are richly interconnected and computation is intertwined, as a single neural structure subserves a multitude of functions simultaneously [29]. As a result, computational strategies for building collective systems require the development of new architectures and a new design methodology. Mead [30] presented such methodology for implementing biological inspired architectures. This thesis work investigates a new design methodology innovated during the development of the analog neural network for the proposed RPC with the use of simple electronic tools. The neural architecture introduced in this research can be implemented with or without the technical support of VLSI.

1.2 FUNDAMENTALS ON POWER FACTOR

This section will explain some basic terminology on power factor, reactive power and other related topics under three articles.

1.2.1 Power Factor

Most frequently, an industrial installation is fed from a high-voltage system and comprises :

- a transformer station,
- “resistive” loads, such as ovens, radiators, filament lamps, etc.,
- “inductive” loads, such as transformers, motors, etc.

Let it be assumed that the system is single phase. V will be the voltage in volts at the secondary terminals of the supply transformer that is should to be at full load, and I will be the total absorbed current in amperes.

It should be noted that the **Actual or Active Consumed Power** P_w , in watts, is lower than the product: volts \times amperes representing the **Apparent Power** of the installation P_a in a ratio that is equal to the **Power Factor**.

$$\frac{P_w(\text{watts})}{P_a(\text{VA})} = \cos \varphi < 1 \quad \dots \quad \dots \quad \dots \quad (1.1)$$

The power factor is graphically represented by the cosines of the angle obtained by the difference in phase between current and voltage; the angle represents the lag between I and V . Figure 1.1 gives a graphical example of power factor.

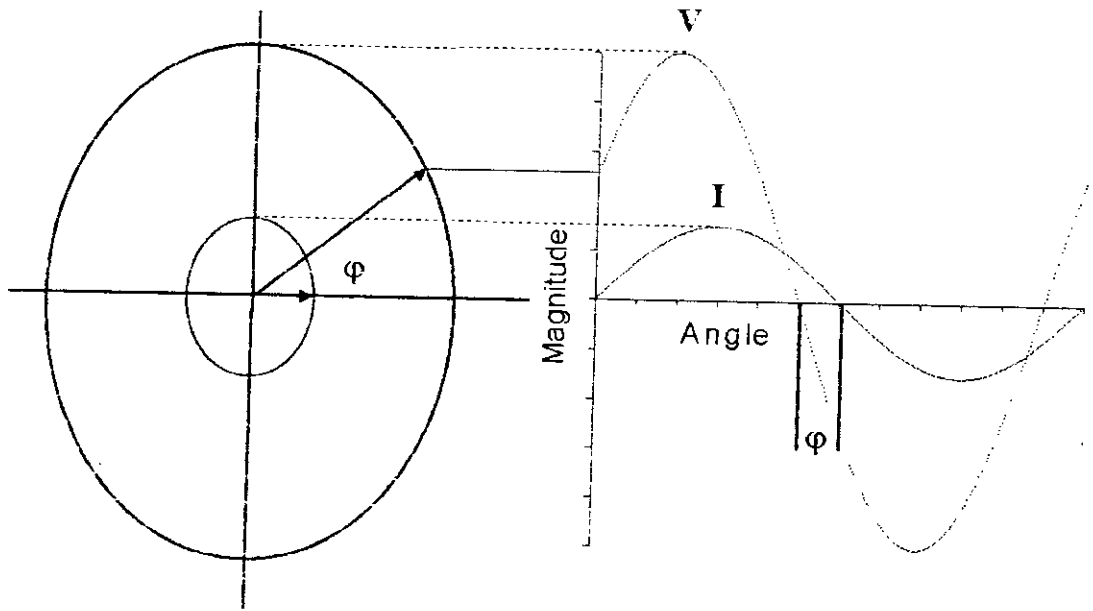


Figure 1.1 A graphical representation of Power Factor.

The $\cos\phi$ depends only on the characteristics of the load and on their operating conditions (type of motor, speed, load); it is independent of the efficiency of the loads. The loads of the “inductive” type absorb :

- an active power $P_w = V.I. \cos\phi$ (watts);
- a reactive power $P_R = V.I.\sin\phi$ (VA react.).

The **Apparent Power** P_a equal to the product $V \times I$, represents the geometrical sum of the active and reactive powers. The **Active Power** P_w is integrally converted into work and heat (losses), being measured with a suitable watt-meter. The active energy W_a which corresponds to it is recorded by an active energy meter; it is charged to the consumer by the power supply company. The **Reactive Power** P_R is the power necessary for the magnetic excitation of the said loads. The reactive energy W_r , which corresponds to it can be recorded by a reactive energy meter. The loads of the “resistive” type absorb active power only; the power factor is equal to unity and $P_a = P_w$.

The **Average Power Factor** of an installation, over a long period of time of operation of the workshops is obtained by means of the angle tangent :

$$\text{average tg } \phi = \frac{W_r (\text{VA hour - reactive})}{W_a (\text{watts - hour})} \quad \dots \quad \dots \quad \dots \quad (1.2)$$

When the tangent is known, trigonometric tables give the value of the corresponding $\cos\phi$.

1.2.2 Consumption of Reactive Energy

1. Motors : These are among the loads which consume the greatest amount of reactive energy.
2. Transformers : By design, all transformers consume reactive energy used for the magnetization of their cores.
3. Transmission lines : Transmission lines, especially overhead lines, have a comparatively high reactance ($X_L = 0.3$ to $0.5 \Omega\text{-km/phase}$). The reactive power which they absorb therefore depends upon the load which they carry according to the following formula :

$$P_R = I^2.X_L \text{ (I = line current)} \quad \dots \quad \dots \quad \dots \quad (1.3)$$

Since transmission lines have also capacitive characteristic X_C delivering a reactive power equal to $V^2.X_C$ (where V is the service voltage), their consumption of reactive voltage depends only on the load, namely when $I^2.X_L$ becomes higher than $V^2.X_C$.

1.2.3 Necessity of Better Power Factor

The advantage of good power factor are multifold and all result in a substantial economy in the operation of electrical installations.

1. cutting down penalties for excessive consumption of reactive energy
2. reducing line losses : Even when the resistance of conductors is largely calculated, it always causes watt losses which are added to the active consumption of the installation. These losses are proportional to the square of the current carried which for the same active power, decreases as the power factor is increasing.

3. increasing line power carrying-capacity with equal losses : If it is considered that an installation which, further to an extension of its activity for instance, has to carry a higher active power, only the improvement of its power factor will allow such an increase without augmenting line losses and, most often, if the initial power factor is small, without modifying the lines p-ower.
4. increasing power available at supply transformers : When the power factor increases, the apparent power for the same active power decreases.
5. reducing voltage drop : In overloaded low-voltage distribution lines supplying workshops with a small power factor, voltage drops often occur; these are likely to impair the satisfactory operation of motors, even if the voltage at the transformer output is correct. Switching on a capacitor bank at the end of the lines causes a voltage rise ΔV defined by the formula :

$$\Delta V\% = \frac{X_L \times Q}{10 V^2} \quad \dots \quad \dots \quad \dots \quad \dots \quad (1.4)$$

Where :

- X_L = line reactance in ohms,
- Q = output of the capacitor bank in KVAR,
- V = rated voltage of capacitor in kV.

Switching on a capacitor bank at the terminals of a transformer causes a voltage rise :

$$\Delta V\% = (Q/P).V_{CC} \quad \dots \quad \dots \quad \dots \quad \dots \quad (1.5)$$

Where :

- Q = output of the capacitor bank in KVAR,
- P = power of transformer in KVA,
- V_{CC} = transformer impedance voltage (in percentage).

Such a voltage rise, often necessary at full load, could be disastrous at no load. It is therefore necessary to switch off capacitor banks during light-load conditions.

1.3 CONVENTIONAL PFI PLANT

The previous section has explained the advantage of high power factor and the importance of improving power factor in power system. Power Factor Improvement (PFI) plants are used for this purpose. This section will describe the operation and special features of conventionally used PFI plant. Now-a-days, the most widely used PFI plants are microprocessor controlled. The main features of the microprocessor based PFI controller is described in Appendix-1.

1.4 THESIS OBJECTIVE AND LAYOUT

This research work is a combination of theoretical study and practical implementation. A neural network based reactive power controller will be designed and developed, and finally, the performance of the proposed RPC will be compared with the conventional microprocessor controlled RPC in this thesis. The objective of this thesis is classified in two broad view points presented below.

1. There are many methodologies presented by researchers on the implementation of neural networks. One objective of this research work is to introduce a new technology on implementing analog neural networks using simple electronic tools.

It is hoped that the invented methodology will make neural network implementation easier and simpler.

2. Conventional microprocessor controlled relays are pretty costly. This research work will show that neural network based controller relays will perform almost as same as the conventional relays, but the cost will be reduced drastically. The second objective of this project is to propose a new way of controlling power factor relays utilizing the emerging techniques of neural systems.

The thesis layout has been confined within three main chapters. Chapter two describes neural network theories and procedures of design a neural network module for the proposed RPC. Chapter three gives an extensive description of the implementation techniques used in the development of the RPC. The final chapter shows the results of the thesis work and presents the performance of the implemented analog neural network based RPC.

CHAPTER TWO

DESIGN AND TRAINING OF THE NEURAL NETWORK

2.1 INTRODUCTION

This chapter explains the back propagation algorithm and utilizes the algorithm to develop an artificial neural network for the reactive power controller. It also assists to form a theoretical back ground on artificial intelligence, learning algorithm and sensitivity of different parameters on convergence of the network. A neural network module for implementing the proposed RPC is presented in this chapter.

2.2 BACK PROPAGATION ALGORITHM

Several methodologies of the artificial neural network have been developed starting from the perceptron idea of Rosenblatt [31]. Among them, the **Backpropagation network** is one of the most effective versatile tool that is readily applied to a number of diverse problem in artificial neural network. To a large extent, its versatility is due to the general nature of the network learning process. This algorithm has established its popularity over other neural network algorithms; specially in the field of power system analysis. In this present project of developing a reactive power controller based on artificial neural system the BPN algorithm, therefore, becomes a natural choice. In this section the theory of back propagation technic is presented.

2.2.1 BPN Operation

A summary description of BPN operation is described to illustrate how the BPN can be used to solve complex problems. A three layer back propagation architecture is shown in Figure 2.1. The layers are fully interconnected. When signal patterns are applied to the input layer of the network it propagates upwards

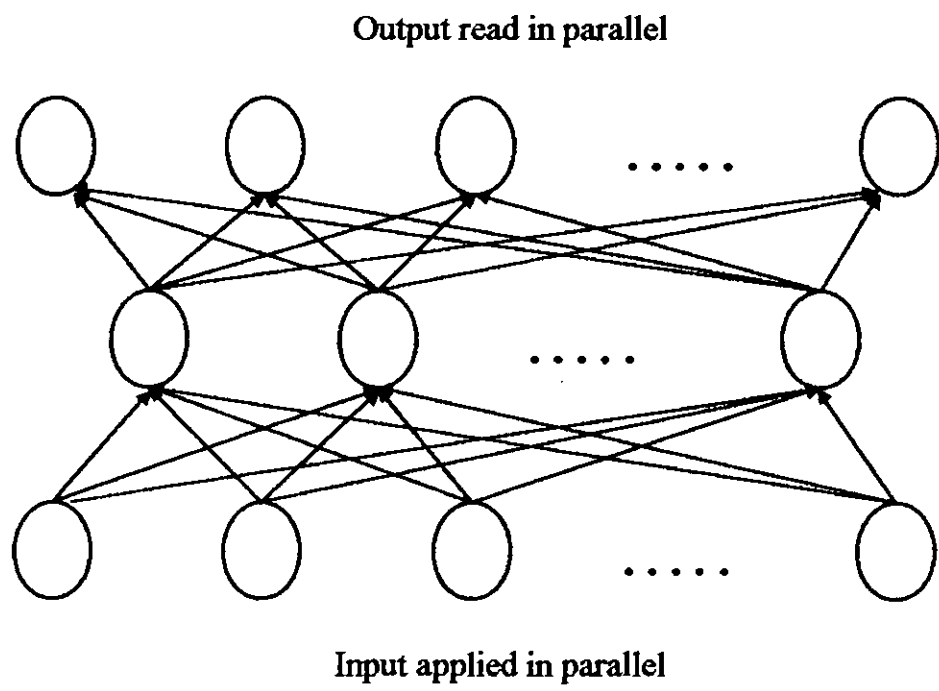


Figure 2.1 The general back propagation network architecture.

towards the output layer through the interconnections of the middle layer, known as hidden layer. It is required that the propagated signal will finally produce a desirable output pattern.

The network learns a predefined set of input-output example pairs by using a two-phase **propagate-adapt** cycle. After an input pattern has been applied as a stimulus to the first layer of the network units, it is propagated through each upper layer until an output is generated. This output pattern is then compared to the desired output, and an error signal is computed for each output unit. The error signals are then transmitted backward from the output layer to each node in the intermediate layer that contributes directly to the output. However, each unit in the intermediate layer receives only a portion of the total error signal, based roughly on the relative contribution the unit made to the original output. This process repeats, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the total error. Based on the error signal received, connection weights are then updated by each unit to cause the network to converge toward a set that allows all the training patterns to be encoded.

The significance of this process is that, as the network trains, the nodes in the intermediate layers organize themselves such that different nodes learn to recognize different features of the total input space. After training, when presented with an arbitrary input pattern, the units of the hidden layers of the network will respond with an active output which is very close to the target value.

As the signal propagates through the different layers in the network, the activity pattern present at each upper layer can be thought of as a pattern with

features that can be recognized by units in the subsequent layer. The output pattern generated can be thought of as a feature map that provides an indication of the presence and absence of many different feature combinations at the input. The total effect of this behavior is that the BPN provides an effective means of allowing the total system to examine data patterns that may be untrained and to recognize the corresponding output.

Several researchers have shown that during training, BPNs tend to develop internal relationships between nodes so as to organize the training data into classes of patterns. This tendency can be extrapolated to the hypothesis that all the hidden units in the BPN are somehow associated with specific features of the input pattern as a result of training. Exactly what association is may or may not be evident to the human observer. What is important is that the network has found an internal representation that enables it to generate the desired outputs when given the training inputs. This same internal representation can be applied to inputs that were not used during training. The BPN will classify these previously unseen inputs according to the features they share with the training examples.

2.2.2 Mathematical Analysis on BPN

In this article, a rigorous mathematical description of BPN will be represented with the detail derivation of **generalized delta rule (GDR)**, which is the learning algorithm for the network. Figure 2.2 is the repetition of Figure 2.1 where suffix are included to serve as the reference of the discussion. The BPN is a layered, feedforward network that is fully interconnected by layers. There are no feedback connections and no connections that bypass one layer to go directly to a later layer.

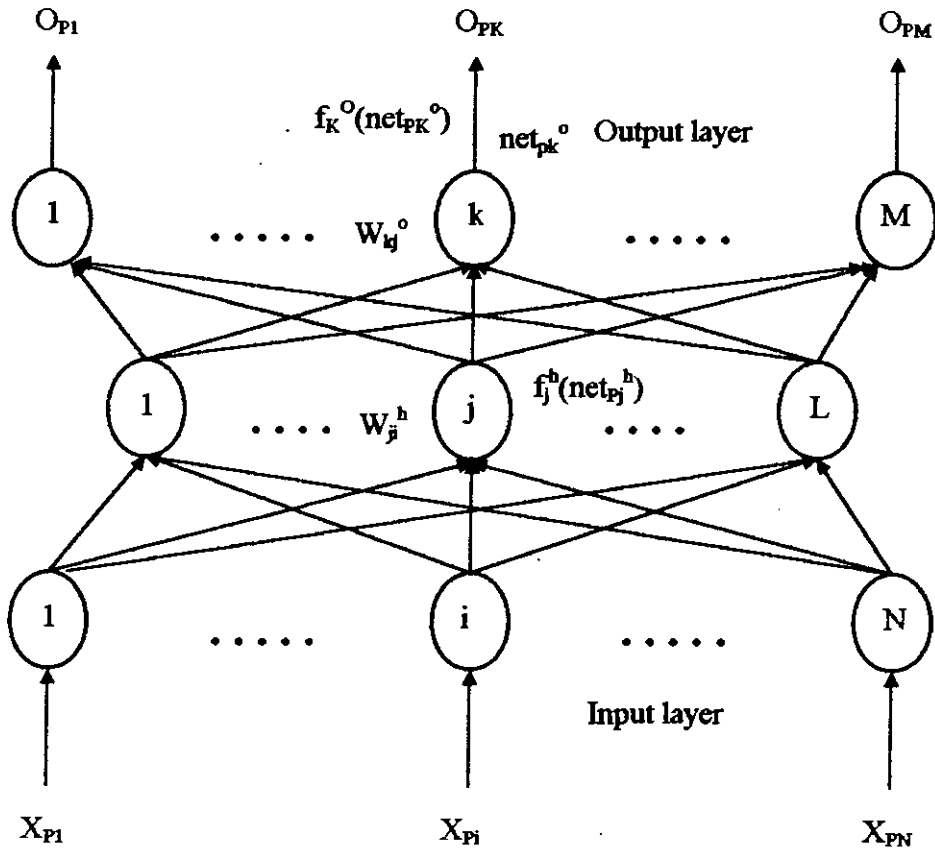


Figure 2.2 The BPN network with suffix

The network will be trained to learn a functional mapping $y = \varphi(\mathbf{x}) : \mathbf{x} \in \mathbf{R}^N, \mathbf{y} \in \mathbf{R}^M$. A set of P vector pairs of the function are $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_P, \mathbf{y}_P)$. Considering the mapping to be nonlinear and multidimensional, the iterative version of the simple least square method, called **steepest descent technique**, will be employed.

An input vector, $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pN})$, is applied to the input layer of the network. The input units distribute the values to the hidden layer units. The net input to the j th hidden unit is

$$\text{net}_{pj}^h = \sum_{i=1}^N w_{ji}^h \cdot x_{pi} \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.1)$$

where w_{ji}^h is the weight on the connection from the i th input unit. The "h" superscript refers to quantities on the hidden layer. For a defined activation function of this node, the output of this node will be

$$i_{pj} = f_j^h(\text{net}_{pj}^h) \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.2)$$

The equations for the output nodes are

$$\text{net}_{pk}^o = \sum_{j=1}^L w_{kj}^o \cdot i_{pj} \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.3)$$

$$O_{pk} = f_k^o(\text{net}_{pk}^o) \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.4)$$

where “o” superscript refers to quantities on the output layer. The initial set of weights represents a first guess as to the proper weights for the problem.

The error value at a single output unit “k” is defined as $\delta_{Pk} = (y_{Pk} - O_{Pk})$, where the subscript “p” refers to the pth training vector, and y_{Pk} is the desired output value. The error that is minimized by the GDR is the sum of the squares of the errors of all the output units.

$$E_p = 0.5 \cdot \sum_{k=1}^M \delta_{Pk}^2 \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.5)$$

To determine the direction in which to change the weights, negative of the gradient of E_p , ∇E_p , with respect to the weights, w_{kj} is calculated. Then, the weights can be adjusted in such way so that the total error is reduced.

Considering only for the kth output unit, the component of ∇E_p is calculated separately.

$$E_p = 0.5 \cdot \sum_k (y_{Pk} - O_{Pk})^2 \quad \dots \quad \dots \quad \dots \quad \dots \quad (2.6)$$

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{Pk} - O_{Pk}) \cdot \frac{\partial f_k^o}{\partial(\text{net}_{Pk}^o)} \cdot \frac{\partial(\text{net}_{Pk}^o)}{\partial w_{kj}^o} \quad \dots \quad \dots \quad (2.7)$$

The last factor in Eq. (2.7) is

$$\frac{\partial(\text{net}_{pk}^o)}{\partial w_{kj}^o} = \frac{\partial}{\partial w_{kj}^o} \sum_{j=1}^L w_{kj}^o \cdot i_{pj} = i_{pj} \dots \dots \dots \quad (2.8)$$

Combining Eqs. (2.7) and (2.8), the negative gradient is

$$-\frac{\partial E_p}{\partial w_{kj}^o} = (y_{pk} - o_{pk}) \cdot f_k^{o'}(\text{net}_{pk}^o) \cdot i_{pj} \dots \dots \dots \quad (2.9)$$

Thus the weights of the output layer are updated according to

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \Delta_p \cdot w_{kj}^o(t) \dots \dots \dots \quad (2.10)$$

$$\Delta_p \cdot w_{kj}^o = \eta \cdot (y_{pk} - o_{pk}) \cdot f_k^{o'}(\text{net}_{pk}^o) \cdot i_{pj} \dots \dots \dots \quad (2.11)$$

The factor η is called the learning rate parameter. It is usually less than 1.

The weight update Eq. (2.10) can be reformed by defining a quantity

$$\begin{aligned} \delta_{pk}^o &= (y_{pk} - o_{pk}) \cdot f_k^{o'}(\text{net}_{pk}^o) \\ &= \delta_{pk} \cdot f_k^{o'}(\text{net}_{pk}^o) \dots \dots \dots \end{aligned} \quad (2.12)$$

The weight update equation thus becomes

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta \cdot \delta_{pk}^o \cdot i_{pj} \dots \dots \dots \quad (2.13)$$

So far only the weights of the output layer have been modified. The weights of the hidden layers should need modification as error signal propagates downwards. Going back to Eq. (2.6) :

$$\begin{aligned}
 E_p &= 0.5 \cdot \sum_k (y_{pk} - O_{pk})^2 \\
 &= 0.5 \cdot \sum_k (y_{pk} - f_k^o(\text{net}_{pk}^o))^2 \\
 &= 0.5 \cdot \sum_k (y_{pk} - f_k^o(\sum_j w_{kj}^o \cdot i_{pj}))^2
 \end{aligned}$$

Again, i_{pj} depends on the weights on the hidden layer through Eqs. (2.1) and (2.2). Exploiting this fact to calculate the gradient of E_p with respect to the hidden layer weights :

$$\begin{aligned}
 \frac{\partial E_p}{\partial w_{ji}^h} &= 0.5 \cdot \sum_k \frac{\partial}{\partial w_{ji}^h} (y_{pk} - O_{pk})^2 \\
 &= - \sum_k (y_{pk} - O_{pk}) \cdot \frac{\partial O_{pk}}{\partial(\text{net}_{pk}^o)} \cdot \frac{\partial(\text{net}_{pk}^o)}{\partial i_{pj}} \cdot \frac{\partial i_{pj}}{\partial(\text{net}_{pj}^h)} \cdot \frac{\partial(\text{net}_{pj}^h)}{\partial w_{ji}^h} \quad (2.14)
 \end{aligned}$$

$$= \sum_k (y_{pk} - O_{pk}) \cdot f_k^{o'}(\text{net}_{pk}^o) \cdot w_{kj}^o \cdot f_j^{h'}(\text{net}_{pj}^h) \cdot x_{pi} \quad \dots \quad (2.15)$$

With the help of Eq. (2.15) the weights of the hidden layer are updated.

$$\Delta_p \cdot w_{ji}^h = \eta \cdot f_j^{h'}(\text{net}_{pj}^h) \cdot x_{pi} \cdot \sum_k (y_{pk} - O_{pk}) \cdot f_k^{o'}(\text{net}_{pk}^o) \cdot w_{kj}^o \quad (2.16)$$

where η is once again the learning rate.

The weight updating Eq. (2.16) for the hidden layer can be rearranged with the help of δ_{pk}^o from Eq. (2.12).

$$\Delta_p \cdot w_{ji}^h = \eta \cdot f_j^{h'}(\text{net}_{pj}^h) \cdot x_{pi} \cdot \sum_k \delta_{pk}^o \cdot w_{kj}^o \quad \dots \quad \dots \quad \dots \quad (2.17)$$

A hidden layer error term similar to δ_{pk}^o can be defined.

$$\delta_{pj}^h = f_j^{h'}(\text{net}_{pj}^h) \cdot \sum_k \delta_{pk}^o \cdot w_{kj}^o \quad \dots \quad \dots \quad \dots \quad (2.18)$$

Finally, weight update equation for the hidden layer is reduced to the following form :

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \cdot \delta_{pj}^h \cdot x_{pi} \quad \dots \quad \dots \quad \dots \quad (2.19)$$

Before leaving this section there is one point yet to be mentioned. So far the activation function of the nodes were mathematically defined by Eqs. (2.2) and (2.4). These functions require to be differentiable. The simplest of all that can be thought of is surely the straight line function. But for a nonlinear mapping the activation function should have to be nonlinear. The most utilized function prescribed by Hopfield is the **sigmoid function**. It is said that this function closely resembles the biological neuron activation. The mathematical equation of sigmoid function is given below.

$$f_k^o(\text{net}_{jk}^o) = 0.5.(1 + \tanh(\lambda.\text{net}_{jk}^o)) \quad \dots \quad \dots \quad \dots \quad (2.20)$$

The derivative of the sigmoid function can be arranged in the following way

$$f_k^{o'}(\text{net}_{jk}^o) = f_k^o(1-f_k^o) = O_{Pk}(1-O_{Pk}) \quad \dots \quad \dots \quad \dots \quad (2.21)$$

2.2.3 BPN Features

In the previous article relevant mathematical equations required for BPN programming were presented. A computer program “LEARN.FOR” (Appendix-2) has been developed based on those equations. This program will calculate the weights of different layers for network convergence within acceptable error limit. Apart from the mathematical analysis of BPN, certain practical features of its algorithm require special attention which are discussed in this article.

- **Training Data** : There are no hard and fast rule of selecting the training patterns for BPN learning. Experience is often the best teacher. Yet it should be kept in mind that BPN is very good in generalization but equally bad in extrapolation. If a BPN is inadequately and insufficiently trained on a particular class of input vectors subsequent identification of members of that class may be unreliable. So training vectors should be selected in such way that they will cover the total range of variation the network might experience in practical field. For this present project 55 input patterns were generated which ultimately converged satisfactorily for about 896 patterns.
- **Network sizing** : The size of the input and output layer are usually dictated by the nature of the application. Determining the number of units to use in the hidden

layer is not so straight forward. The main idea is to use as few hidden layer units as possible. Because this makes the learning process fast and implementation of the network easy. But in case of too much complex mapping, size of the hidden layer may be large for network convergence. Usually networks are initially designed big in size. After learning, the network is pruned by examining the weight values.

- **Initial weights** : The initial weights are generally selected at random. Values within ± 0.5 are chosen frequently. But there is always a possibility that the network may stuck to a **local minimum** in weight space. Figure 2.3 illustrates this phenomena. Once a network settles on a minimum, whether local or global, learning ceases. If a local minimum is reached, the error of the network may still be unacceptably high. In such a case initial weights need to be changed. Sometime increase in number of hidden layer or learning rate can fix the problem. But if the error keeps within acceptable limit, whether the network has stuck into local or global minimum does not matter.
- **Learning rate parameter** : Selection of the value of the learning rate parameter, η , has a significant effect on network performance. Usually, η must be a smaller number, on the order of 0.05 to 0.25, to ensure that the network will settle to a solution. A small value of η makes the iteration process slow. Too large value of η may make the network bounce around too far from the actual minimum value. It is often suggested that η initially kept high and as the network proceeds close to convergence, the value of η be reduced. Learning rate can also be modified just like the weights are updated within the computer program. For the present project it has been found that a constant η of 0.2 is good enough for convergence.

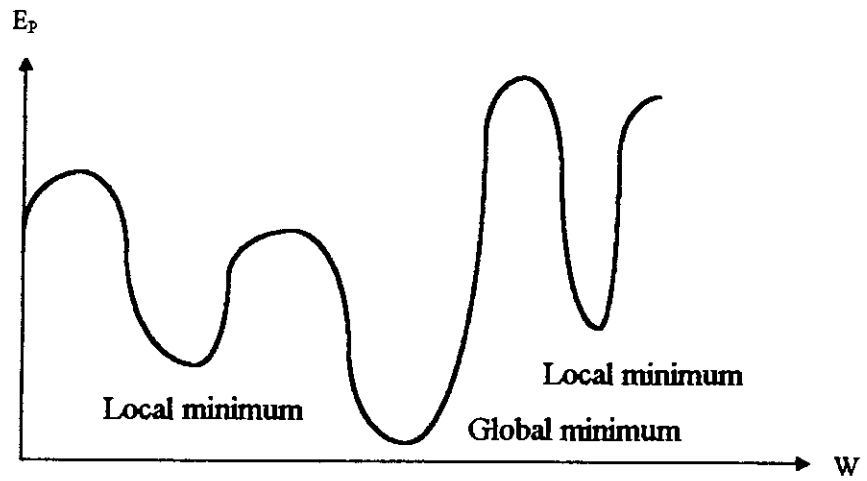


Figure 2.3 Hypothetical error surface showing global and local minimum.



2.3 BPN FOR REACTIVE POWER CONTROLLER

The preceding section has elaborately explained different aspects of BPN algorithm. In this section total attention will be focused on the project of developing a “Reactive Power Controller” based on BPN.

The mathematical expression for reactive power is

$$\text{KVAR} = (\text{VOLT} \cdot \text{CURRENT} \cdot \text{SIN}\theta) / 1000 \quad \dots \quad \dots \quad \dots \quad (2.22)$$

where, θ is the phase angle difference between voltage and current. Hence, it is evident from Eq. (2.22) that the network will have three input and one output variables. The network will converge for a three dimension, nonlinear, multiplication mapping. If the complexity of the network can be reduced to some extent, then the learning procedure of the network as well as its implementation will be much more easier. The following articles are devoted on the manipulation of input and output variables to reduce calculation complexity of the network.

2.3.1 Input Variables of the Network

The three input variables of the network are line voltage (V), load current (I), and the phase angle difference (θ) between V and I. The objective of a reactive power controller is to maintain the power factor of the system within an acceptable range. To perform this, the controller delivers reactive power to the load from a capacitor bank connected in parallel with the load. In a power system, fluctuation of voltage range within a small limit. Moreover, because both the load and the capacitor bank are placed in parallel, any fluctuation of the voltage will equally effect the amount of KVAR demanded by the load as well as the KVAR supplied

by the capacitor bank. So, we can eliminate the necessity of voltage sensing in the input. The phase angle difference, θ , can have a positive or negative value depending on whether the current is lagging or leading the voltage. To reduce complexity, θ is restricted only to its positive value. A simple module has been designed later to distinguish the lead-lag condition. Generally, in an industry, the power factor varies within 1.0 to 0.6. This practical consideration will limit the variation of θ within 0° to 52° . Again, a power factor above 0.95 is quite acceptable for the power system. So, sensing KVAR for a power factor above 0.95 is not required. This consideration prohibits the variation of θ under 18° .

The maximum current sensing capability of the circuit has been limited up to 7 amperes for the time being. Of course, this rating can be enhanced several times with the use of current transformer. Capacitor banks delivers reactive power in discrete mode. There is always a limitation that the capacitor bank cannot deliver reactive power under a certain extent. So, training patterns, where KVAR demand is below unacceptable limit, can be eliminated.

2.3.2 Output Variable of the Network

The only one output variable of the network is the KVAR requirement. The target pattern of the output is slightly modified in the computer program LEARN.FOR (Appendix-2) so that the output of the network limits within -0.4 to 0.4. It is focused that if the target pattern resembles a symmetry with the origin, then the net inputs in the hidden layers will be concentrated on the part of the sigmoid function ranging from -0.4 to 0.4. Figure 2.4 shows the graphical mapping of the two input variables with the output variable.

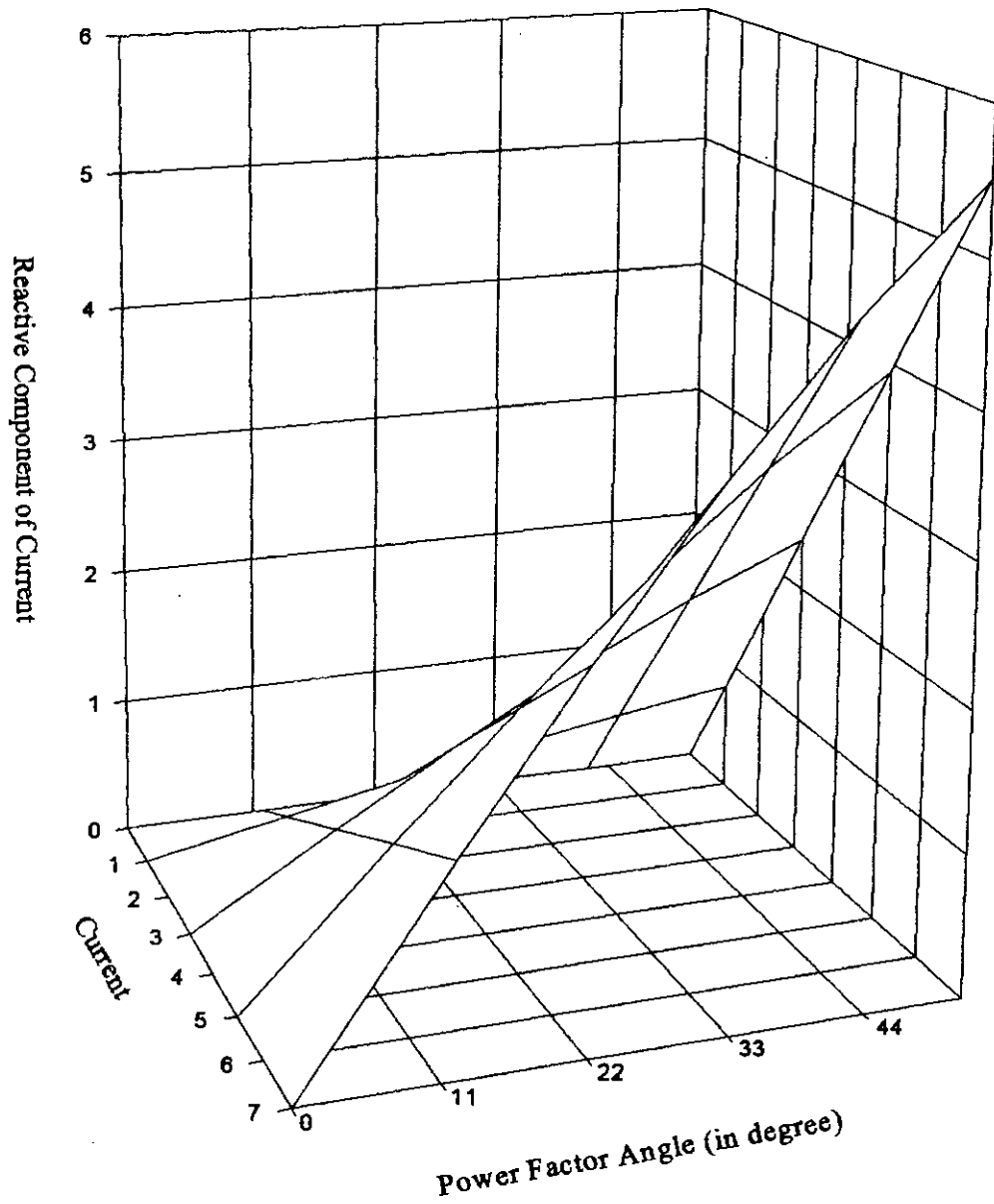


Figure 2.4 A 3-D mapping of inputs with the output.

2.3.3 Hidden Units of the Network

Initially the number of hidden units is arbitrarily chosen to be 4. The activation function of the hidden layers are flat sigmoid ($\tanh(0.5.net)$) function.

Hence, a BPN model has been introduced consisting of two input units, one output unit and 4 units of hidden layer having sigmoid activation function. For the convenience of proper convergence, all the input vectors are made fraction in the computer program LEARN.FOR (Appendix-2). The current vector is divided by an arbitrary constant 9.6 as well as the phase angle is divided by another arbitrary constant 98.7.

2.4 CONVERGENCE OF THE NETWORK

The BPN model was trained by 55 input patterns. The patterns were generated by within the program "LEARN.FOR" and was stored in a data file named "OUTPUT.DAT" (Appendix-3). Initial weights were chosen arbitrarily. At the beginning of training the learning rate was kept at 0.2. As learning proceeded and became slower, the learning rate parameter was made a function of error ($\eta = 40.error$). The weights of the network after satisfactory convergence are given below :

- **Total iteration cycle : 576440**
- **Final RMS error : 0.00033**
- **Weights for node θ : $W_{11} = 2.028, W_{21} = 0.01, W_{31} = 1.048, W_{41} = 15.535$**
- **Weights for node I : $W_{12} = 0.6044, W_{22} = 2.968, W_{32} = 2.791, W_{42} = 4.81$**
- **Weights at output : $W_1 = 4.220, W_2 = 5.976, W_3 = -7.4676, W_4 = -0.6$**

To check the convergence of the learned weight, another computer program "CHECK.FOR" (Appendix-4) is developed. This program checked the weights for 896 generated test pattern and found 844 of those data had errors less than 0.001. Figure 2.5 shows the correlation between target and output values.

For a perfect convergence the relation between target values and output values should have abided a perfect equality relation; i.e. target = output. The graphical result show that the gradient of the best fit line is 1.002 and the line constant is 0.0008756. This result indicates that output \approx target. So the obtained converging result is satisfactory.

2.4.1 Hidden Units

To observe the response of the hidden layers of the BPN network the "CHECK.FOR" program generates output data for each of the four nodes of the hidden layer. The behaviors of the four nodes for the test patterns are shown in Figure 2.6.a to 2.6.d. It will be observed that each node is activated only within a certain portion of the sigmoid function. This observation is very important because during the implementation of the hidden layers only that portion of the sigmoid function will be implemented within which the node activates. A detail analysis will be forwarded in chapter 3. Moreover, it is seen that, response of node 4 shown in figure 2.6.d has its output limited within 0.98 to 1.0. So it can be predicted that node 4 acts as a fixed bias for the network module because its output is always confined within the very small region close to unity of the sigmoid function. The fixed biasing introduced by this node is W_4 which equals to -0.6. So the pruned BPN architecture ultimately reduces to a network consisting of three hidden unit and one fixed biased unit, which is drawn in Figure 2.7.

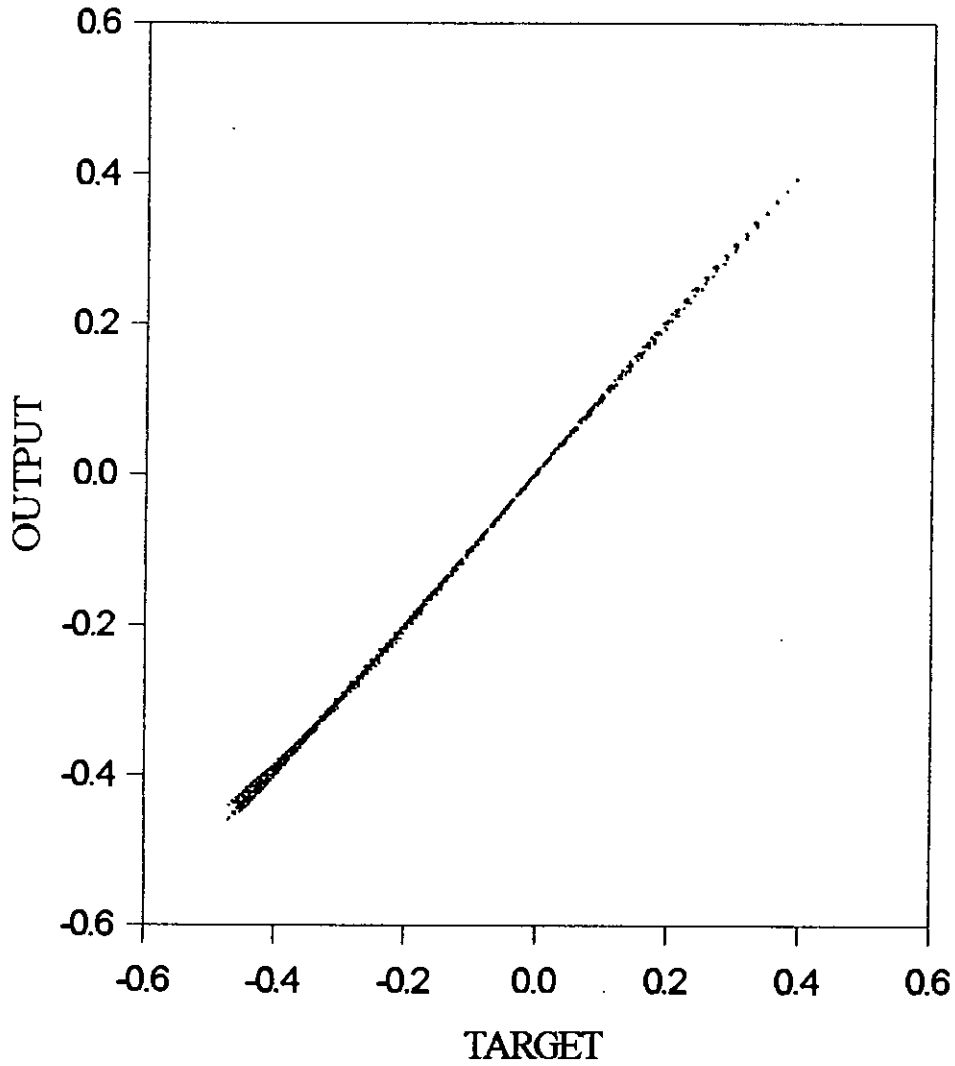


Figure 2.5 The correlation between output and target values of the test patterns. The graph shows a best fit analysis of $\text{output} = 1.002 \times \text{target} + 0.0008756$.

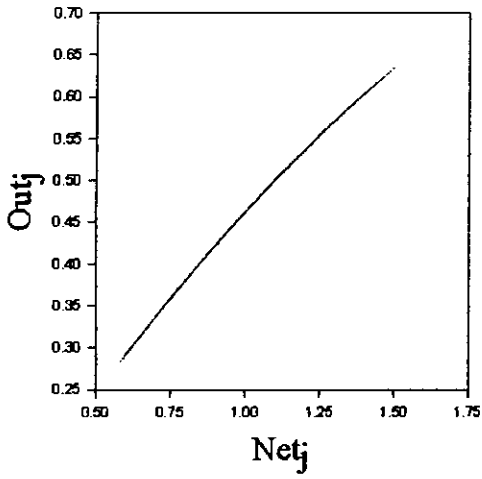


Figure 2.6.a Response of node 1.

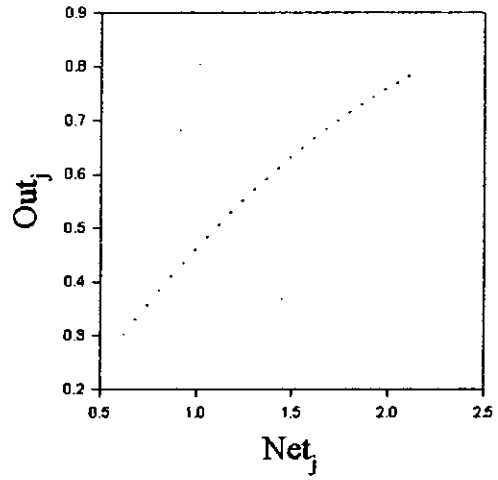


Figure 2.6.b Response of node 2.

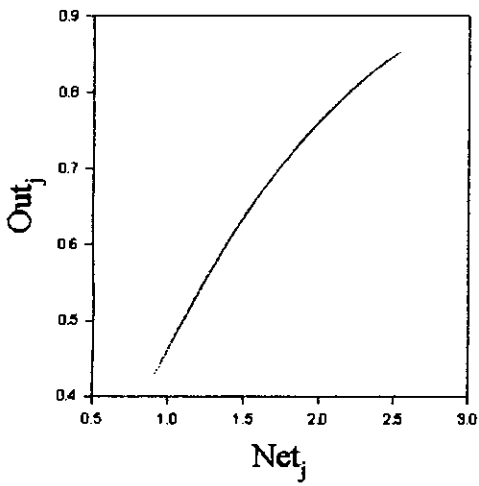


Figure 2.6.c Response of node 3.

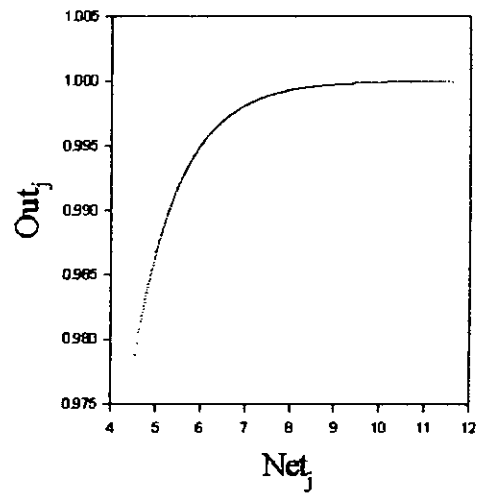
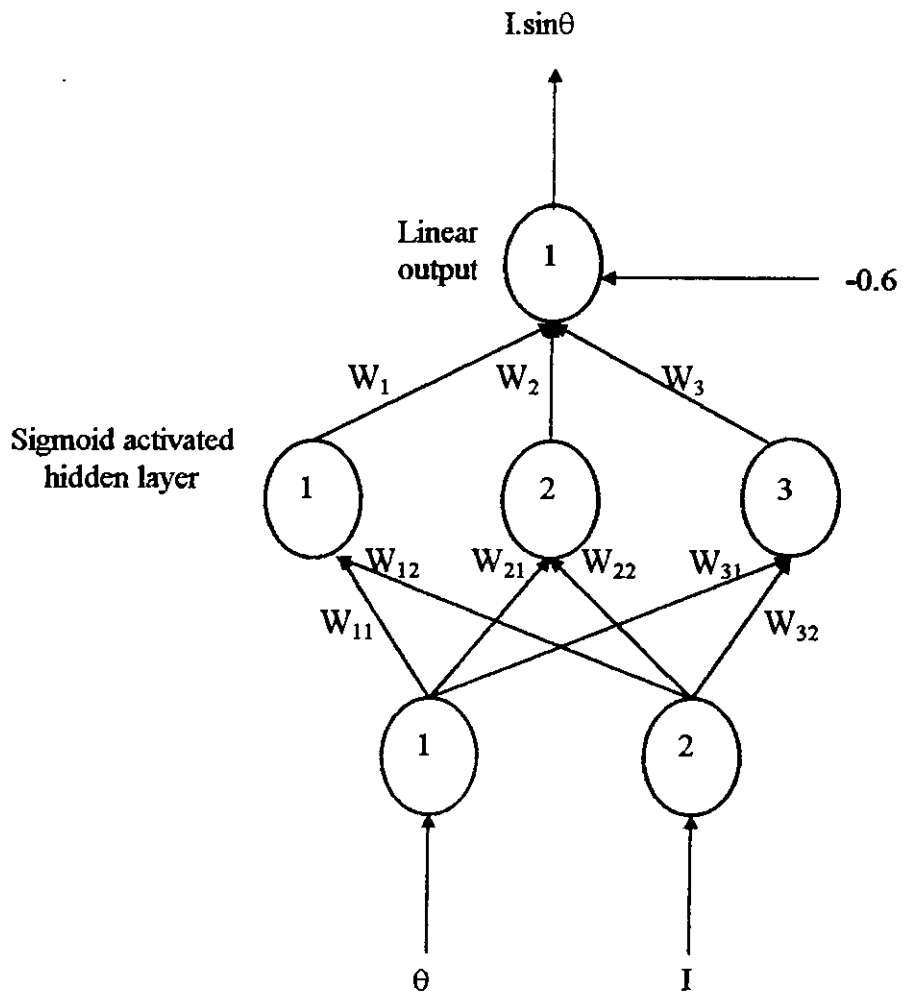


Figure 2.6.d Response of node 4.



$$W_{11} = 2.028, W_{21} = 0.01, W_{31} = 1.048$$

$$W_{12} = 0.6044, W_{22} = 2.968, W_{32} = 2.791$$

$$W_1 = 4.220, W_2 = 5.976, W_3 = -7.4676$$

Figure 2.7 BPN module for the Reactive Power Controller.

2.5 BPN WITH PIECE-WISE LINEAR ACTIVATION

In the preceding section, BPN model was developed based on a non-linear sigmoid activation function. Hopfield [32] introduced an electronic circuit using nonlinear amplifiers and resistors, which suggests the possibility of building the sigmoid activation function using VLSI technology. But it is easier to implement piece wise linear function with diodes and resistors. In this section, a new BPN model will be designed with piece-wise linear activation function in the hidden units.

2.5.1 Segmenting the Sigmoid Function

The sigmoid function will be segmented in a number of sections so that it can be closely approximated by linear functions. Figure 2.8 shows the resemblance between these two functions. The equations for PWL are given below :

$$y = 0.4475.x \quad -1.2 \leq y < 1.2 \quad \dots \quad \dots \quad (2.23.a)$$

$$y = 0.263.x + 0.2214 \quad 1.2 \leq y < 2.2 \quad \dots \quad \dots \quad (2.23.b)$$

$$y = 0.12167.x + 0.53233 \quad 2.2 \leq y < 3.2 \quad \dots \quad \dots \quad (2.23.c)$$

$$y = 0.04878.x + 0.76555 \quad 3.2 \leq y < 4.2 \quad \dots \quad \dots \quad (2.23.d)$$

$$y = 0.02955.x + 0.864634 \quad 4.2 \leq y < 5.2 \quad \dots \quad \dots \quad (2.23.e)$$

$$y = 1 \quad 5.2 < y \quad \dots \quad \dots \quad (2.23.f)$$

$$y = 0.263.x - 0.2214 \quad -1.2 > y \geq -2.2 \quad \dots \quad \dots \quad (2.24.b)$$

$$y = 0.12167.x - 0.53233 \quad -2.2 > y \geq -3.2 \quad \dots \quad \dots \quad (2.24.c)$$

$$y = 0.04878.x - 0.76555 \quad -3.2 > y \geq -4.2 \quad \dots \quad \dots \quad (2.24.d)$$

$$y = 0.02955.x - 0.864634 \quad -4.2 > y \geq -5.2 \quad \dots \quad \dots \quad (2.24.e)$$

$$y = -1 \quad -5.2 > y \quad \dots \quad \dots \quad (2.24.f)$$

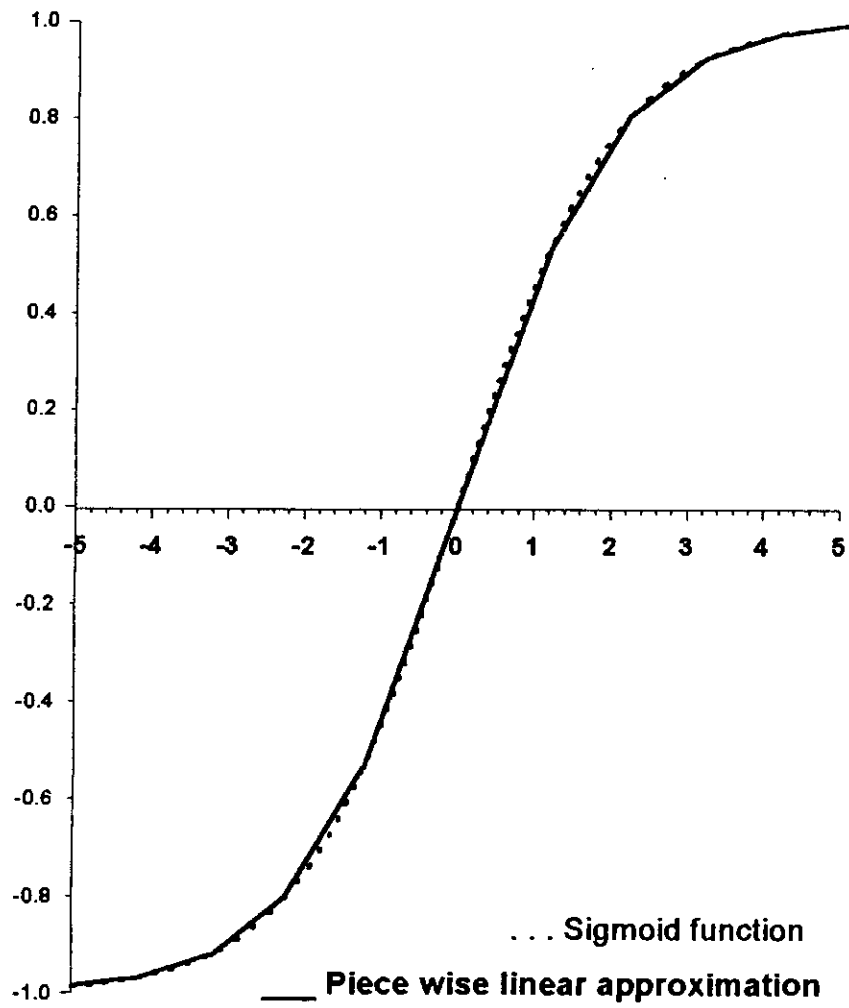


Figure 2.8 Approximating sigmoid to piece wise linear function.

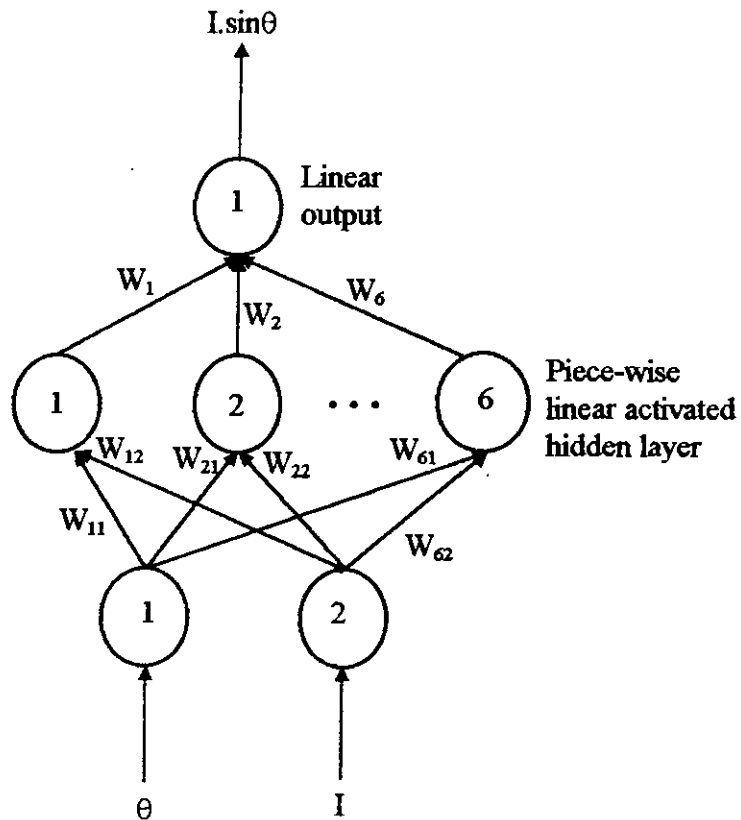
To get proper convergence, it is mentioned in the mathematics of BPN that the activation function should be differentiable. The PWL activation function proposed here may not be differentiable as a whole, but it is piece wise differentiable. So, BPN learning is possible. Though Eqs. (2.23.f) and (2.24.f) are of constant values, yet a slope of 0.01 is assumed during learning process. A computer program, LRNSRT.FOR (Appendix-5) has been developed on this respect.

2.5.2 Training of the Network

The BPN model is trained with the same 55 learning patterns used in the previous section. But this time number of hidden units has been increased to six. The learning rate parameter has been unchanged to 0.20 although the training process. After satisfactory convergence, the results of the training process are given below :

- **Total iteration cycle :** **80800**
- **Final RMS error :** **.00099**
- **Weights for node θ :** **$W_{11} = -1.644,$ $W_{21} = -1.69,$ $W_{31} = 4.77,$
 $W_{41} = 4.305,$ $W_{51} = -8.416,$ $W_{61} = 2.38.$**
- **Weights for node I :** **$W_{12} = -1.2808$ $W_{22} = 2.155$ $W_{32} = -5.376$
 $W_{42} = 3.525,$ $W_{52} = 3.2767,$ $W_{62} = -0.11$**
- **Weights for output node :** **$W_1 = -1.0811,$ $W_2 = 1.241,$ $W_3 = -0.192$
 $W_4 = -2.0332,$ $W_5 = -0.269,$ $W_6 = 1.88$**

The BPN architecture for this purpose is shown in Figure 2.9.



$$\begin{aligned}
 W_{11} &= -1.644, W_{21} = -1.69, W_{31} = 4.77, \\
 W_{41} &= 4.305, W_{51} = -8.416, W_{61} = 2.38, \\
 W_{12} &= -1.2808, W_{22} = 2.155, W_{32} = -5.376 \\
 W_{42} &= 3.525, W_{52} = 3.2767, W_{62} = -0.11 \\
 W_1 &= -1.0811, W_2 = 1.241, W_3 = -0.192 \\
 W_4 &= -2.0332, W_5 = -0.269, W_6 = 1.88
 \end{aligned}$$

Figure 2.9 The BPN architecture of the neural network module with piece-wise linear activation.

2.5.3 Convergence of the network

A computer program, CHKSRT.FOR (Appendix-6), has been developed to check the convergence of the network with 896 test patterns. The correlation between output and target values are given in Figure 2.10. Comparing this relationship with the previous correlation found in Figure 2.5 it is seen that the previous one converged better than the present case. This is expected because the final RMS error for this case is higher than the previous learning case. Yet the gradient of the best fit straight line is 1.00673 and the line constant is 0.00186 which indicates that $\text{output} \approx \text{target}$. So the converging result may be inferior to the previous learning case but it is still satisfactory. Moreover, it is realized that 100% accuracy is not required for power factor sensing, which is our ultimate desire.

The responses for each of the hidden layers are given in Figure 2.11.a to 2.11.f. These graphs show that the response of the nodes are limited within particular regions. For example, node 1,2 and 6 have response within the first two straight lines of the sigmoid function. There is only one break point for these three hidden nodes. So, it is only necessary to implement that portion of the piece-wise linear sigmoid function within which the node operates. This consideration simplifies the implementation of the hidden layer which will be explained in details in chapter 3. On the other hand node 4 and 5 have their response spreading through three break points. So during the implementation of these two nodes special attention and care has to be observed. The response graphs of figure 2.11.a to 2.11.f will be needed during the implementation of the nodes which is described in detail in chapter 3.

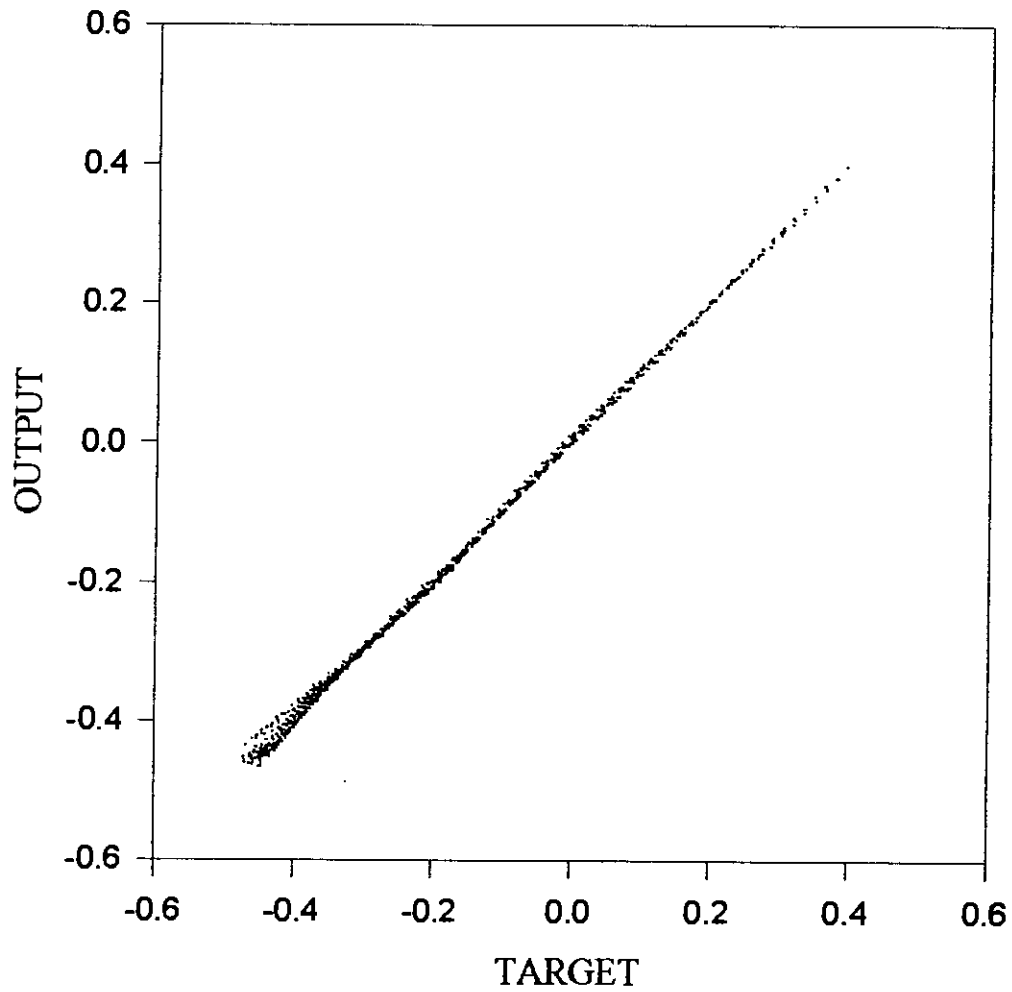


Figure 2.10 The correlation between output and target values of the test patterns. The graph shows a best fit analysis of $\text{output} = 1.00673 \times \text{target} + 0.00186$.

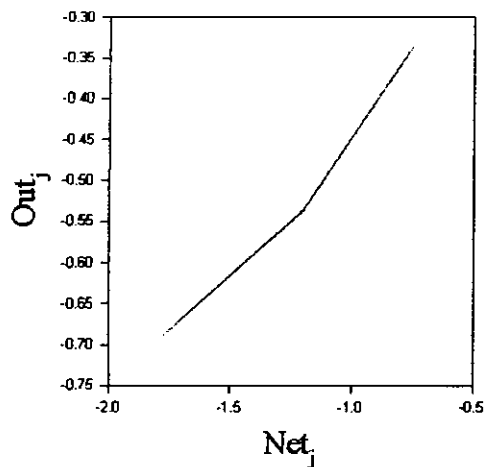


Figure 2.11.a Response of node 1

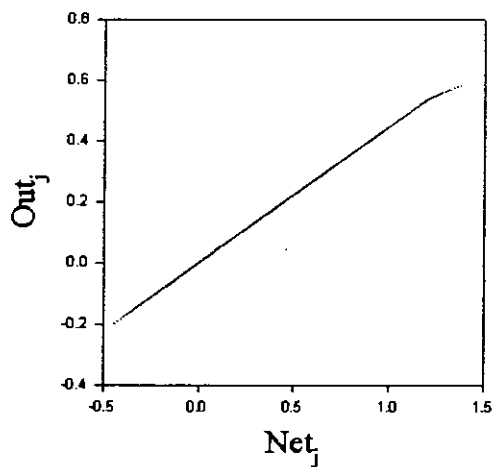


Figure 2.11.b Response of node 2

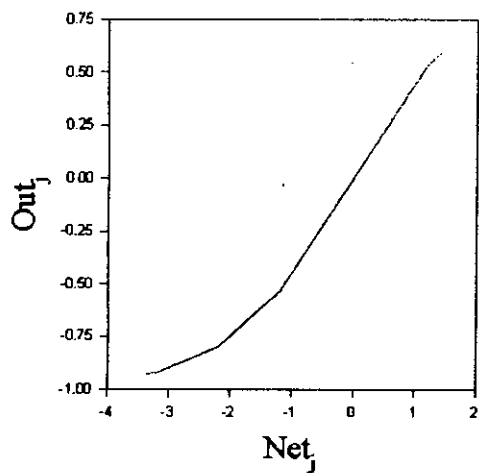


Figure 2.11.c Response of node 3

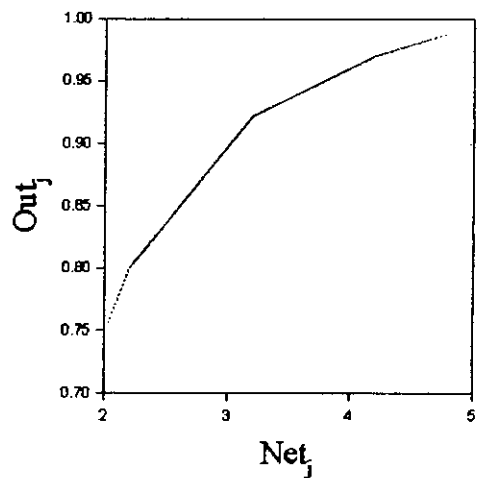


Figure 2.11.d Response of node 4

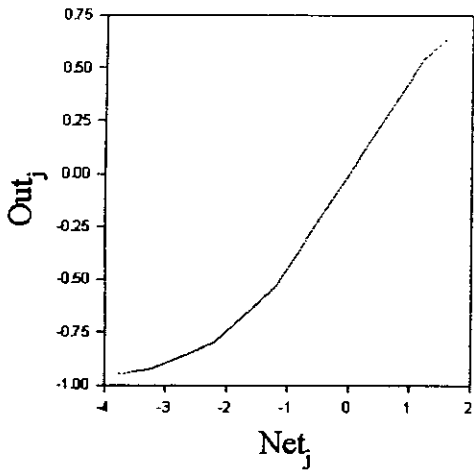


Figure 2.11.e Response of node 5.

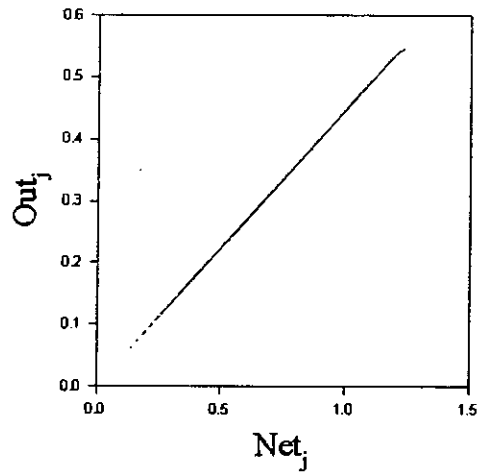


Figure 2.11.f Response of node 6.

2.6 SENSITIVITY ANALYSIS OF THE NETWORK

So far, a neural network model has been designed to perform the functional mapping $I \cdot \sin\theta$ for developing a reactive power controller. A theoretical analysis on the designed neural network will now be examined. Some interesting behaviors of the network during learning will also be put forward for future analysis in this final section.

During the period of learning process of the network with sigmoid activated function in the hidden layer, a faster convergence has occurred in comparison to the piece-wise linear activation function. This phenomena was expected because the sigmoid function is smooth and differentiable. Moreover, the number of hidden nodes required for learning is half of that required with piece-wise linear activated function. The percentage of error was also greater in the latter case. So, obviously a smooth and differentiable function is preferable in the hidden nodes. But, the present study has proven that, convergence with ramp activation may be difficult, but not impossible.

Learning rate has influenced the speed of convergence. Initially the learning rate was kept constant. As the convergence of the network became slower, the learning rate was made a function of root mean square of the error of the network. This increased the speed of convergence. But interestingly the change in learning rate had inverse effect with the network having piece-wise linear activation. Any change in the initial value of the learning parameter had increased the root mean square error of the network. So, for the second network a constant learning rate was observed.

The mapping of the function $I \cdot \sin\theta$ seems simple. But practically it has proven to be pretty difficult. Because the network has to converge for almost infinite combinations of I and θ . During the process of learning, the network used to stuck to a minimal where the change in error halted. To simplify such case, a condition was imposed on the neural network. It was considered that a minimum KVAR demand of 0.233 will be neglected by the RPC having maximum capacity of 1.2 KVAR. This consideration increased the convergence of the network to a greater extent.

The final network to be implemented by electronic circuit is shown in Figure 2.9. As it is closely observed, the response data of its nodes shows that node number 4 of the hidden layer is the most sensitive node among the others. Because, the response of node 4 ranges within 2 volts to 5 volts, and has the highest amplification weight of 2.0332 connected with the output node. So it may fairly be assumed that node 4 is the controlling node of the network and needs special attention during implementation. Observing the other nodes, it is seen that node 6 is almost dependent only on input 1. Because input 2 has a weak connection of 0.11 compared to the weight 2.38 connected with input 1.

Hence, a neural network module for the Reactive Power Controller has been designed and its various aspects are analyzed. The next chapter will concentrate on the implementation technic of the network.

CHAPTER THREE

IMPLEMENTATION OF THE NEURAL NETWORK

3.1 INTRODUCTION

This chapter explores the steps involving the development of neural network based reactive power controller. A hardware implementation of a fully analog three layer perceptron artificial neural network is presented using simple electronic tools. A methodology is proposed showing the technology to by-pass the need of amplifiers for constructing synaptic weights in the hidden layers of the neural network. Mathematical analysis is forwarded in this respect. The implementation technology introduced in this chapter is a generalized approach considering the aspects from a broad point of view. So this chapter is dedicated not only to implement the neural network based RPC, but it also invents a very unique technology of implementing neural networks in general with the simplest of electronic tools. Implementation of other auxiliary interfacing circuits for the RPC is also described in this chapter. Finally, the performance of the network has been successfully tested.

3.2 CONTROL ALGORITHM

The neural network module designed in chapter 2 consists of two inputs and one output nodes. It was elaborately explained in chapter 2 that the network will sense load current and power factor in their equivalent DC voltage and will produce an output in DC which has a linear relation with the KVAR demand of the load. The control circuits of the developed RPC will sense the output of the neural network and initiate proper switching signals to activate capacitor banks connected with the RPC to supply the KVAR. Based upon the model designed in chapter 2 a simplified line diagram of the complete NN based RPC is developed in Figure 3.1. It shows the control algorithm of the proposed analog neural network based Reactive Power Controller.

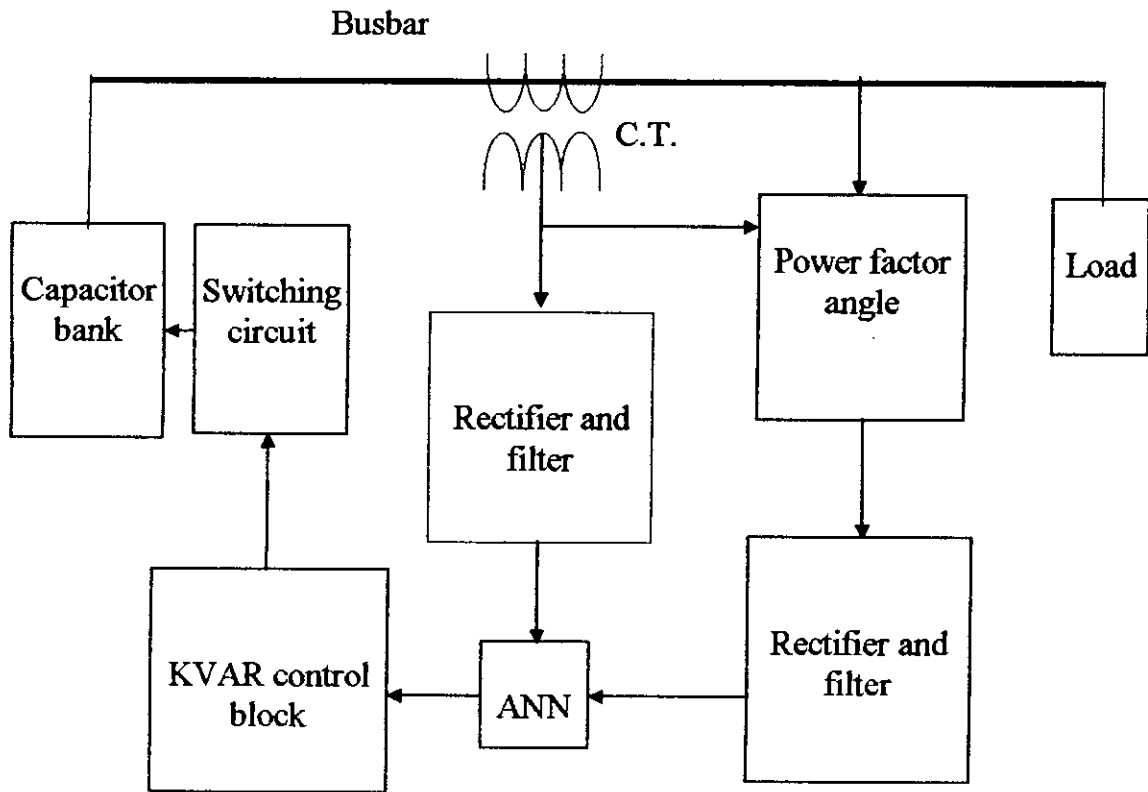


Figure 3.1 Line diagram of the neural network based RPC.

3.3 CIRCUIT IMPLEMENTATION

The hardware development of the RPC is divided in four main stages. The first stage develops the rectifier-filter circuit block to convert the AC input signals to equivalent DC inputs. The next stage invents a unique technique to implement the neural network module. The third stage describes the implementation process of the KVAR control circuit block. Finally, a brief description of the widely implemented capacitor switching block is presented.

3.3.1 Input Signal Conditioning Circuit

The neural network module for the proposed RPC has two inputs; current and power factor angle. Both of these signals should be in DC form. Simple rectifier-filter blocks is developed to perform this AC/DC conversion. The following article is presented in two separate sub-articles for the two different input signal conditioning circuits.

3.3.1.1 Current Sensing Unit

The high ampere AC load current is stepped down to maximum 5 amperes small signal AC current by a C.T. as shown in Figure 3.1. The AC current is converted to equivalent AC voltage signal by a power resistor. Half wave rectification is used for the AC/DC conversion so that both input and output signals have identical grounding. The input-output relation of the circuit should abide Eq. 3.1, which was assumed during learning of the neural network (Appendix-5). The circuit diagram of this block is given in Figure 3.2.

$$\text{DC output signal for current} = \{(\text{rms current from C.T.}) \times 16\} / 9.6 \quad (3.1)$$

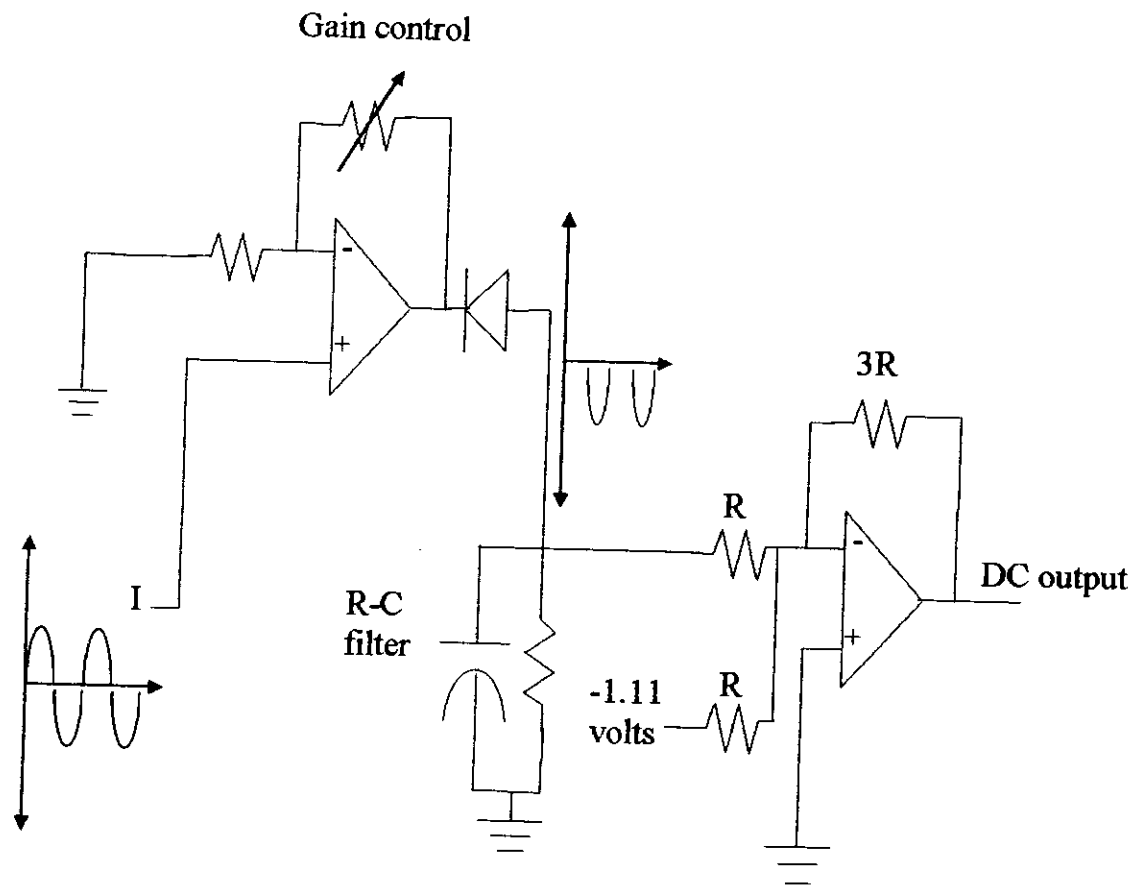


Figure 3.2 The rectifier-filter circuit for current

One interesting addition in the above circuit is a constant $-1.11 \times (-3.R/R) = 3.33$ volt DC biasing. This biasing is introduced because, during the learning process of the neural network module described in chapter 2, the current range was limited within 2 ampere to 7 ampere for the ease of convergence in the learning software routine “LRNSRT.FOR” (Appendix-5). Eq. 3.1 indicates that rms 2 ampere corresponds to 3.33 DC voltage. The current range can be changed to the conventionally used 0 to 5 ampere range simply by introducing a DC biasing of 3.33 voltage. Hence, an input current of rms 0 to 5 ampere should be linearly converted into 3.33 to 11.66 DC voltage through this implemented circuit. The gain control resistance is provided to establish this linear relationship.

3.3.1.2 Power Factor Sensing Unit

The phase angle difference between current and voltage is converted in equivalent DC voltage through the circuit shown in Figure 3.3. The current signal is taken from the power resistor terminal and the voltage signal is taken from a simple voltage divider. An equivalent voltage drop across the power resistor will act as the reference for the sensing of current. On the other hand, a small sample of the 220 volt power line from the voltage divider will be sufficient to obtain the phase angle between the voltage and the current. The width of the output pulses from the comparator shown in Figure 3.3 is directly related with the phase angle of the power line. Finally, the pulses are rectified and converted to pure DC voltage. The relation between phase angle and DC signal output is linear from 18° to 52° and follows Eq. 3.2. The variable gain control resistance will help to establish this relation.

$$\text{DC output signal for phase shift} = \{(\text{phase shift in degrees}) \times 16\} / 98.6 \quad (3.2)$$

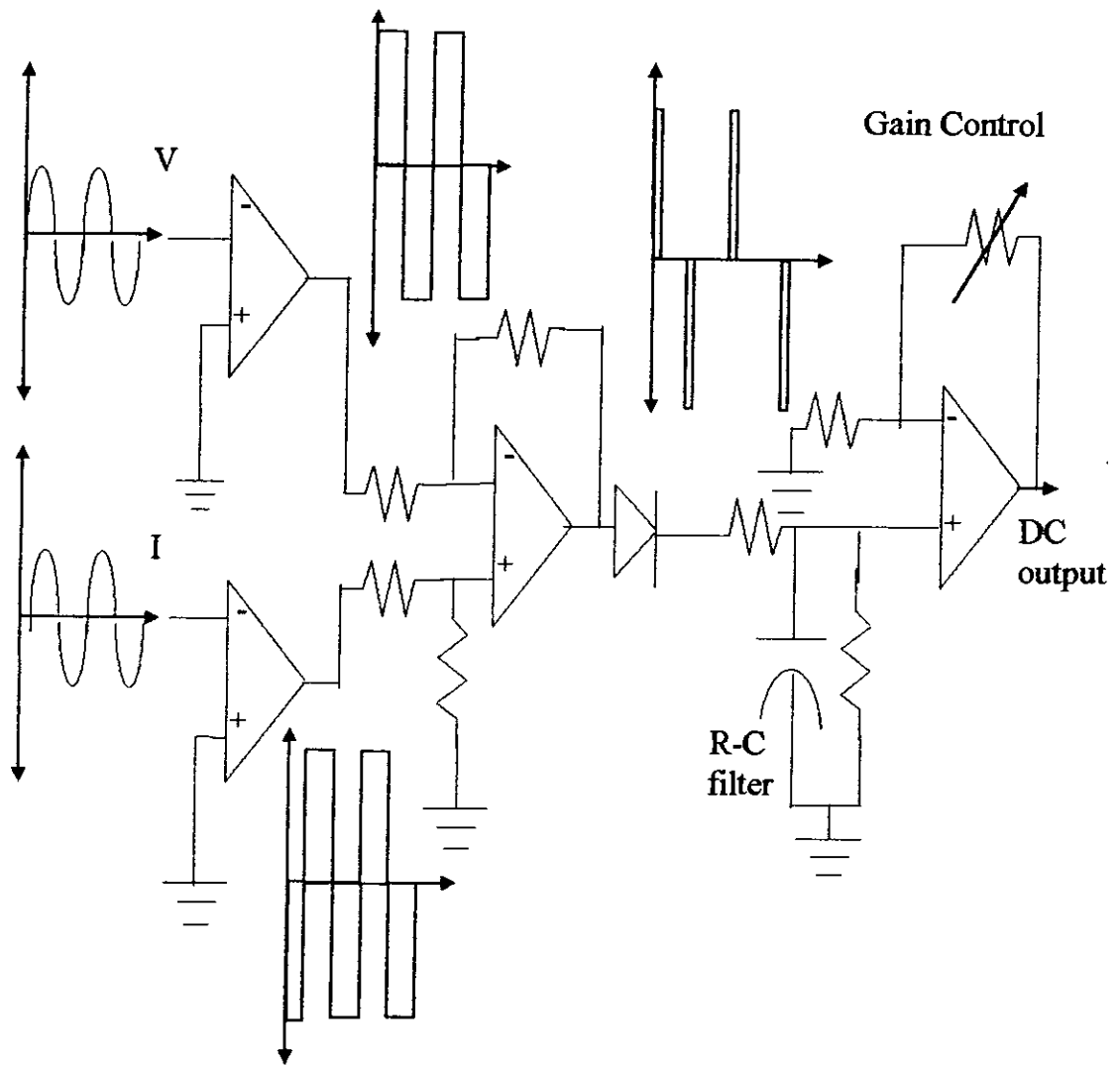


Figure 3.3 Rectifier-filter circuit for power factor sensing.

3.3.2 Simulation of Activation Function

This section describes the technique used to implement the neural network module. The fabrication of the neural module is based on the theoretical investigation of the converged network described in chapter 2. It has been mentioned in the conclusion of chapter 2 that piece-wise linear (PWL) activation function will be implemented for the purpose. This section also gives a broad description of the methodology forwarded for implementing neural networks in a generalized way. Before going through the implementation details, a general mathematical analysis of the neural network is forwarded.

3.3.2.1 Mathematical Analysis

The mathematical analysis forwarded in this section is applicable for any general neural network having piece-wise linear activation function. So, instead of constricting the mathematics only within piece-wise linear sigmoid function, a more general approach has been taken by considering the ramp function as the activation function of the hidden layers of the neural network having 'n-1' number of break points with 'n' number of different gradient lines.

Figure 3.4 shows the j-th node of the hidden layer of neural network having PWL activation function. The break points of the function are represented by C_1 , C_2 and C_3 , and the slopes are denoted by m_1 , m_2 , m_3 and m_4 . The number of inputs to the j-th node of the hidden layer is 'L'. Weights connected from the inputs to the j-th node are W_{j1} , W_{j2} , W_{j3} , W_{jn} . The output of the j-th node connected with the output layer of the neural network is O_j .

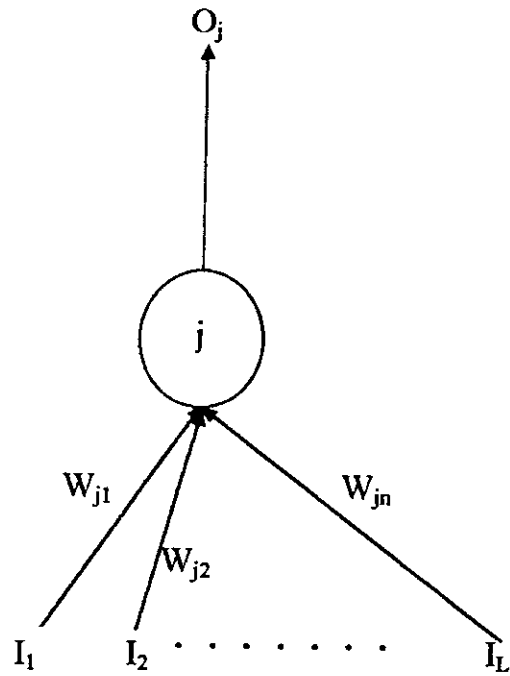


Figure 3.4.a j -th node of the hidden layer

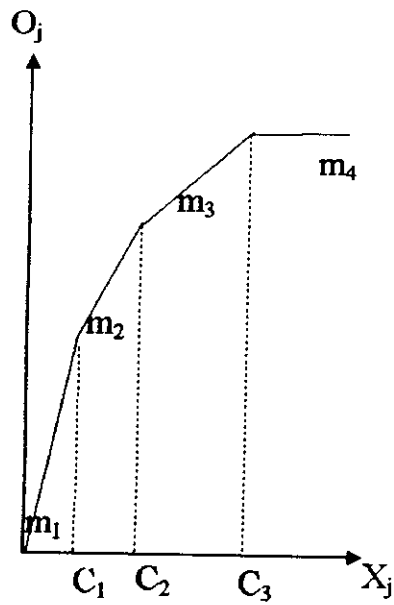


Figure 3.4.b PWL function of the hidden layer

For a generalized solution, the sigmoid function is considered having 'n' no. of slopes and 'n-1' no. of break points. The output to the j-th node O_j is related with the input X_j by

$$X_j = \sum_{i=1}^L I_i \cdot W_{ji} \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.3)$$

$$O_j = m_1 \cdot X_j, X_j \leq C_1 \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.4)$$

and, $O_j = m_n \cdot X_j + \sum_{r=1}^n (m_{r-1} - m_r) \cdot C_{r-1}, C_{n-1} < X_j \leq C_n \quad \dots \quad \dots \quad (3.5)$

Equating Eq.(3.3), (3.4) and (3.5),

$$O_j = m_1 \cdot \sum_{i=1}^L I_i \cdot W_{ji}, X_j \leq C_1 \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.6)$$

$$O_j = m_n \cdot \sum_{i=1}^L I_i \cdot W_{ji} + \sum_{r=1}^n (m_{r-1} - m_r) \cdot C_{r-1}, C_{n-1} < X_j \leq C_n \quad \dots \quad (3.7)$$

3.3.2.2 Implementation

The above equations govern the relations among inputs and outputs of a neural network having PWL activation. The equations were developed in a generalized way. So, the implementation technology developed in this section is also described in a broad and extensive way; applicable for implementing any neural networks. Figure 3.5 shows a simple electronic circuit with diodes and resistances to fabricate the PWL function together with the weights connected with the hidden nodes.

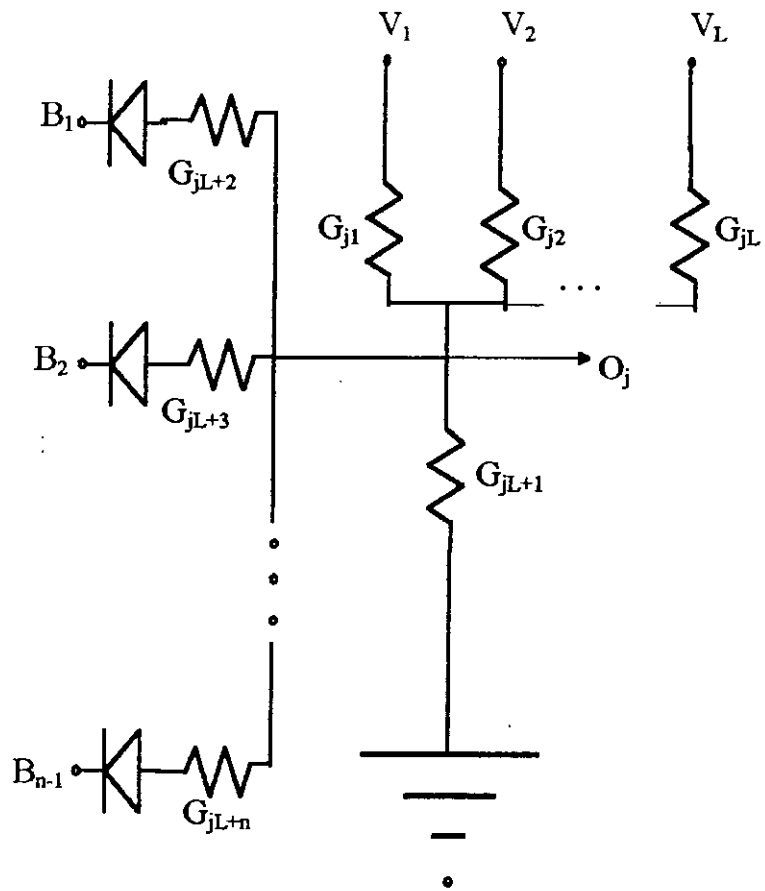


Figure 3.5 A simple voltage divider circuit

In the circuit shown in Figure 3.5, the input signal I_i of the NN module is considered proportional to the voltage V_i . Output voltage O_j for $|O_j| \leq |B_1|$ is,

$$O_j = \frac{\left[\sum_{i=1}^L V_i \cdot G_{ji} \right]}{\left[\sum_{i=1}^{L+1} G_{ji} \right]} \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.8)$$

Solving Eq. (3.6) and Eq. (3.8)

$$W_{j1} / G_{j1} = W_{j2} / G_{j2} = \dots = W_{jL} / G_{jL} \quad \dots \quad \dots \quad \dots \quad (3.9)$$

$$\text{and, } G_{jL+1} = (1-m_1 \cdot \sum_{i=1}^L W_{ji}) \cdot G_{j1} / (m_1 \cdot W_{j1}) \quad \dots \quad \dots \quad \dots \quad (3.10)$$

Assuming a suitable and practical value for one of the resistances the values of the others can be established. For $|B_2| \geq |O_j| > |B_1|$,

$$O_j = \frac{\left[\sum_{i=1}^L V_i \cdot G_{ji} \right] + [B_1 \cdot G_{jL+2}]}{\left[\sum_{i=1}^{L+2} G_{ji} \right]} \quad \dots \quad \dots \quad \dots \quad (3.11)$$

From Eq. (3.7) and Eq. (3.11)

$$G_{jL+2} = [(1-m_2 \cdot \sum_{i=1}^L G_{ji}) \cdot G_{j1} / (m_2 \cdot W_{j1})] - G_{jL+1} \quad \dots \quad \dots \quad (3.12)$$

$$\text{and, } B_1 = [C_1 \cdot (m_1 - m_2) \cdot \sum_{i=1}^{L+2} G_{ji}] / G_{jL+2} \quad \dots \quad \dots \quad \dots \quad (3.13)$$

From Eq. (3.12), the value of G_{jL+2} is obtained, while Eq. (3.13) establishes the first breaking potential. A generalized equation substituting (3.12) and (3.13) are

$$G_{jL+n} = [(1-m_n \cdot \sum_{i=1}^L G_{ji}) \cdot G_{j1} / (m_n \cdot W_{j1})] - \sum_{i=1}^{n-1} G_{jL+i} \quad \dots \quad (3.14)$$

$$\text{and, } B_{n-1} = [\{ \sum_{r=1}^n (m_{r-1} - m_r) \cdot C_{r-1} \cdot \sum_{i=1}^{L+n} G_{ji} \} - \sum_{i=1}^{n-2} B_i \cdot G_{jL+i+1}] / G_{jL+n} \quad (3.15)$$

It is evident from Eq. (3.10) that for a non negative value of G_{jL+1} the condition that should be imposed is,

$$m_1 \cdot \sum_{i=1}^L W_{ji} < 1 \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.16)$$

So, there is a fair amount of possibility that all nodes cannot be implemented with a voltage divider circuit. Yet, this condition can be satisfied with some adjustment to the input values. Eq. (3.7) shows that the weights can be attenuated by a certain factor without hampering the output O_j , provided that the inputs are conversely amplified by the same factor. So, by choosing a proper attenuating factor Eq. (3.16) can be satisfied. It is not necessary to implement the whole PWL function for every node. Rather, it is essential to implement that portion of the PWL function within which the node operates. This consideration has simplified the implementation of the hidden layer. Finally, the output nodes are developed by using analog adders. Thus, necessity of using amplifier to implement weight for the hidden layer is eliminated.

Thus, it has been shown that a voltage divider circuit is quite capable to be used as the nodes of the hidden layer. The neural network developed for the reactive power controller is described in appendix-7.

3.3.3 KVAR Control Circuit

In the previous section the neural network module was developed to sense the amount of KVAR required by the power system. A linear relationship between the output of the neural network and the KVAR demand was obtained. This section explains the development of the control circuitry required to switch proper capacitor banks to improve the power factor of the system.

So far it was considered during the training of the neural network that maximum 7 amperes of current will be sensed by the network. The operation of the network showed that for this maximum current rating, the output of the neural network is 2.8 volts and the minimum KVAR sensed by the neural network corresponds an output of -3.03 volts. Hence, for an N step capacitor switching circuit the output range of -3.03 volt to 2.8 volt of the network should be divided in N linear ranges. The proposed RPC is designed with 3 stages of capacitor switchings. For such case, a maximum of $2^3 - 1 = 7$ steps can be obtained. The KVAR control circuit of the designed RPC is implemented mainly with two IC's; LM3914 and 74LS148. Along them, an inverter IC 7404 and a voltage regulated IC 7805 were also used for the KVAR control block. The pin and block diagram of only the main two IC's are described in the following article.

3.3.3.1 Function of the IC's

LM3914 : This IC is a simple analog level triggering chip consisting of 10 voltage comparators. This IC is widely used in bar displaying and is capable of driving 10 LEDs together. In this case, it is used to separate the linear output range of the neural network in seven uniformly divided sections. The internal components of this IC is shown in Figure 3.6.

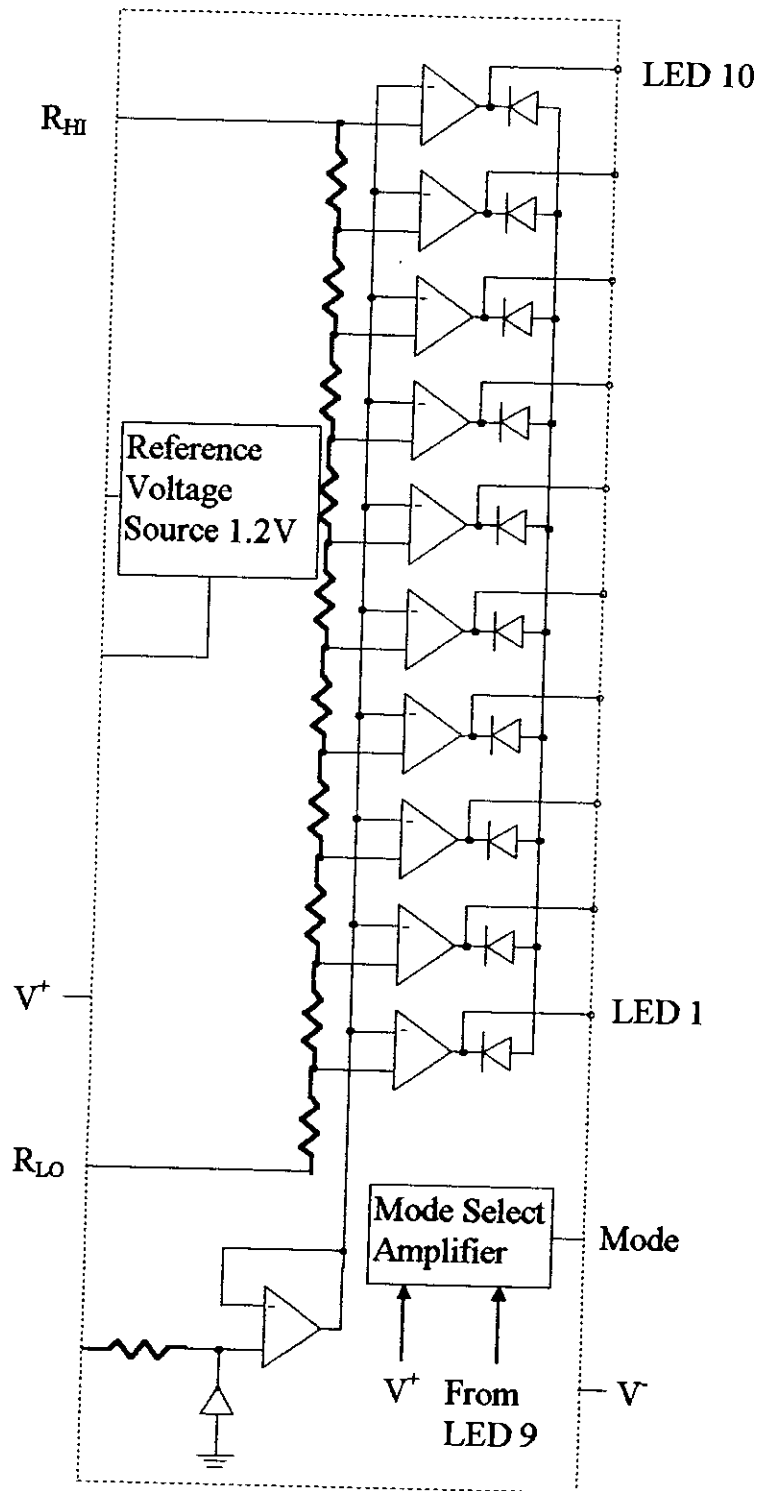


Figure 3.6 Internal representation of IC LM3914

The figure shows that there are 10 voltage level comparators. The number of those comparators being used for the KVAR control circuit depends on the number of steps demanded in the capacitor switching stage. The biasing voltage, V_H and V_L , applied to the potential divider circuit of LM3914 will be obtained from solving the two Eq.s 3.17 and 3.18 given below:

$$(V_H - V_L) \cdot (N/10) + V_L = V_{HT} \quad \dots \quad \dots \quad \dots \quad (3.17)$$

$$(V_H - V_L) \cdot (1/10) + V_L = V_{LT} \quad \dots \quad \dots \quad \dots \quad (3.18)$$

Where :

V_H = Biasing voltage applied to R_{HI}

V_L = Biasing voltage applied to R_{LO}

V_{HT} = Threshold voltage to trigger maximum KVAR demand

V_{LT} = Threshold voltage to trigger minimum KVAR demand

N = No. of capacitor switching steps.

74LS148 : This is a Decimal/Binary high priority encoder chip. The pin diagram of this IC is shown in Figure 3.7.

The N signal outputs from LM 3914 will be the inputs of 74LS148. The three outputs of the encoder chip will generate binary signals equivalent to its highest input being activated. The binary signals will activate corresponding capacitor switching relays which is explained elaborately in the next section.

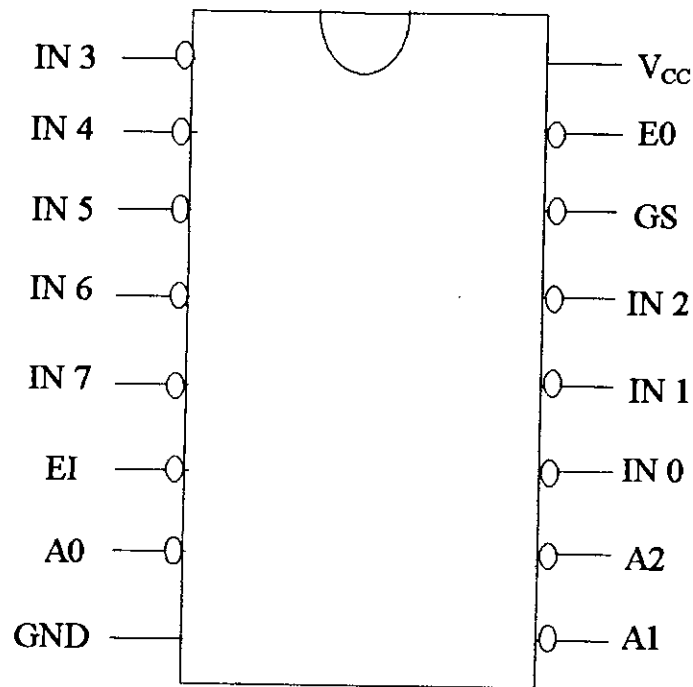


Figure 3.7 The pin diagram of 74LS148

3.3.3.2 Elimination of Low KVAR Demand

Recalling back to the learning algorithm of the neural network developed in chapter 2 reminds that the network was not trained for patterns corresponding to very low KVAR demands. A simple logic function was introduced in the program “LRNSRT.FOR” (Appendix-5) for this purpose. The logic block divides the output function $I.\sin\theta$ in two regions by two straight lines defined by Eq. 3.19 and Eq. 3.20.

$$\text{Input}_1 \times 0.25 + \text{Input}_2 = 0.2675 \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.19)$$

$$\text{Input}_1 + \text{Input}_2 = 0.53 \quad \dots \quad \dots \quad \dots \quad \dots \quad (3.20)$$

It was observed that the two input data combinations falling above both of the lines gives a KVAR output greater than 0.2 KVAR. So, any input combinations not satisfying the equations or having outputs less than the right hand side of the equations will be eliminated through the logic block. In this way the low KVAR demand of the circuit can be removed.

The block representing this logic function is implemented with three comparators shown in Figure 3.8. The upper comparator having output X will be positive; i.e. +15 volts only when the two input combinations lies above the Eq. 3.20. Similarly, the lower comparator having output Y will be positive; i.e. +15 volts only when the two input combinations lies above the Eq. 3.19. Finally the three resistance matrix is built to create an analog logic circuit. A brief analysis is presented in table 3.1. The output of this logic block will be used to activate the encoder chip by connecting it to the $\overline{\text{EI}}$ pin of 74LS148. The complete connection diagram of the control circuit is given in Figure 3.9.

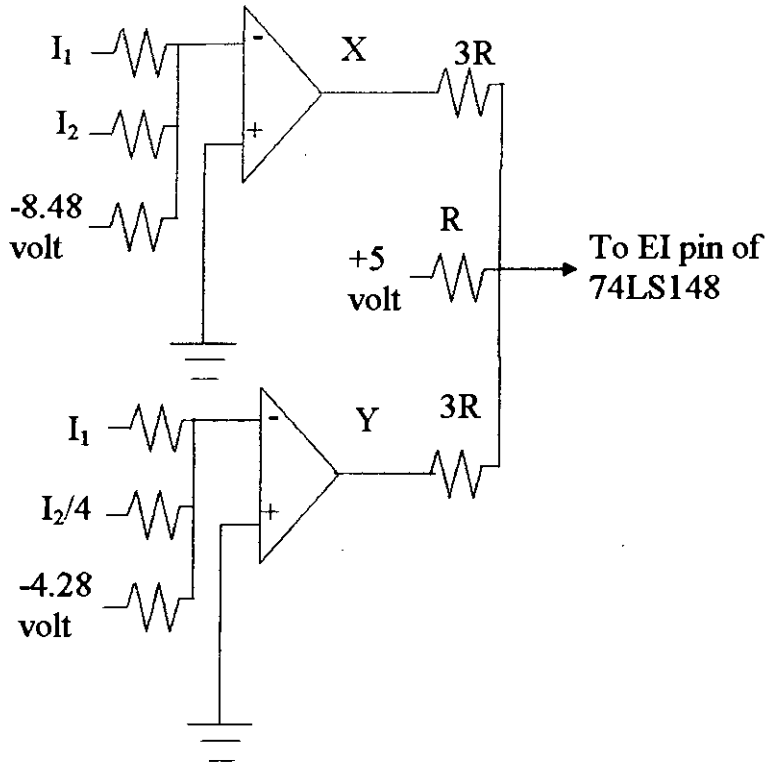


Figure 3.8 The logic block prohibiting the KVAR control circuit to operate at very low KVAR demand

Operation of the logic block:

$X = -15$ volts if $I_2 + I_1 > 8.48$ volts; else $X = +15$ volts

$Y = -15$ volts if $I_2/4 + I_1 > 4.28$ volts; else $Y = +15$ volts

Output = $X/5 + Y/5 + 2.0$

X	Y	Analog output	Logic output	Operation of 74LS148
+15	+15	8	1	Prohibits
+15	-15	2	1	Prohibits
-15	+15	2	1	Prohibits
-15	-15	-4	0	Inhibits

Table 3.1 Operation of the logic block

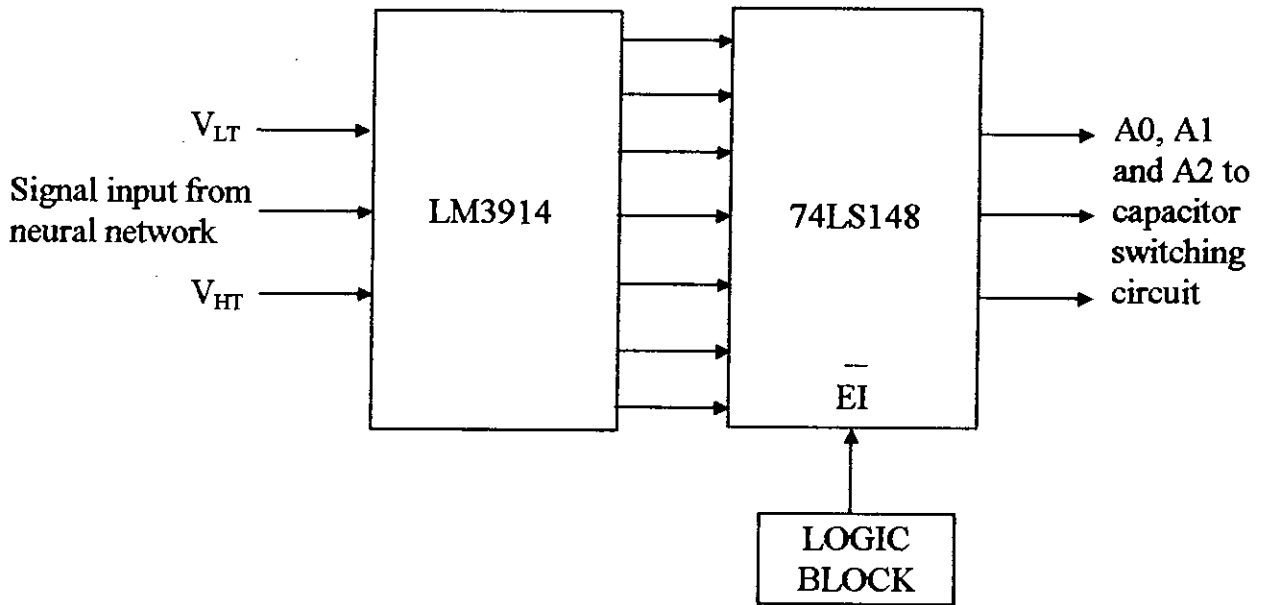


Figure 3.9 The block diagram of KVAR control circuit

3.3.4 Capacitor Switching Circuit

The three controlling signals A0, A1 and A2, obtained from the KVAR control circuit described in the previous article, is used to initiate three individual 5 volt DC normally open relays. These relays are in series with the magnetic contactors of each capacitor banks. The block diagram of capacitor switching circuit is shown in Figure 3.10.

The three controlling signals A0, A1 and A2 drives the base of the transistors to saturation or to cut off region according to the digital signals generated by the KVAR control circuit. Hence, the transistors acts as switches for the DC relays. The DC relays are initiated by these signals from the KVAR control circuit, current flows in the coil of the magnetic contactors from L_1 to L_2 and the capacitor banks are thus connected with the three bus bars. The connected capacitor bank delivers reactive power to improve power factor.

This chapter has described the implementation of the NN based RPC. The next chapter will discuss on the results obtained during the experiments done on the implemented circuits and highlight on the overall performance of the network.

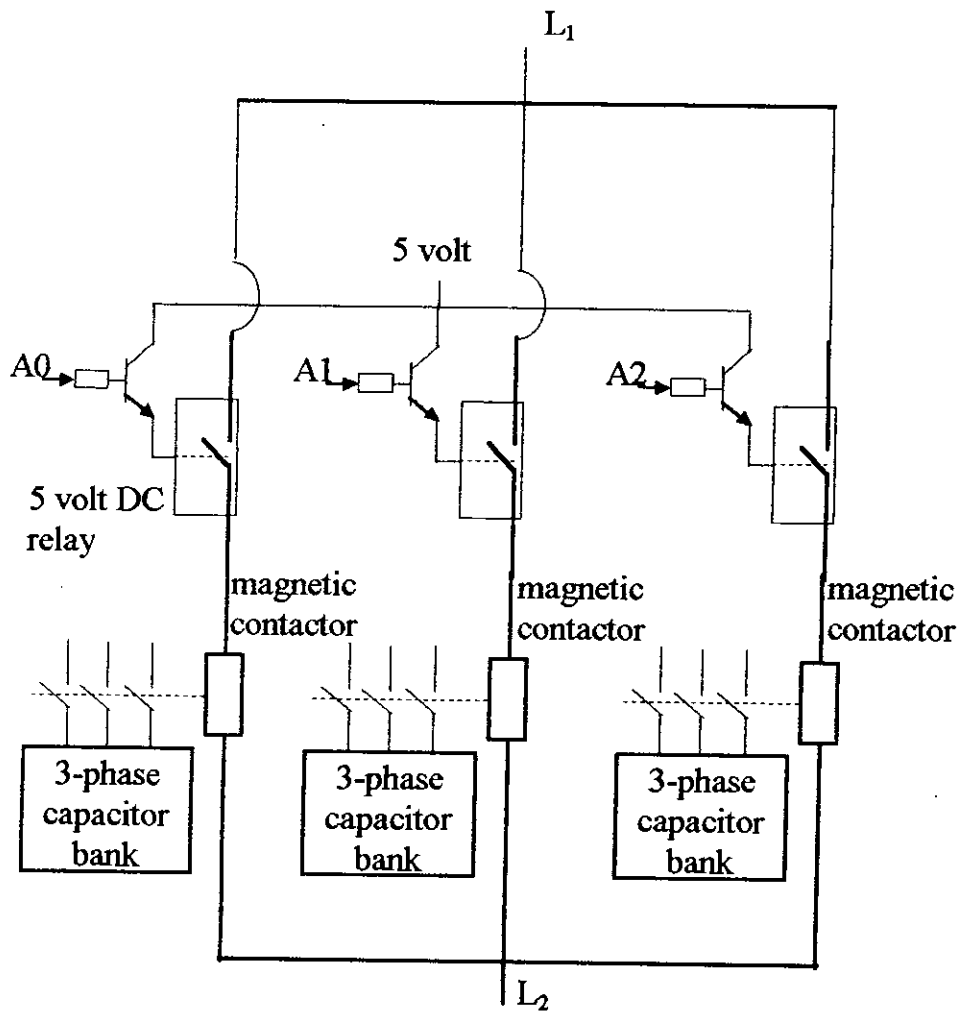


Figure 3.10 Capacitor switching circuit

CHAPTER FOUR

RESULTS

4.1 INTRODUCTION

The neural network based reactive power controller was developed systematically step by step. Design and implementation technology for developing the operating modules for the NN were elaborately described in chapter 3. All those modules were tested part by part and necessary measures were taken for precise and accurate operations of the modules. These modules were finally joined together to built the neural network based reactive power controller. At the end rigorous laboratory tests were performed on the implemented network. This chapter forwards the experimental setups and test results performed on the process of developing the neural network.

4.2 PERFORMANCE TESTING ON THE NN MODULES

Chapter 3 has divided the complete network in four sections. They are: Input signal conditioning unit, Activation function section, KVAR control block and Capacitor bank switching circuit. These four modules were developed separately and performance of all these modules were tested precisely. This section elaborates those tests in the following articles.

4.2.1 Testing on Input Signal Conditioning Unit

The neural network has two input nodes. They are: Current sensing unit and Power factor sensing unit. The circuits implemented for these two units are described in sections 3.3.1.1 and 3.3.1.2. The desired response to be obtained from these two units are expressed in Eq. 3.1 and Eq. 3.2. To observe the actual response of both of these units a laboratory test was performed. The experimental setup is presented in Figure 4.1.

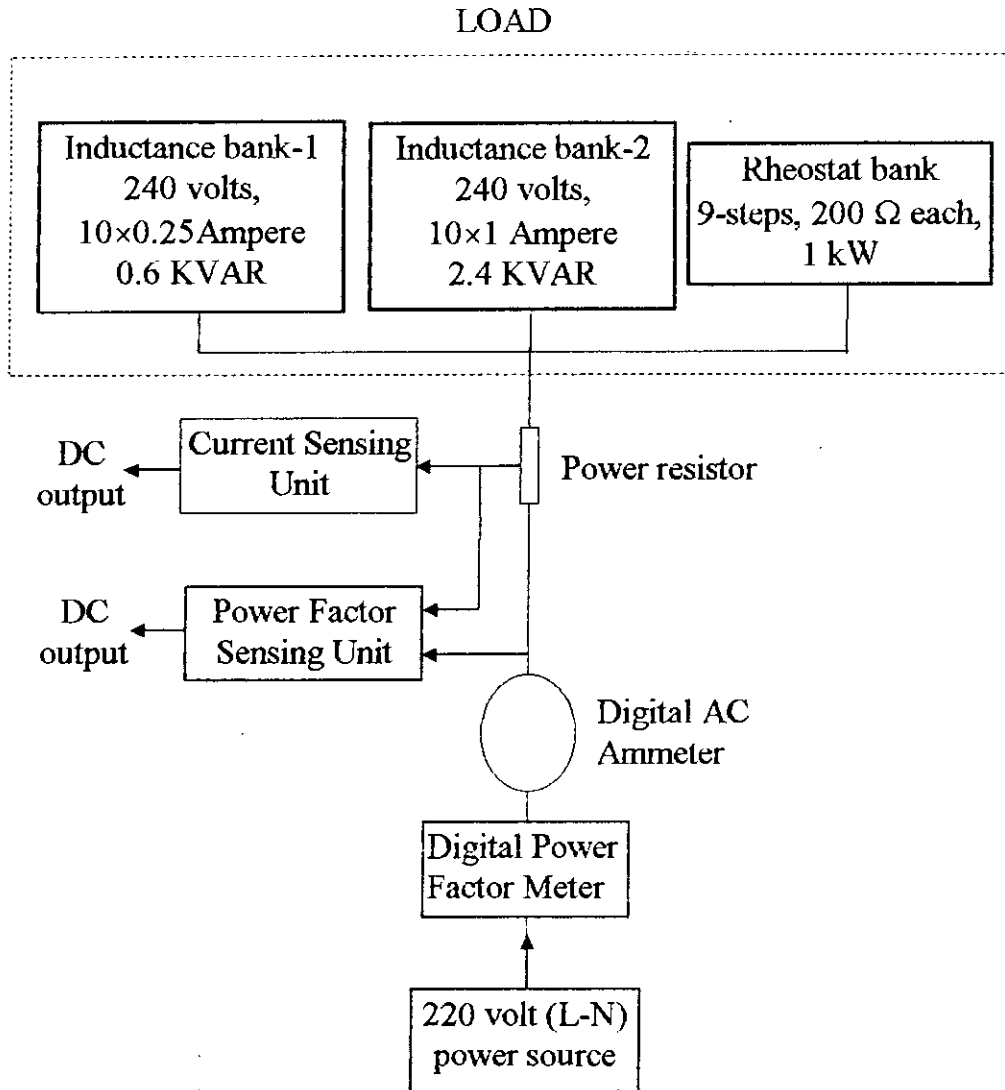


Figure 4.1 Experimental setup for testing input signal conditioning unit.

Different combinations of loading were presented to the input units. It was explained in article 2.3.1 and article 3.3.1.1 that the current sensing of the neural network is limited within 2 ampere to 7 ampere and the power factor angle sensing is confined within 1.0 to 0.6 lagging. So, testing was performed within this extent. Results obtained from the test are placed in table 4.1.a and 4.1.b. The graphical output of the desired response and the actual response for both of the input signal conditioning blocks are forwarded in Figure 4.2.a and 4.2.b. Both of the graphs shows a remarkable resemblance between the obtained result with the expected values. The test proved the input module to be operating satisfactorily.

4.2.2 Testing on Hidden Nodes

The developed neural network has six hidden layers. Implementation of the hidden nodes were explained in section 3.3.2.2 and appendix-7. These sections have elaborately explained modifications done on the weights and the breaking potentials. Resistance values for each node were also presented in Table A-7. Response expected from these nodes during learning are shown in Figure 2.11. But due to the modifications mentioned above, the response curves shown in Figure 2.11 will need to be extended 3 times along both axis. To test the actual response of the developed hidden nodes, a continuous AC wave of 15 volt peak to peak was applied to one of the two inputs keeping the other one grounded. The output of the nodes were observed in the oscilloscope in X-Y mode. Same test was done for the input previously grounded; now keeping the alternate input grounded. The test setup is given in Figure 4.3. A comparison between the expected and actual response for node 4 is given in Figure 4.4. The graph shows satisfactory results. Same results were obtained for other nodes also.

Table 4.1.a Test results for current sensing unit

No. of obs.	Load Current in r.m.s.	Corresponding D.C. voltage
1	2.02	3.33
2	2.83	4.45
3	3.71	6
4	4.68	8
5	5.70	9.5
6	6.70	11.5
7	6.96	12

Table 4.1.b Test results for power factor sensing unit

No. of obs.	Power factor angle in degrees	Corresponding D.C. voltage
1	12.24	2
2	20.16	3.27
3	30	5.2
4	39.6	6.4
5	41.4	6.8
6	54	8.5
7	60	9

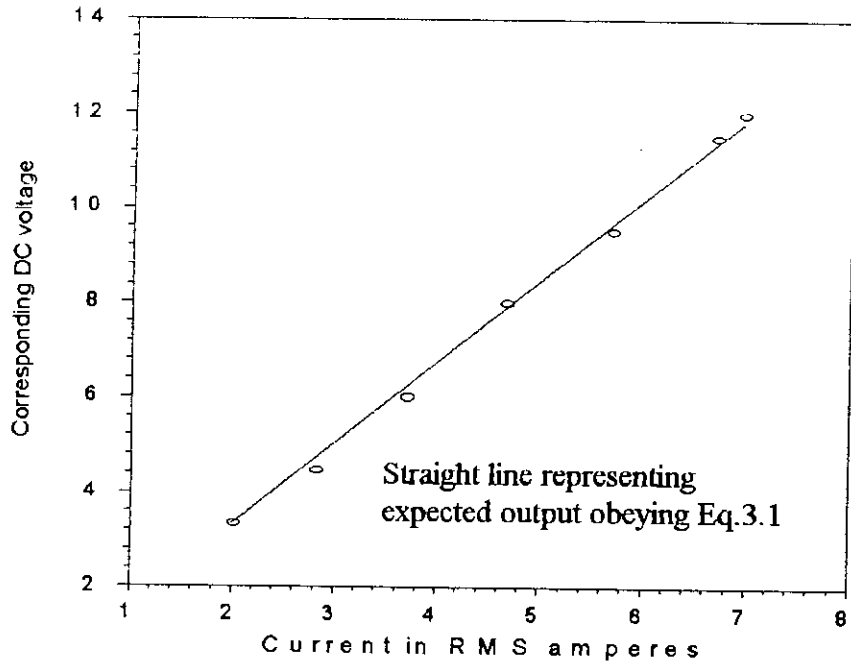


Figure 4.2.a Response of current sensing unit

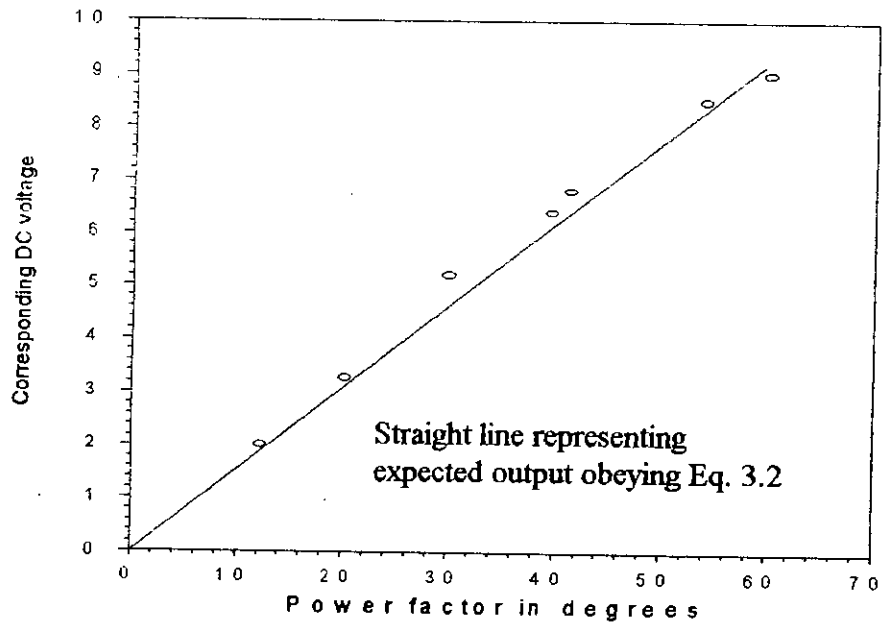


Figure 4.2.b Response of power factor sensing unit

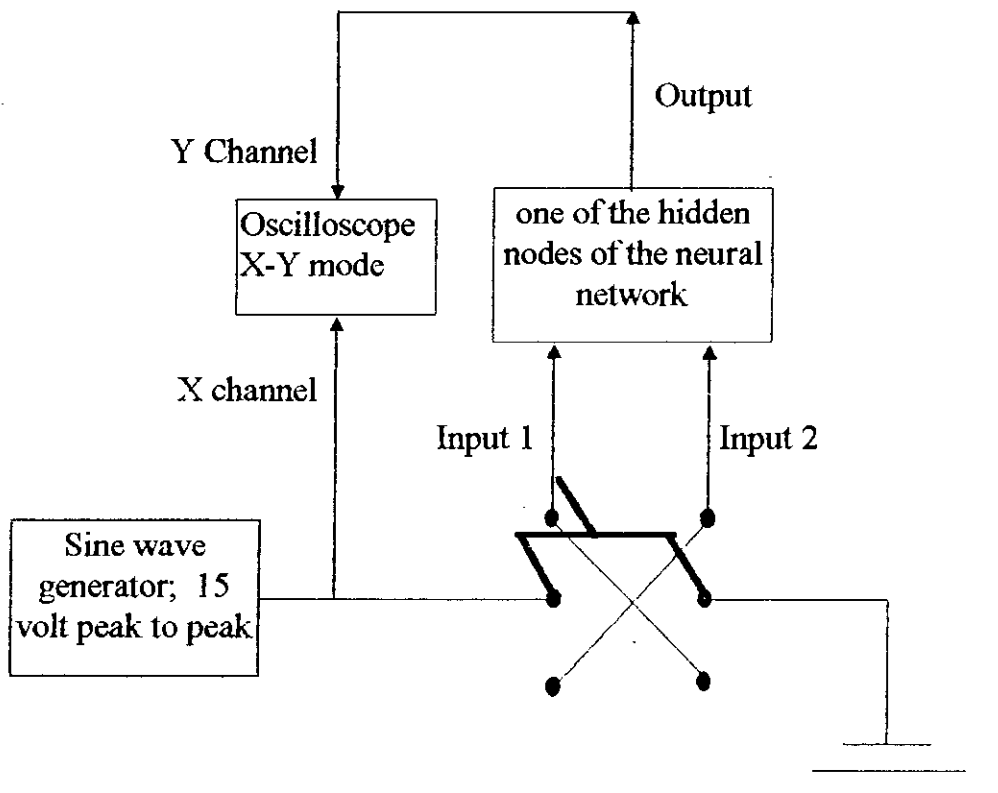


Figure 4.3 Test setup for hidden nodes

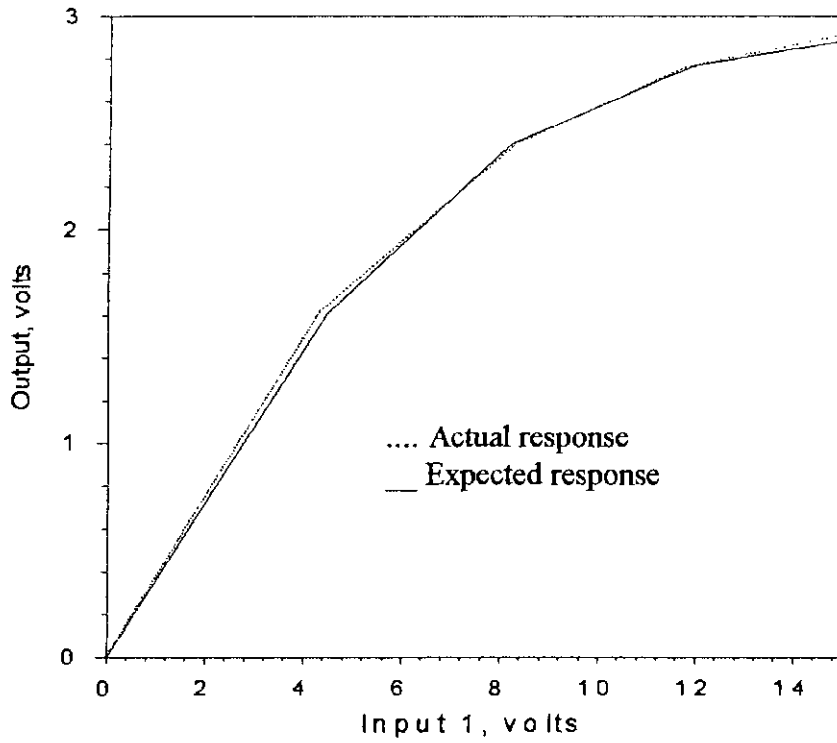


Figure 4.4.a Response of node 4 with input 2 grounded

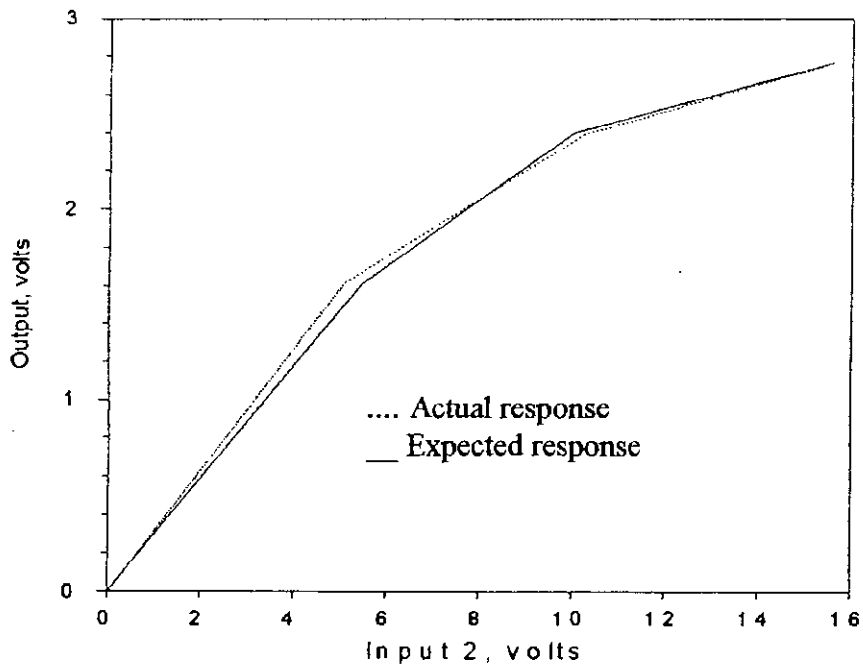


Figure 4.4.b Response of node 4 with input 1 grounded

4.2.3 Testing on Neural Network

After the satisfactory operation of the input signal conditioning units and the hidden layers, the two modules were joined together to form the neural network for the reactive power controller. The expected response of this joint module was given in Figure 2.10. It was mentioned in article 3.3.2 and in appendix-7 that an overall amplification of 6 times was introduced at the output of the neural network for better operation during convergence of the network. So expected data of Figure 2.10 needs to be multiplied six times to compare them with the actual output of the network. To verify the performance of the combined modules, an experimental setup same as Figure 4.1 was chosen and DC voltage output from the neural network was compared for different loading conditions. These readings are compared with the expected output of the neural network. The comparison is shown in tabular form in Table 4.2 and in graphical form in Figure 4.5. Close results are obtained in every case.

4.2.4 Testing on KVAR Control and Capacitor Switching Unit

The two final output modules, KVAR control and capacitor switching circuits were tested together. The development of these two modules were explained in article 3.3.3 and 3.3.4. Seven step three stage switching options were made during implementation of these two units. The input signal to the KVAR control block comes from the neural network output which produced voltage within the range of -3 volt to 2.8 volt as shown in Figure 4.5. So testing on the KVAR control blocks and capacitor switching circuits was done by simply varying the input voltage within the mentioned range through a variable DC supply. All seven stage switching signals were observed and the triggering voltages for each step was

Table 4.2 Test results in the neural network output

No. of obs.	Load current in Amperes	Power factor angle in degree	DC output of the neural network (volts)	Expected output of neural network (volts)
1	2.00	37	-3.00	-2.84
2	2.78	53	-1.55	-1.51
3	2.87	54	-1.4	-1.376
4	3.53	54	-0.8	-0.678
5	3.38	36	-2.0	-1.815
6	3.93	44	-0.9	-0.84
7	4.12	46	-0.7	-0.535
8	4.87	55	0.85	0.8
9	4.21	24.5	-2.00	-2.13
10	4.45	30	-1.5	-1.5
11	5.18	43	0	0.209
12	5.14	16	-2.7	-2.56
13	5.55	27	-1.10	-1.11
14	6.41	20	-1.6	-1.545
15	6.8	27	-0.373	-0.4

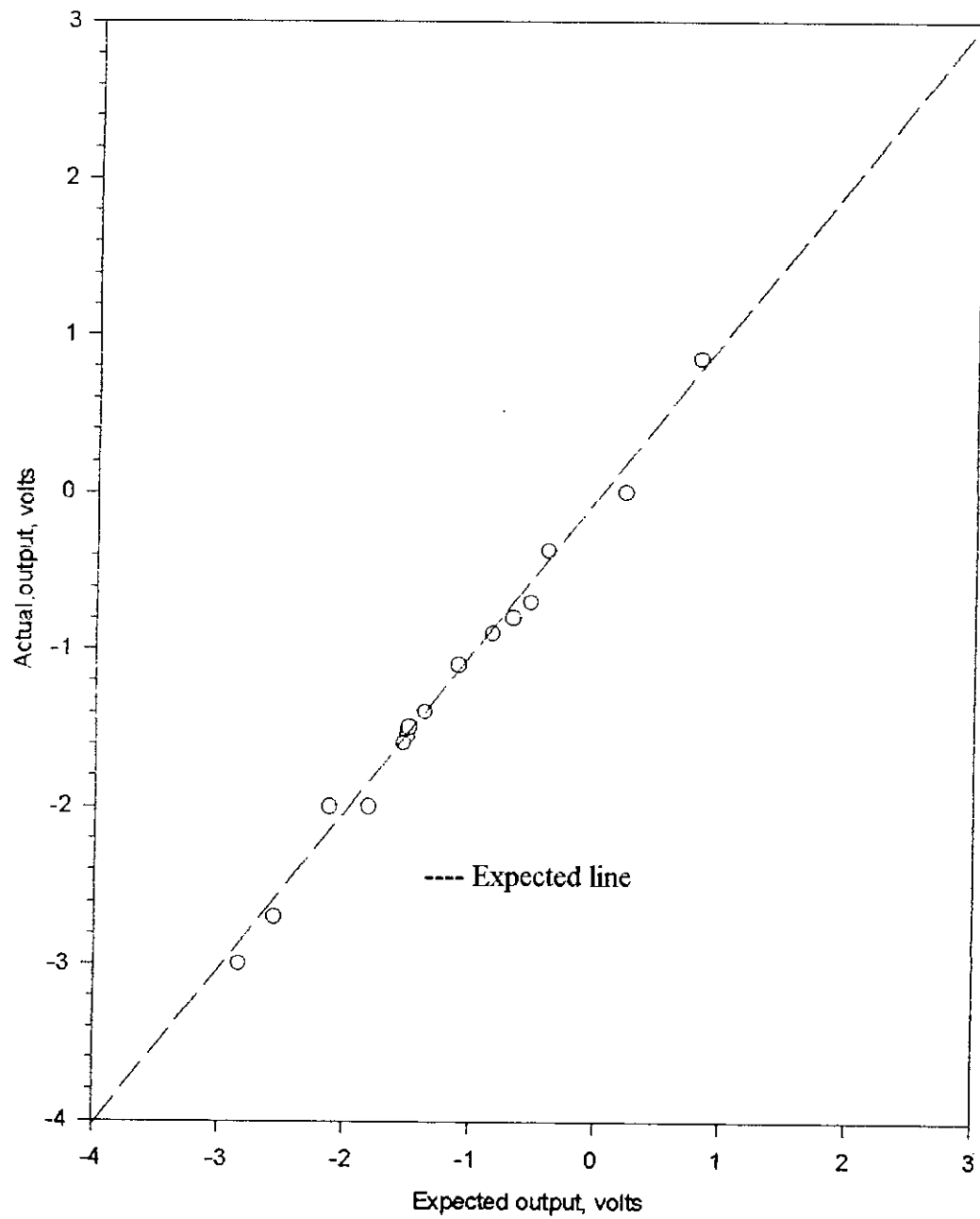


Figure 4.5 Comparison between expected and actual output

observed and calculated through an oscilloscope. The trigger voltages for all seven stages are mentioned in Table 4.3.

4.3 PERFORMANCE TEST ON NN BASED RPC

In the previous articles step by step development and testing of all neural network modules were done separately. Finally, all the modules are put together to form the complete NN based RPC. The developed network is put on to physical operation and rigorous laboratory tests were performed on the implemented network. The performance of the NN based RPC is compared with the conventional microprocessor based RPC. The following articles will highlight on this final experiment and the observed test results.

4.3.1 Experimental Setup

The experimental setup for the tests to be done on the neural network is shown in figure 4.6. The figure shows that the NN controller and μ P controller are operating in tandem. The μ P controller senses the overall line current through a 2:1 C.T. while the NN controller senses only the load current through its power resistor. The load is made by paralleling a rheostat bank having nine 200 ohms 1 kW rheostats with two inductance banks. One inductance bank has a rating of 240 volts, 0.6 KVAR, 10×0.25 ampere inductor and the other one has 240 volts, 2.4 KVAR, 10×1.0 ampere inductor. The three stage capacitor banks have stepping in the order of 1:2:3 and each steps had capacitors of $15 \mu\text{f}$. The supply voltage was 3 phase, 220 volts line to neutral. The load current for both μ P and NN controller is sensed in single phase because the three phase power system is considered to be balanced. Moreover, there is no decisive definition of power factor for unbalanced three phase system.

Table 4.3 Triggering voltage for different capacitor switching

Step No.	Binary Equivalence	DC Trigger voltage	KVAR
1	001	-3.385	0.173
2	010	-2.35	0.346
3	011	-1.324	0.52
4	100	-0.2928	0.693
5	101	0.738	0.866
6	110	1.769	1.0392
7	111	2.8	1.213

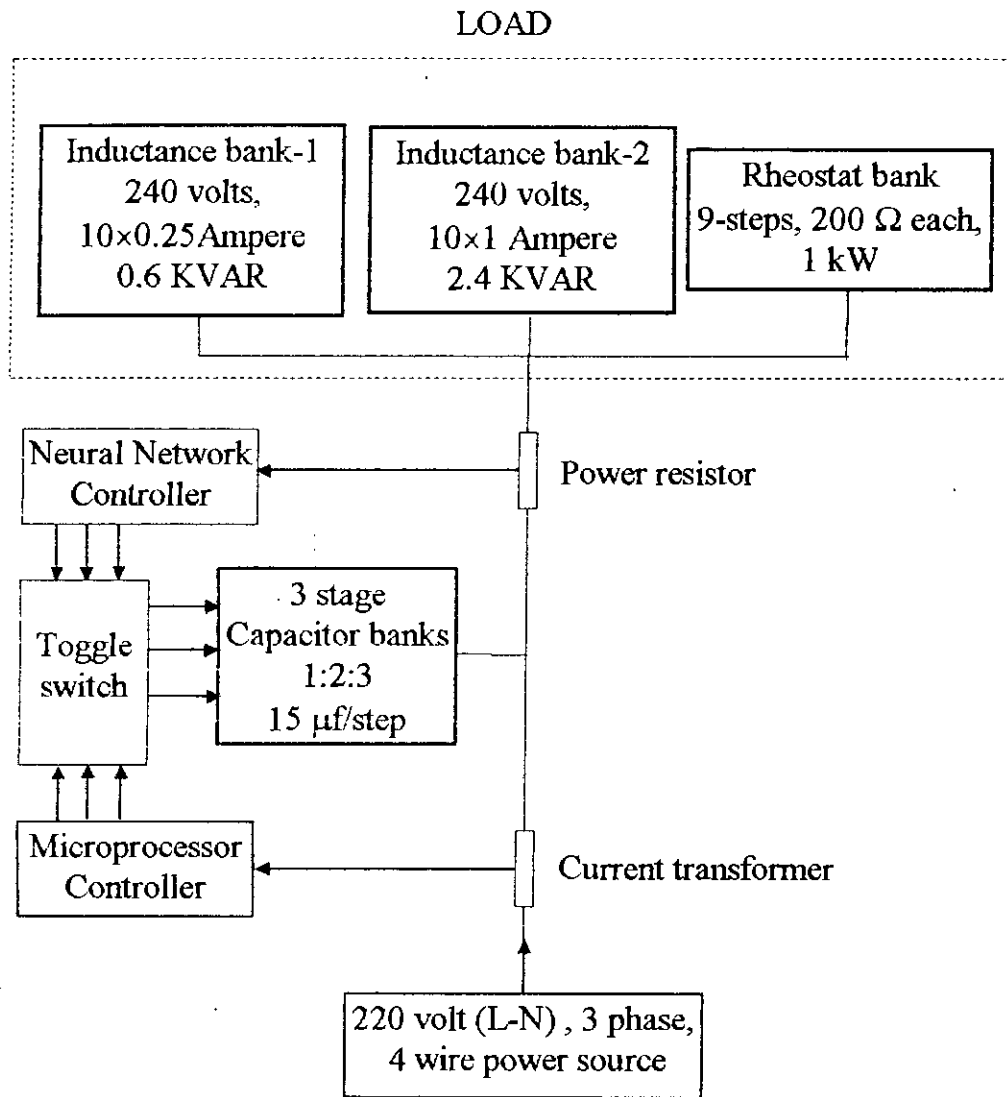


Figure 4.6 Experimental setup for performance test of the NN based RPC

The performance of the NN based RPC will be compared with the performance of the conventionally used μ P based RPC. The operation of μ P controller has been elaborately described in Appendix-1. The μ P controller operates in digital mode and senses current through C.T. It calculates the KVAR demand from voltage and overall line current. The μ P performs hunting by switching capacitors one by one and senses whether the power factor has improved. It stops hunting when the power factor enter the desired range or the highest switching step has been reached. So, the operation is a closed loop system and switching is done through hunting process.

The capacitor switching stage for both of the μ P and NN controller are built identically. A clear description with figure of the capacitor switching stage is given in article 3.3.4. As the DC relays of the capacitor switching stage are activated it connects one of the power line to the magnetic relays of the PFI plant. These magnetic relays then make contacts between the 3 phase capacitor banks with the power line. A toggle switch shown in figure 4.6 is used to toggle the power line between the μ P and NN controller. When the switch connects the power line with the DC relays of the NN controller then it is the NN controller which activates the capacitor banks. On the other hand, when the switch connects the power line with the DC relays of the μ P controller then it is the μ P controller which activates the capacitor banks. This technique is introduced so that both controllers can operate in tandem and comparison between them can be made.

4.3.2 Experimental Data

Different combinations of loading were presented to the controller circuit by varying the combination of resistance and inductance. It was kept in consideration

that the neural network had the capacity to handle currents from 2 amperes to 7 amperes and the power factor should not be less than 0.5. The sets of data taken during the experiment is given in table 4.4 and a comparative study of capacitor switching is given in figure 4.7.

4.4 RESULTS

Data obtained from the experiments clearly shows a satisfactory performance of the NN controller as in each and every case the power factor was improved towards unity. The switching time of the NN controller was also observed to be faster than the μ P controller. The loads were varied at random. No preset data were prepared to match the capacitor stages. The load KVAR to be compensated, in each load combinations, was decided by the available inductive and resistive banks. Of course, the discrete change of inductive and resistive currents were made by combination of available load switches. The minimum inductive current to be adjusted was limited to 0.25A and the resistive current to 2.2A. It is interesting to note that the connected capacitor banks were put in such a combination that the load power factor is properly compensated. The maximum deviation was noted to be 0.98 pf which is within permissible range.

Although for discrete control of power factor, increase in number of stages will make the control smoother, in this particular case it was unnecessary. Moreover, an increase in number of stages would cause a frequent ON and OFF of the magnetic contactors for the capacitor banks. This is not desirable. In this setup, frequent operation of magnetic contactors were not observed.

Table 4.4 Data obtained from performance testing on NN based RPC

Line current	Uncompensated power factor	Load KVAR	KVAR compensated	Compensated Power factor by NNC	Compensated Power factor by μ PC
2.00	0.79	0.269	0.228	0.99; (001)	0.99; (001)
2.78	0.61	0.484	0.456	1.0; (010)	1.0; (010)
2.87	0.58	0.514	0.456	0.99; (010)	0.99; (010)
3.53	0.5	0.672	0.684	1.0; (011)	1.0; (011)
3.38	0.8	0.446	0.456	0.99; (010)	0.99; (010)
3.93	0.72	0.6	0.684	0.99; (011)	0.99; (011)
4.12	0.69	0.656	0.684	1.0; (011)	1.0; (011)
4.87	0.57	0.88	0.912	0.99; (101)	0.99; (101)
4.21	0.91	0.384	0.456	1.0; (010)	1.0; (010)
4.45	0.86	0.499	0.456	1.0; (010)	1.0; (010)
5.18	0.73	0.779	0.684	0.99; (100)	0.99; (100)
5.14	0.96	0.316	0.228	1.0; (001)	1.0; (001)
5.55	0.89	0.556	0.684	1.0; (100)	1.0; (100)
6.41	0.94	0.481	0.456	0.98 (010)	0.98 (010)
6.8	0.89	0.682	0.684	0.99; (011)	0.99; (011)

* NOTE: Figures in the brackets of the last two columns indicate the switching states of the three stages of the capacitor bank.

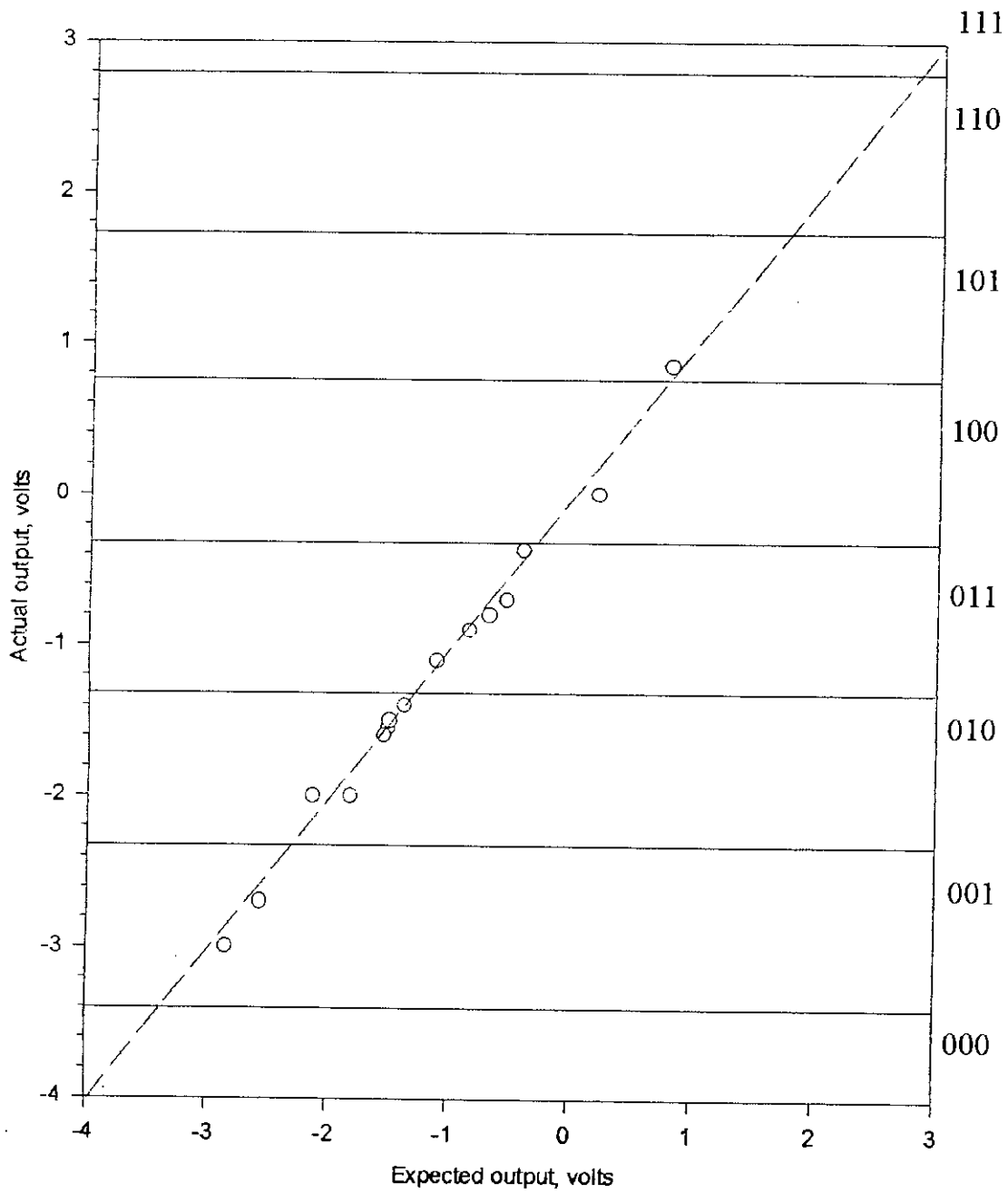


Figure 4.7 The overall response of the NN based RPC.

It should be noted that the maximum combination of '111' for full compensation could not be tested with the available load facility. But the controller for the combination was tested indirectly. The DC voltage corresponding to current and that to power factor were applied at the input nodes of the NN so that a situation of '111' output was achieved. The output for the case was found to be perfectly all right and all of the switches operated. Hence, results decisively conclude that the implemented NN based RPC worked satisfactorily.

4.5 COST ANALYSIS

The controller circuit for the NN based RPC developed in the laboratory costs around Tk. 2000.00 only as the electronic circuit was developed mainly with resistances, operational amplifiers and some simple relay circuits. Comparing this cost with the conventionally used microprocessor based RPC it is found that only the microprocessor used in the controller circuit costs over Tk. 15000.00 in the local market. It is true that the conventional RPC gives more options like economic switching facilities and harmonics indications than the developed NN based RPC. But these options can be included in the NN based RPC which has been suggested in the future works described in section 5.2. Even if these extra facilities are included, yet it is expected that the cost required for the NN based RPC will be half of the cost involved in developing microprocessor based RPC. The two main reasons for such conclusion is 1. The NN based RPC do not require sophisticated environment for implementation and 2. The components required for developing NN based RPC is simple and discrete fundamental electronic tools; i.e. no specially designed electronic tool is required.

CHAPTER FIVE
CONCLUSION

5.1 CONCLUSION

A new technology for improving power factor in an industry has been proposed and the fast emerging control algorithm of neural network (NN) architecture is used for this purpose. The NN based controller circuit is designed with the widely available electronic tools, Moreover, in doing so, piece-wise linear approximation of the smooth sigmoid function has proved quite satisfactory for the research prospect.

First of all, an off-line training of the neural network was performed using calculated data for input and output. When training was complete, theoretical tests were carried out with set of data which were not used in the training phase. Confirming the acceptable accuracy of the theoretical test, the weights and the constants of the sigmoid functions were decided to implement. The implementation of the NN was, then, followed as described in chapter 3. A series of test, on the performance of the electronic circuits were performed rigorously and the performance were found to be satisfactory.

The implemented NN based controller was compared with the conventionally used microprocessor (μ P) based controller in a power factor improvement plant. Test results showed the superiority of the NN based controller over the μ P based controller in speed. Due to its analog nature, it was obvious that a NN based controller would act faster than the μ P based controller. Moreover, a μ P based controller hunts for better combination of capacitor switching and a sequence of switching stages occurs before it finally settles down. But NN based controller already knows through the training phase the appropriate switching stage for each KVAR demand and hunting, therefore, is not required.

NN based reactive power controller (RPC) reduces the implementation cost compared to the μ P based RPC. As the NN based RPC was developed with simple electronic circuitry, its manufacturing process is easier and no special environment is required compared to the procedure required for μ P fabrication.

5.2 FURTHER WORK

As the technology described in designing and implementing the NN based RPC is quite new, there are opportunities of further works to be done in this area. Some of such fields demanding special attentions are described below:

1. The implemented NN used feed-forward training. NN based RPC works on an open loop system. That is, it does not sense the overall line current but works only on sensing the load current of the system. So, the implemented NN does not have the error correction capability. This limitation may be overcome if closed loop training is introduced to the NN controller.
2. It was shown in the thesis work that though piece-wise linear sigmoid function is easy to implement than the smooth sigmoid, yet the time for convergence during learning is very long. Smooth sigmoid function is possible to implement with operational transconductance amplifier (OTA). So, NN based RPC will be less time consuming in training cycles if OTA based system is developed.
3. RPC works on discrete mode. But in the present thesis work the NN was learnt with such accuracy that NN converges for all combinations of $l.\sin\theta$. So, the present NN had only one output. The NN module can be made more effective by introducing the capacitor switching signals to the output of the NN. This future NN

module will have outputs equal to the capacitor switching stages and there will be a threshold voltage deciding when which capacitor stage should stay on or off. So, NN convergence for infinite combinations of $I \cdot \sin\theta$ will not be required. This will reduce training time and make NN implementation easier.

4. The present NN model was unable to converge for very low KVAR demand. Learning was ceased in the very low KVAR demand for better accuracy. But by introducing more than six hidden layers and applying more training cycles this limitation can be eliminated. This approach will also reduce the clumsy logic block developed for bypassing low KVAR demand in the present NN module.

REFERENCES

- [1] T. Fukuda and T. Shibata, "*Research trends in neuromorphic control*", J. Robotics Mechatron., Vol. 2, No. 4, 1991, pp. 4-18.
- [2] T. Kohonen, *Self-organization and Associative Memory*, Berlin: Springer-Verlag, 1984.
- [3] B. Widrow and M. A. Lehr, "*Thirty years of adaptive neural networks: Perceptron, Madaline, and back-propagation*", Proc. IEEE, Vol. 78, No. 9, Sept. 1990, pp. 1415-1441.
- [4] R. P. Lippmann, *An introduction to computing with neural nets*", IEEE, ASSP Mag., Vol. 4, April 1987, pp. 4-22.
- [5] T. Poggio and F. Girsi, "*Networks for approximation and learning*", Proc. IEEE, Vol. 78, No. 9, Sept. 1990, pp. 1481-1497.
- [6] S. Satyanarayana, Y. Tsividis and P. Graf, "*A reconfigurable VLSI neural network*", IEEE J., Solid-State Circuits, Vol. 25, June 1990, pp. 849-855.
- [7] C. E. Cox and W. Ekkahard Blanz, "*Ganglon—A fast hardware implementation of a connectionist classifier*", IEEE-CICC, Phoenix, AZ, 1991.
- [8] Michel Verleysen, Bruno Sirletti, Andre Vandemeulebroecke, and Paul G. A. Jespers, "*A High-Storage Capacity Content-Addressable Memory (CAM) and Its Learning Algorithm*", IEEE Trans. Circuits and Systems, Vol. 36, No. 5, May 1989, pp. 762-765.
- [9] Joydeep Ghosh, Patrick Lacour, and Spence Jackson, "*OTA-Based Neural Network Architectures with On-Chip Tuning of Synapses*", IEEE Trans. Circuits and Systems, Vol. 41, No. 1, Jan 1994, pp. 49-57.

- [10] Russel D. Reed, Randell L. Geiger, "A Multiple Input OTA Circuit for Neural Networks", IEEE Trans. Circuits and Systems, Vol. 36, No. 5, May 1989, pp. 767-769.
- [10] B. Feher, "Resonator based digital filters using field programmable gate array elements", Fifth Ann. IEEE Int. ASIC Conf. Rochester, NY, Sept. 1992.
- [11] Nazeih M. Botros, and M. Abdul-Aziz, "Hardware Implementation of an Artificial Neural Network Using Field Programmable Gate Arrays (FPGA's)", IEEE Trans. Industrial Electronics, Vol. 41, No. 6 Dec. 1994, pp. 665-667.
- [12] M. A. Mahowald and C. Mead, "Silicon retina", in Carver Mead, *Analog VLSI and Neural Systems*, Reading, MA: Addison-Wesley, 1989, Chap. 15.
- [13] R. F. Lyon and C. Mead, "Electronic Cochlea", *ibid.*, chap. 16.
- [14] J. P. Sage, K. Thompson and R. S. Withers, "An artificial neural network integrated circuit based on MNOS / CCD principles", Am. Inst. of Phys. Conf. Proc. 151, 1986, pp. 381-385.
- [15] P. W. Hollis and J. J. Paulos, "Neural network using analog multipliers", Proc. IEEE Int. Conf. Circuits Syst. May 1988, pp. 499-502.
- [16] H. P. Graf, L. D. Jackel and W. E. Hubbard, "VLSI implementation of a neural network model", *Computer*, March 1988, pp. 41-49.
- [17] A. P. Thakoor, A. Moopenn, J. Lamb and S. K. Khanna, "Electronic hardware implementation of neural networks", *Applied Optics*, No. 26, Dec. 1987, pp. 5085-5092.
- [18] Y. Owechko and B. H. Soffer, "Programmable multi-layer optical neural networks with asymmetric interconnection weights", Proc. IEEE Int. Conf. on Neural Networks, 1988, pp. 385-393.

- [19] T. Inoue, F. Ueno and S. Sonobe, "*Switched capacitor building blocks for fuzzy logic and neural networks*", Trans. IEICE, 71, Dec. 1988, pp. 1259-1260.
- [20] J. E. Hansen, J. K. Skelton, and D. J. Allstot, "*A time-multiplexed switched capacitor circuit for neural network applications*", Proc. IEEE Int. Symp. Circuits Syst., 1989, pp. 2177-2180.
- [21] Y. Tsvividis and S. Satyanarayana, "*Analogue circuits for variable synapse electronic neural networks*", Electron lett., 23, Nov. 1987, pp. 1313-1314.
- [22] N. Morgan (Ed.), *Artificial neural networks—electronic implementations*, Los Alamitos, CA: IEEE Computer society Press, 1990.
- [23] ANZA/DP Plus High-Speed Neurocomputing Coprocessor, HNC, Inc., San Diego, CA.
- [24] Delta II Floating Point Processor, Science Applications International Corp., (SAIC), San Diego, CA.
- [25] Neural Emulation Tool (NET) Processor Board, Ford Aerospace, Houston, TX.
- [26] R. Mandal, S. K. Basu, A. Kar, and S. P. Chowdhury, "*A Microprocessor Based Power Factor Controller*", IEEE Trans. Industrial Electronics, Vol. 41, No. 3, June 1994, pp. 361-371.
- [27] H. M. El-Bolok, M. E. Masoud, and M. M. Mahmoud, "*A Microprocessor-Based Power Factor Corrector for Nonlinear Loads*", IEEE Trans. on Industrial Electronics, Vol. 37, No. 1, Feb. 1990, pp. 77-81.
- [28] G. Shepherd, "*The neuron doctrine: A revision of functional concepts*", Yale J. Biol. Med., Vol. 45, No. 6, Dec. 1972, pp. 584-599.

- [29] R. Shapley and C. Enroth Cugell, "*Visual adaptation and retinal gain controls*", in *Progress in Retinal Research*, N. N. Osborne and G. J. Chader, Eds., Vol. 3, Oxford, England: Pergamon, 1984, pp. 263.
- [30] C. Mead, *Analog VLSI and Neural Systems*, Reading, MA: Addison-Wesley, Co., 1989.
- [31] F. Rosenblatt, "*The perceptron: A probabilistic model for information storage and organization in the brain*", *Psychological Rev.*, vol. 65, pp. 368-408, 1958.
- [32] John J. Hopfield and David W. Tank, "*Collective computation in neuron like circuits*", *Scientific American*, 257(6), 1987, pp. 104-114.

APPENDICES

APPENDIX-1

Among the various microprocessor controller available, the two widely used relays type RM 9606 manufactured by FRAKO, Kondensatoren, D-79331, Teningnen and BLR-MC manufactured by BELUK GMBH, D-8000, Munich 45, Germany has been taken as an example. Figure A.1.1 shows the attractive features of this type of controller.

A.1.1 Design and Mode of Operation

The RM 9606 of FRAKO and MC series of BELUK microprocessor controlled power factor control relays are the latest addition of the long standing pedigree of the reactive controllers. These relays are the result of many years experience in this specialized field of power factor measurement and control technology. The electronic measuring circuit has been tried and tested over many years and now. With the introduction of microprocessor technology, additional features are provided such as digital indication of the system power factor and the number of switching steps in circuit. These two values are given out via separate outputs, so that a printer or chart recorder may be connected.

The measurement system measures all four quadrant of the wave form and is consequently independent of system harmonics. This means that even when active power is fed back onto the main bus, the control relay ensures compensation for the reactive power which has been drawn from the main. This current-time integrated measurement circuit has been proven over many years and is particularly necessary on installations having thyristor-controlled machines.

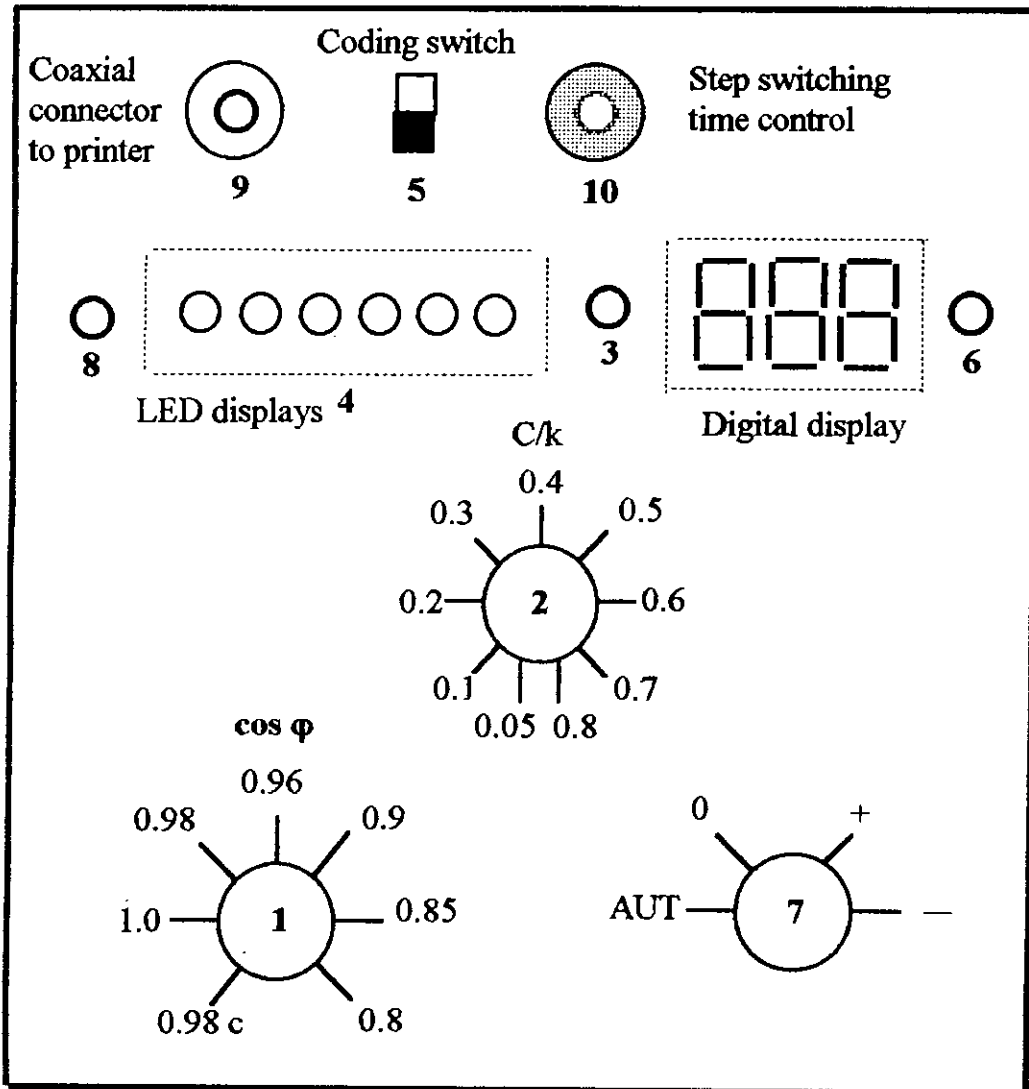


Figure A.1.1 Main features of RM 9606 type microprocessor based RPC controller.

The analogue signal produced by the measuring device is digitized and processed in a microprocessor. All switching program known at this time are made possible by means of this processor. Its output signals control miniature relays via drive circuits which actuate the capacitor switching contactors to control the power capacitors. The diverse features of this family of controllers, type BLR-MC, is given in Table A.1. The only difference within the MC range relates to the number of output relays. BLR-MC 03 has 3 whilst BLR-MC 14 has 14 output relays. A total of six different versions are available : MC 03, MC 06, MC 09, MC 12 and MC 14. The switching programs can be selected by means of small DIP switches located behind the removable nameplate on the front of the relay. The possible programming are listed in Table A.1.

The digital indicator will show the number of steps switched on the 'S' or 'K' program. The 'S' program actuates the capacitor sequentially, e.g. from 1 up to 6 and go back from 6 down to 1 or 0. The 'K' programs always actuate and deactivate steps in the same direction of rotation; 1-2-3-4, "down": 1-2-3, "up": 5-6-1-2, "down" : 4-5-6-1-2 etc. The 'K' programs thus distribute operating hours evenly over all capacitors and also permit faster regulation. Faster switching is possible because, once a capacitor stage is deactivated, it has enough time to discharge before it is reactivated. Even in a limit situation, when all steps of a capacitor bank are switched in and, because of the load, one stage has just been deactivated and must be reactivated immediately afterwards, the BLR-MC relay takes account of this independently and delays the reactivating time accordingly. The well-proven and problem-free matching to system conditions by means of $\cos\phi$ and C/k adjustable settings remains unaltered in spite of microprocessor technology.

Type BLR-	D I P					Program	Switching Sequence	Number of	
	1	2	3	4	5			Capacitors	Stages
MC 03	1	0	0	0	0	S 3	1:1:1	3	3
MC 03	1	0	0	0	1	S 35	1:2:2	3	5
MC 03	1	1	1	1	0	S 36	1:2:3	3	6
MC 03	1	1	0	1	0	S 37	1:2:4	3	7
MC 06	0	1	0	0	0	S 6	1:1: :1	6	6
MC 06	0	1	0	0	1	S 611	1:2: :2	6	11
MC 06	0	1	1	0	0	K 6	1:1: :1	6	6
MC 06	0	1	1	0	1	K 611	1:2: :2	6	11
MC 06	0	0	1	1	0	S 415	1:2:4:8	4	15
MC 06	1	0	1	1	0	S 515	1:2:4:4:4	5	15
MC 06	0	1	1	1	0	S 619	1:2:4: :4	6	19
MC 08	1	1	1	0	0	K 8	1:1: :1	8	8
MC 08	1	1	1	0	1	K 815	1:2: :2	8	15
MC 09	1	1	0	0	0	S 9	1:1: :1	9	9
MC 09	1	1	0	0	1	S 917	1:2: :2	9	17
MC 12	0	0	1	0	0	S 12	1:1: :1	12	12
MC 12	0	0	1	0	1	S 1223	1:2: :2	12	23
MC 12	0	0	0	1	0	K 10	1:1: :1	10	10
MC 12	0	0	0	1	1	K 1019	1:2: :2	10	19
MC 12	1	0	0	1	0	K 12	1:1: :1	12	12
MC 12	1	0	0	1	1	K 1223	1:2: :2	12	23
MC 14	1	0	1	0	0	S 14	1:1: :1	14	14
MC 14	1	0	1	0	1	S 1427	1:2: :2	14	27
MC 14	0	1	0	1	0	K 14	1:1: :1	14	14
MC 14	0	1	0	1	1	K 1427	1:2: :2	14	27

Table A.1 Programming features of BLR-MC.

A.1.2 Cos ϕ Adjustment

Adjustment of the target power factor by rotating the knob (1) results in a shift of the C/k strip around the zero point of the co-ordinate system of the real power axis P and power factor control axis Q. The range of shift extends from $\cos\phi = 0.80$ inductive through 1 to 0.98 capacitive.

It is also possible to have an adjustment range of 0.80 inductive through 1 to 0.95 capacitive as a special model. In this case, however, the voltage supply changes to L1-N (model "e" = single phase, Appendix-D5).

A.1.3 C/k Threshold Settings

The function of the capacitor control relay is to switch capacitor switch in or out according to the reactive load. Capacitor steps value 'C' is therefore an important variable which must be known, along with the current transformer ratio 'k'. The most common values of C/k at 400 V are given on the table which is adhered to the relay. The calculated value of C/k is set by rotating knob (2).

In case of capacitor banks which has different values of 'C' (e.g. 1:2:2:2...2), the setting is always calculated by taking the smallest step : value '1' for 'C'.

The power factor control relays have stepless adjustment of the C/k range between $0.05 A_r$ and $0.80 A_r$. These values are the reactive threshold activating currents on the relay. If the reactive current content of the load exceeds the set C/k value, LED (3) extinguishes. The control relay either begins to activate (+) or

deactivate (-) steps. For different supply voltages, the C/k settings can be calculated from the following formula :

$$C/k = 0.66 \times \frac{P_c}{\sqrt{3} \cdot V \cdot k} \quad \dots \quad \dots \quad \dots \quad (A.1)$$

Where :

P_c = capacitor step power (kvar)

V = supply voltage in kV (phase-phase voltage on three phase system)

k = transformer ratio of the current transformer, e.g. 1000 A/ 5 A = 200.

The factor 0.66 or 66% is fixed so that e.g. a 10 kvar capacitor will only be actuated when 6.6 kvar inductive reactive power is exceeded. The range can be varied between 60 and 90% if necessary.

A.1.4 No-volt Release

If the power supply is briefly interrupted for more than approx. 35 msec. the MC relay immediately switches out all capacitors. When the main supply voltages is restored, the control procedures starts after a lock-out time of approx. 90 sec. during which time LED (3) flashes. This lock-out time also applies to initial operation i.e. whenever the mains supply voltage is reapplied to the control relay.

A.1.5 Display of Activated Capacitors

When DIP switch 6, behind the removable nameplate lid, is switched off, the number of activated capacitor stages and the power factor are digitally displayed alternately. The additional expense of the supplementary LED's (4) was considered essential, so as to show which exit relays are closed. This is particularly important with the economy switching program.

A.1.6 Digital $\cos\phi$ Display

A standard feature of the microprocessor controlled relay is a constant digital display of the power factor $\cos\phi$ when DIP switch 6 is in the "on" position. This display is independent of the wave-form of current and voltage. In this way, particularly for thyristor controlled systems, so-called "distorted reactive power" is incorporated into the power factor display. When DIP switch 6 is switched off, the digital display alternates between system power factor and number of switching steps accomplished, and, at the same time, digital indicator LED (6) comes on to indicate that switching step 6 has been reached.

Display cycle : **15 sec. $\cos\phi$,**
 3 sec. stage indication.

A.1.7 Hand/Auto Change Over Switch

Rotary switch (7) enables the relay to operate in the Hand, Auto or Hold mode. The switch is rotated with a screw driver and provides the following options.

- **AUT** = Relay operates automatically.
- **0** = Relay holds the switch stage it has reached. LED (3) flashes. Fault signal may be triggered.
- **+** = Relay switches capacitors in, according to the selected switching program. LED (3) flashes.
- **-** = Relay switches capacitors out, according to the selected switching program. LED (3) flashes.

The selected switching program is retained in any position of switch (7). In position 0 switching up or down is terminated. In position AUT, the relay switches again automatically according to the measured reactive load.

A.1.8 Switching Times

Switching time is adjusted by means of a stepless potentiometer, located behind the removable name-plate. Step switching time can be reduced down to 5 sec. per step, when commissioning or testing. The longest switching time is 70 sec. per step.

A.1.9 Fault Signaling Device

A fault signaling device has been incorporated in the BLR-MC relay. LED (8) will come on if the target power factor has not been reached, if insufficient capacitance has been installed for example. DIP switch 7 will deactivate the fault signaling device if it is not required. A triggered fault signal indicated by the LED (8) can be acknowledged or canceled by briefly turning the rotary switch (7) to "0".

A.1.10 Fault Signaling Contact (m) for External Signaling

A built in signaling relay (m) can provide a valuable extra check of the system by monitoring the function of the compensation equipment and the control relay. Over or under compensation related to the selected target power factor closes a relay contact after 50 times the switching time between steps. An audible or visual alarm can be activated. If an external power supply is available, failure of the measuring voltage will also be signaling by this contact. This feature makes it possible to localize and rectify faults such as insufficient installed capacitance or contactor or fuse failure as soon as they occur, rather than waiting until excessive reactive current costs appear on the electricity bill. Figure A.12 shows the circuit diagram of the relay connection with the power line.

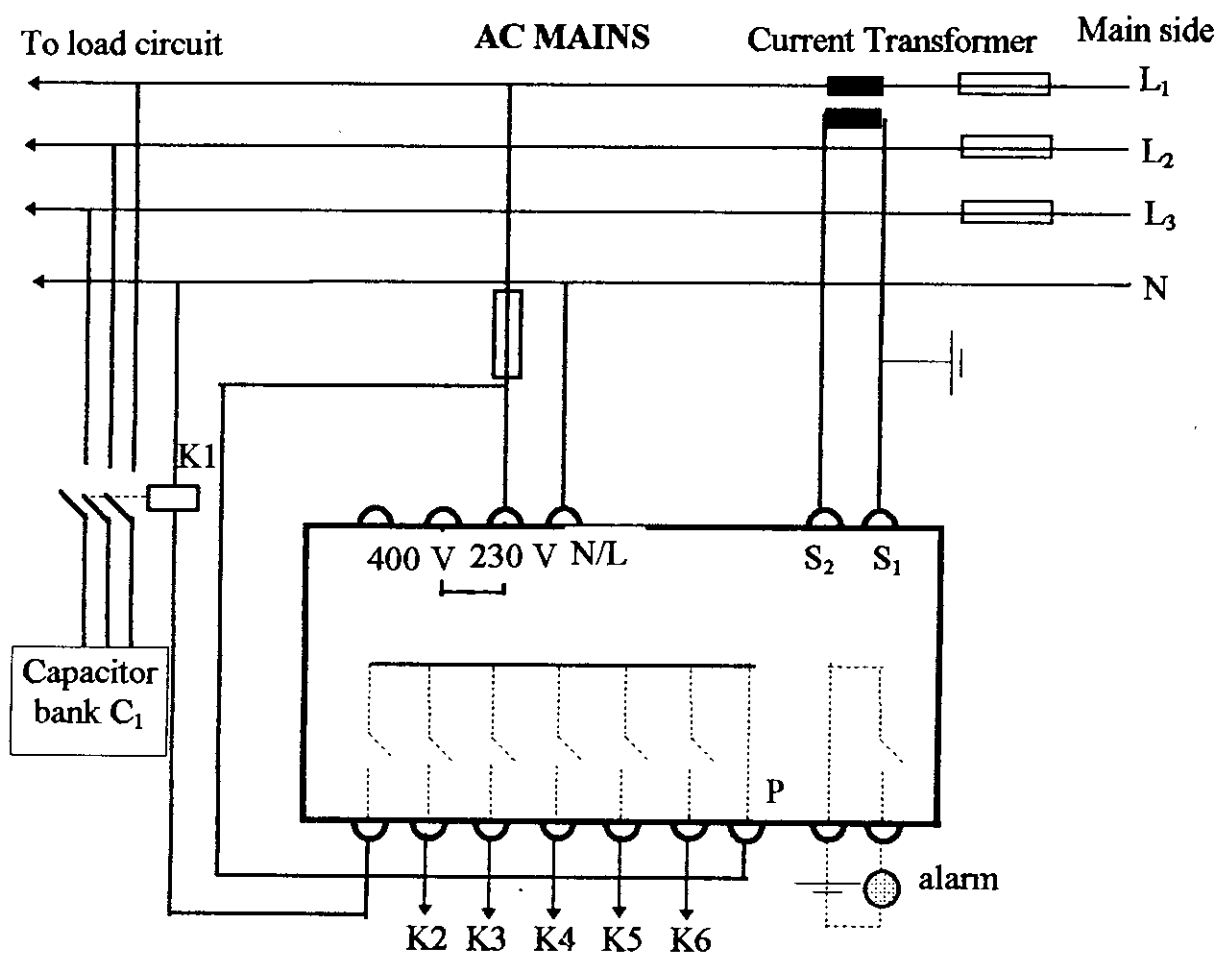


Figure A.1.2. Circuit diagram of RM 9606 connected with the bus line

APPENDIX-2

```
* LEARN.FOR
* PROGRAM FOR NEURAL NET SOLUTION WITH SIGMOID FUNCTION

DATA NP,NT,NI,NJ,NK /64,500,2,4,1/

PRINT *, 'DO YOU WANT TO START FROM PREVIOUS VALUES ? (1/0)'
READ (*,*) NOYES
PRINT *, 'NO OF ITERATION ? '
READ(*,*) NT

CALL ALRN(NP,NT,NI,NJ,NK,NOYES)

STOP
END

SUBROUTINE ALRN(NP,NT,NI,NJ,NK,NOYES)
REAL NNW,NNW0
INTEGER ITER
REAL OI(2,600),OJ(4),OK(1),WJ(4,2),WK(1,4)
REAL DWK(1,4),DELJ(4),DELK(1),THJ(4),THK(1)
REAL DTHJ(4),DTHK(1),NETJ(4),NETK(1),DWJ(4,2)
REAL TK(1,600)

DATA ALP,NNW,THO1,THO2,EPS/0.0,0.5,1.0,1.0,1E-10/
OPEN(UNIT=1,FILE='C:\FORTRAN\NEURAL\VAR.DAT')
OPEN(UNIT=2,FILE='C:\FORTRAN\NEURAL\WT.DAT')
OPEN(UNIT=3,FILE='C:\FORTRAN\NEURAL\OUT.DAT')

IP=1

DO 190 CUR=2.0,7,1.0
DO 190 TH=0,52,4.0
    OI(2,IP)=CUR/9.6
    OI(1,IP)=TH/98.7
*****
    X=(OI(2,IP)*.25)+OI(1,IP)-.2675
    Y=(OI(2,IP)*1)+OI(1,IP)-.53
    IF((X.GT.0).AND.(Y.GT.0))THEN
*****
        TK(1,IP)=(CUR*SIN(TH/57.3)-2.4)/5.0
```

```

        VAR=220*CUR*SIN(TH/57.3)
        WRITE(1,*)VAR

        IP=IP+1
    ENDIF
190  CONTINUE
        NP=IP-1
    DO 333 IP=1,NP
333  TK(1,IP)=TK(1,IP)-.2
        WRITE(*,*) 'NUMBER OF PATTERN = ',NP
        IF(NOYES.EQ.1) THEN
            READ(2,*) ITP
            READ(2,*) ((WJ(J,I),I=1,NI),J=1,NJ)
            READ(2,*) (THJ(J),J=1,NJ)
            READ(2,*) ((WK(K,J),J=1,NJ),K=1,NK)
            READ(2,*) (THK(K),K=1,NK)
            READ(2,*) NNW
        ENDIF
        IF (NOYES.EQ.0) THEN
            ITP=0
        DO 232 I=1,NJ
232  THJ(I)=0.0
        DO 233 I=1,NK
233  THK(I)=0.0
            WJ(1,1)=.1
            WJ(1,2)=-.1
            WJ(2,1)=-.234
            WJ(2,2)=0.34
            WJ(3,1)=.21
            WJ(3,2)=-0.3314
            WJ(4,1)=.1
            WJ(4,2)=-.091
            WK(1,1)=.132
            WK(1,2)=-.11
            WK(1,3)=.23
            WK(1,4)=-0.5
        ENDIF
        DO 2 I=1,NJ
            DTHJ(I)=0.0
        DO 2 J=1,NI
            DWJ(I,J)=0.0
2  CONTINUE
        DO 3 I=1,NK

```

```

        DTHK(I)=0.0
DO 3 J=1,NJ
        DWK(I,J)=0.0
3 CONTINUE
        ITER=0
        IF(NOYES.EQ.1) THEN
        ITER=ITP
        NT=NT+ITP
        ENDIF

        NNW=NNW
DO 210 L=1,NT
        ERR=0.0
DO 10 IP=1 ,NP
11 DO 40 IJ=1,NJ
        NETJ(IJ)=0.0
DO 50 II=1,NI
        NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
        IF (NETJ(IJ).GT.10292) PRINT *, NETJ(IJ)
50 CONTINUE

        SS=EXP(-(NETJ(IJ)-THJ(IJ))/THO1)
        OJ(IJ)=(1.0-SS)/(1.0+SS)

40 CONTINUE
DO 60 IK=1,NK
        NETK(IK)=0.0
DO 70 IJ=1,NJ
        NETK(IK)=NETK(IK)+WK(IK,IJ)*OJ(IJ)
        IF (NETK(IK).GT.10292) PRINT *, NETK(IK)
70 CONTINUE
        OK(IK)=(NETK(IK)-THK(IK))/THO2
60 CONTINUE
DO 80 IK=1,NK
        DELK(IK)=(TK(IK,IP)-OK(IK))*(1./THO2)
DO 90 IJ=1,NJ
        DWK(IK,IJ)=NNW*DELK(IK)*OJ(IJ)+ALP*DWK(IK,IJ)
        WK(IK,IJ)=WK(IK,IJ)+ DWK(IK,IJ)
90 CONTINUE
        THK(IK)=THK(IK)+DTHK(IK)
80 CONTINUE
DO 100 IJ=1,NJ
        SUM=0.0

```

```

DO 200 IK=1,NK
    SUM=SUM+DELK(IK)*WK(IK,IJ)
200 CONTINUE
    DELJ(IJ)=OJ(IJ)*(1.0-OJ(IJ))*SUM

DO 110 II=1,NI
    DWJ(IJ,II)=NNW*DELJ(IJ)*OI(II,IP)+ALP*DWJ(IJ,II)
    WJ(IJ,II)=WJ(IJ,II)+ DWJ(IJ,II)
110 CONTINUE
    THJ(IJ)=THJ(IJ)+DTHJ(IJ)
100 CONTINUE

DO 777 IK=1,NK
    ERR=ERR+0.5*(TK(IK,IP)-OK(IK))**2
777 CONTINUE
10 CONTINUE

    ITER=ITER+1
    WRITE(*,'(I6,2X,3(F12.7,2X))') ITER,NNW,ERR
    IF(ERR.LE.EPS.OR.ITER.EQ.NT) THEN
        CLOSE(2)
        WRITE(*,*) 'STORE THE WTS AND THITAS IN FILE ? (1/0)'
        READ(*,*) NY
        IF(NY.EQ.1) THEN
            OPEN(UNIT=2,FILE='C:\FORTRAN\NEURAL\WT.DAT')
            WRITE(2,*) ITER
            WRITE(2,*) ((WJ(J,I),I=1,NI),J=1,NJ)
            WRITE(2,*) (THJ(J),J=1,NJ)
            WRITE(2,*) ((WK(K,J),J=1,NJ),K=1,NK)
            WRITE(2,*) (THK(K),K=1,NK)
            WRITE(2,*) NNW
        ENDIF
        GOTO 250
    ENDIF
    NNW=.2

210 CONTINUE
250 IF(ERR.LE.EPS) THEN
    PRINT *,'CONVERGED'
ELSE
    PRINT *,'NONCONVERGENT'
ENDIF

```

```

DO 290 IP=1,NP
DO 940 IJ=1,NJ
    NETJ(IJ)=0.0
DO 950 II=1,NI
    NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
950 CONTINUE
    SS=EXP(-(NETJ(IJ)-THJ(IJ))/THO1)
    OJ(IJ)=(1.0-SS)/(1.0+SS)

940 CONTINUE

DO 960 IK=1,NK
    NETK(IK)=0.0
DO 970 IJ=1,NJ
    NETK(IK)=NETK(IK)+WK(IK,IJ)*OJ(IJ)
970 CONTINUE
    OK(IK)=(NETK(IK)-THK(IK))/THO2
960 CONTINUE
    OI1=OI(1,IP)
    OI2=OI(2,IP)
    TK1=TK(1,IP)
    ERROR=TK1-OK(1)
    WRITE(3,'(5(E14.7,2X))')OI1,OI2,TK1,OK(1),ERROR
290 CONTINUE

    CLOSE(2)
    CLOSE(3)
    RETURN
END

```


APPENDIX-3

DATA FILE OUT.DAT SHOWING LEARNING PATTERNS WITH FINAL OUTPUT

<u>INPUT#1</u>	<u>INPUT#2</u>	<u>TARGET</u>	<u>FINAL OUTPUT</u>	<u>ERROR</u>
0.3242148E+00	0.2083333E+00	-0.4680463E+00	-0.4394734E+00	-0.2857292E-01
0.3647417E+00	0.2083333E+00	-0.4449009E+00	-0.4204441E+00	-0.2445683E-01
0.4052685E+00	0.2083333E+00	-0.4229007E+00	-0.4032469E+00	-0.1965380E-01
0.4457954E+00	0.2083333E+00	-0.4021530E+00	-0.3883523E+00	-0.1380065E-01
0.4863222E+00	0.2083333E+00	-0.3827586E+00	-0.3760533E+00	-0.6705344E-02
0.5268490E+00	0.2083333E+00	-0.3648122E+00	-0.3665258E+00	0.1713634E-02
0.2431611E+00	0.3125000E+00	-0.4359750E+00	-0.4258411E+00	-0.1013386E-01
0.2836879E+00	0.3125000E+00	-0.3983362E+00	-0.3908576E+00	-0.7478595E-02
0.3242148E+00	0.3125000E+00	-0.3620694E+00	-0.3565120E+00	-0.5557358E-02
0.3647417E+00	0.3125000E+00	-0.3273513E+00	-0.3236881E+00	-0.3663182E-02
0.4052685E+00	0.3125000E+00	-0.2943511E+00	-0.2929412E+00	-0.1409948E-02
0.4457954E+00	0.3125000E+00	-0.2632294E+00	-0.2646264E+00	0.1396954E-02
0.4863222E+00	0.3125000E+00	-0.2341379E+00	-0.2389749E+00	0.4836991E-02
0.5268490E+00	0.3125000E+00	-0.2072183E+00	-0.2161308E+00	0.8912519E-02
0.2026342E+00	0.4166667E+00	-0.4064033E+00	-0.4052601E+00	-0.1143187E-02
0.2431611E+00	0.4166667E+00	-0.3546332E+00	-0.3547563E+00	0.1230538E-03
0.2836879E+00	0.4166667E+00	-0.3044482E+00	-0.3046199E+00	0.1716912E-03
0.3242148E+00	0.4166667E+00	-0.2560925E+00	-0.2558575E+00	-0.2350509E-03
0.3647417E+00	0.4166667E+00	-0.2098018E+00	-0.2091045E+00	-0.6972402E-03
0.4052685E+00	0.4166667E+00	-0.1658014E+00	-0.1647785E+00	-0.1022965E-02
0.4457954E+00	0.4166667E+00	-0.1243059E+00	-0.1231546E+00	-0.1151234E-02
0.4863222E+00	0.4166667E+00	-0.8551717E-01	-0.8441812E-01	-0.1099050E-02
0.5268490E+00	0.4166667E+00	-0.4962435E-01	-0.4868531E-01	-0.9390414E-03
0.1621074E+00	0.5208333E+00	-0.4043824E+00	-0.4052510E+00	0.8685291E-03
0.2026342E+00	0.5208333E+00	-0.3380041E+00	-0.3395542E+00	0.1550108E-02
0.2431611E+00	0.5208333E+00	-0.2732916E+00	-0.2740688E+00	0.7772148E-03
0.2836879E+00	0.5208333E+00	-0.2105602E+00	-0.2099238E+00	-0.6364286E-03
0.3242148E+00	0.5208333E+00	-0.1501156E+00	-0.1478369E+00	-0.2278715E-02
0.3647417E+00	0.5208333E+00	-0.9225221E-01	-0.8827776E-01	-0.3974453E-02
0.4052685E+00	0.5208333E+00	-0.3725176E-01	-0.3156322E-01	-0.5688533E-02
0.4457954E+00	0.5208333E+00	0.1461768E-01	0.2208853E-01	-0.7470846E-02
0.4863222E+00	0.5208333E+00	0.6310356E-01	0.7253772E-01	-0.9434164E-02
0.5268490E+00	0.5208333E+00	0.1079696E+00	0.1196955E+00	-0.1172596E-01
0.1215805E+00	0.6250000E+00	-0.4305241E+00	-0.4271561E+00	-0.3368020E-02
0.1621074E+00	0.6250000E+00	-0.3492589E+00	-0.3472945E+00	-0.1964390E-02
0.2026342E+00	0.6250000E+00	-0.2696048E+00	-0.2675483E+00	-0.2056509E-02
0.2431611E+00	0.6250000E+00	-0.1919499E+00	-0.1891763E+00	-0.2773598E-02
0.2836879E+00	0.6250000E+00	-0.1166723E+00	-0.1129739E+00	-0.3698349E-02
0.3242148E+00	0.6250000E+00	-0.4413871E-01	-0.3946322E-01	-0.4675493E-02

0.3647417E+00	0.6250000E+00	0.2529736E-01	0.3100234E-01	-0.5704984E-02
0.4052685E+00	0.6250000E+00	0.9129786E-01	0.9818274E-01	-0.6884873E-02
0.4457954E+00	0.6250000E+00	0.1535412E+00	0.1619141E+00	-0.8372903E-02
0.4863222E+00	0.6250000E+00	0.2117243E+00	0.2220944E+00	-0.1037014E-01
0.5268490E+00	0.6250000E+00	0.2655635E+00	0.2786674E+00	-0.1310396E-01
0.1215805E+00	0.7291666E+00	-0.3889448E+00	-0.3756517E+00	-0.1329306E-01
0.1621074E+00	0.7291666E+00	-0.2941354E+00	-0.2831166E+00	-0.1101881E-01
0.2026342E+00	0.7291666E+00	-0.2012056E+00	-0.1920120E+00	-0.9193629E-02
0.2431611E+00	0.7291666E+00	-0.1106082E+00	-0.1032116E+00	-0.7396579E-02
0.2836879E+00	0.7291666E+00	-0.2278434E-01	-0.1729017E-01	-0.5494162E-02
0.3242148E+00	0.7291666E+00	0.6183815E-01	0.6536883E-01	-0.3530681E-02
0.3647417E+00	0.7291666E+00	0.1428470E+00	0.1445021E+00	-0.1655132E-02
0.4052685E+00	0.7291666E+00	0.2198475E+00	0.2199318E+00	-0.8434057E-04
0.4457954E+00	0.7291666E+00	0.2924647E+00	0.2915410E+00	0.9237528E-03
0.4863222E+00	0.7291666E+00	0.3603450E+00	0.3592681E+00	0.1076877E-02
0.5268490E+00	0.7291666E+00	0.4231574E+00	0.4230896E+00	0.6777048E-04

APPENDIX-4

* CHECK.FOR

* PROGRAM FOR CHECKING NEURAL NET SOLUTION

DATA NP,NL,NJ,NK /64,2,4,1/

CALL ALRN(NP,NL,NJ,NK)
STOP
END

SUBROUTINE ALRN(NP,NL,NJ,NK)

REAL NNW,NNW0

INTEGER ITER

REAL OI(2,5000),OJ(4),OK(1),WJ(4,2),WK(1,4)

REAL DWK(1,4),DELJ(4),DELK(1),THJ(4),THK(1)

REAL DTHJ(4),DTHK(1),NETJ(4),NETK(1),DWJ(4,2)

REAL TK(1,5000),KVAR(1,5000)

DATA ALP,NNW,THO1,THO2,EPS/0.0,0.4,1.0,1.0,1E-10/

DATA VOLT/220/

OPEN(UNIT=8,FILE='C:\FORTRAN\NEURAL\WT.DAT')

OPEN(UNIT=1,FILE='C:\FORTRAN\NEURAL\OJ1.DAT')

OPEN(UNIT=2,FILE='C:\FORTRAN\NEURAL\OJ2.DAT')

OPEN(UNIT=3,FILE='C:\FORTRAN\NEURAL\OJ3.DAT')

OPEN(UNIT=4,FILE='C:\FORTRAN\NEURAL\OJ4.DAT')

OPEN(UNIT=7,FILE='C:\FORTRAN\NEURAL\TST.DAT')

IP=1

DO 190 CUR=2.0,7,0.2

DO 190 TH=0,52,1

 OI(2,IP)=CUR/9.6

 OI(1,IP)=TH/98.7

 X=(OI(2,IP)*.25)+OI(1,IP)-.2675

 Y=(OI(2,IP)*1)+OI(1,IP)-.53

 IF((X.GT.0).AND.(Y.GT.0))THEN

 TK(1,IP)=(CUR*SIN(TH/57.3)-2.4)/5.0

 KVAR(1,IP)=VOLT*CUR*SIN(TH/57.3)/1000

```

        IP=IP+1
    ENDIF
190  CONTINUE
    NP=IP-1
    DO 333 IP=1,NP
333      TK(1,IP)=TK(1,IP)-0.2

        READ(8,*) ITP
        READ(8,*) ((WJ(J,I),I=1,NI),J=1,NJ)
        READ(8,*) (THJ(J),J=1,NJ)
        READ(8,*) ((WK(K,J),J=1,NJ),K=1,NK)
        READ(8,*) (THK(K),K=1,NK)

        ITER=ITP
    DO 290 IP=1,NP
    DO 940 IJ=1,NJ
        NETJ(IJ)=0.0
    DO 950 II=1,NI
        NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
950  CONTINUE
        SS=EXP(-(NETJ(IJ)-THJ(IJ))/THO1)
        OJ(IJ)=(1.0-SS)/(1.0+SS)

940  CONTINUE

    DO 960 IK=1,NK
        NETK(IK)=0.0
    DO 970 IJ=1,NJ
        NETK(IK)=NETK(IK)+WK(IK,IJ)*OJ(IJ)
970  CONTINUE
        OK(IK)=(NETK(IK)-THK(IK))/THO2
960  CONTINUE
        ERROR=TK(1,IP)-OK(1)
        WRITE(1,(2(E14.7,2X)))NETJ(1),OJ(1)
        WRITE(2,(2(E14.7,2X)))NETJ(2),OJ(2)
        WRITE(7,(4(E14.7,2X)))TK(1,IP),OK(1),ERROR,KVAR(1,IP)
        WRITE(3,(2(E14.7,2X)))NETJ(3),OJ(3)
        WRITE(4,(2(E14.7,2X)))NETJ(4),OJ(4)
290  CONTINUE
    CLOSE(2)
    CLOSE(3)
    RETURN
END

```

APPENDIX-5

* LRNSRT.FOR

* PROGRAM FOR NEURAL NET SOLUTION MAKING SIGMOID
* FUNCTION PIECE-WISE LINEAR

DATA NP,NT,NL,NJ,NK /64,500,2,6,1/

PRINT *, 'DO YOU WANT TO START FROM PREVIOUS VALUES ? (1/0)'

READ (*,*) NOYES

PRINT *, 'NO OF ITERATION ? '

READ(*,*) NT

CALL ALRN(NP,NT,NL,NJ,NK,NOYES)

STOP

END

SUBROUTINE ALRN(NP,NT,NL,NJ,NK,NOYES)

REAL NNW,NNW0

INTEGER ITER

REAL OI(2,600),OJ(6),OK(1),WJ(6,2),WK(1,6)

REAL DWK(1,6),DELJ(6),DELK(1),THJ(6),THK(1)

REAL DTHJ(6),DTHK(1),NETJ(6),NETK(1),DWJ(6,2)

REAL TK(1,600)

DATA ALP,NNW,THO1,THO2,EPS/0.0,0.5,1.0,1.0,1E-10/

OPEN(UNIT=1,FILE='VAR.DAT')

OPEN(UNIT=2,FILE='LINWT.DAT')

OPEN(UNIT=3,FILE='OUTLIN.DAT')

IP=1

DO 190 CUR=2.0,7,1.0

DO 190 TH=0,52,4.0

 OI(2,IP)=CUR/9.6

 OI(1,IP)=TH/98.7

 X=(OI(2,IP)*.25)+OI(1,IP)-.2675

 Y=(OI(2,IP)*1)+OI(1,IP)-.53

 IF((X.GT.0).AND.(Y.GT.0))THEN

 TK(1,IP)=(CUR*SIN(TH/57.3)-2.4)/5.0

```

        VAR=220*CUR*SIN(TH/57.3)
        WRITE(1,*)VAR

        IP=IP+1
    ENDIF
190    CONTINUE
        NP=IP-1
    DO 333 IP=1,NP
333    TK(1,IP)=TK(1,IP)-.2
        WRITE(*,*) 'NUMBER OF PATTERN = ',NP
        IF(NOYES.EQ.1) THEN
            READ(2,*) ITP
            READ(2,*) ((WJ(J,I),I=1,NI),J=1,NJ)
            READ(2,*) (THJ(J),J=1,NJ)
            READ(2,*) ((WK(K,J),J=1,NJ),K=1,NK)
            READ(2,*) (THK(K),K=1,NK)
            READ(2,*) NNW
        ENDIF

        IF (NOYES.EQ.0) THEN
            ITP=0
            THJ(1)=0.1
            THJ(2)=-0.2
            THJ(3)=0.3
            THJ(4)=0.1
            THJ(5)=-0.2
            THJ(6)=-0.1
            THK(1)=0.3

            DO 232 I=1,NJ
232    THJ(I)=0.0
            DO 233 I=1,NK
233    THK(I)=0.0
                WJ(1,1)=.1
                WJ(1,2)=-.1
                WJ(2,1)=-.234
                WJ(2,2)=0.34
                WJ(3,1)=.21
                WJ(3,2)=-0.3314
                WJ(4,1)=.1
                WJ(4,2)=-.091
                WJ(5,1)=-.234
                WJ(5,2)=0.34
            END DO
        END IF
    END DO

```

```

        WJ(6,1)=.053
        WJ(6,2)=-0.09314
        WK(1,1)=.132
        WK(1,2)=-.11
        WK(1,3)=.23
        WK(1,4)=-0.5
        WK(1,5)=0.2
        WK(1,6)=-0. 1
        ENDIF
    DO 2 I=1,NJ
        DTHJ(I)=0.0
    DO 2 J=1,NI
        DWJ(I,J)=0.0
2    CONTINUE
    DO 3 I=1,NK
        DTHK(I)=0.0
    DO 3 J=1,NJ
        DWK(I,J)=0.0
3    CONTINUE
        ITER=0
        IF(NOYES.EQ.1) THEN
            ITER=ITP
            NT=NT+ITP
        ENDIF

        NNW=NNW
    DO 210 L=1,NT
        ERR=0.0
    DO 10 IP=1 ,NP
11    DO 40 IJ=1,NJ
        NETJ(IJ)=0.0
    DO 50 II=1,NI
        NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
    IF (NETJ(IJ).GT.10292) PRINT *, NETJ(IJ)
50    CONTINUE

*****

    TI=NETJ(IJ)

    IF ((TI.LE.1.2).AND.(TI.GT.-1.2)) STRT=.4475*TI
    IF ((TI.LE.2.2).AND.(TI.GT.1.2)) STRT=.263*TI+.2214
    IF ((TI.LE.3.2).AND.(TI.GT.2.2)) STRT=.121668*TI+.532329

```

```

IF ((I.LE.4.2).AND.(TI.GT.3.2)) STRT=.048784*TI+.765557
IF ((TI.LE.5.2).AND.(TI.GT.4.2)) STRT=.02955*TI+.84634
IF (TI.GT.5.2) STRT=1

```

```

IF ((TI.GT.-2.2).AND.(TI.LE.-1.2)) STRT=.263*TI-.2214
IF ((TI.GT.-3.2).AND.(TI.LE.-2.2)) STRT=.121668*TI-.532329
IF ((TI.GT.-4.2).AND.(TI.LE.-3.2)) STRT=.048784*TI-.765557
IF ((TI.GT.-5.2).AND.(TI.LE.-4.2)) STRT=.02955*TI-.84634
IF (TI.LE.-5.2) STRT=-1

```

```
OJ(IJ)=STRT
```

```
*****
```

```

40  CONTINUE
    DO 60 IK=1,NK
        NETK(IK)=0.0
    DO 70 IJ=1,NJ
        NETK(IK)=NETK(IK)+WK(IK,IJ)*OJ(IJ)
        IF (NETK(IK).GT.10292) PRINT *, NETK(IK)
70  CONTINUE
    OK(IK)=(NETK(IK)-THK(IK))/THO2
60  CONTINUE
    DO 80 IK=1,NK
        DELK(IK)=(TK(IK,IP)-OK(IK))*(1./THO2)
    DO 90 IJ=1,NJ
        DWK(IK,IJ)=NNW*DELK(IK)*OJ(IJ)+ALP*DWK(IK,IJ)
        WK(IK,IJ)=WK(IK,IJ)+ DWK(IK,IJ)
90  CONTINUE
    THK(IK)=THK(IK)+DTHK(IK)
80  CONTINUE
    DO 100 IJ=1,NJ
        SUM=0.0
    DO 200 IK=1,NK
        SUM=SUM+DELK(IK)*WK(IK,IJ)
200 CONTINUE

```

```
*****
```

```
TI=NETJ(IJ)
```

```

IF ((TI.LE.1.2).AND.(TI.GT.-1.2)) DELJ(IJ)=.4475
IF ((TI.LE.2.2).AND.(TI.GT.1.2)) DELJ(IJ)=.263
IF ((TI.LE.3.2).AND.(TI.GT.2.2)) DELJ(IJ)=.121668
IF ((I.LE.4.2).AND.(TI.GT.3.2)) DELJ(IJ)=.048784
IF ((TI.LE.5.2).AND.(TI.GT.4.2)) DELJ(IJ)=.02955

```



```

IF (TL.GT.5.2) DELJ(IJ)=0.01

IF ((TL.GT.-2.2).AND.(TILE.-1.2)) DELJ(IJ)=.263
IF ((TL.GT.-3.2).AND.(TILE.-2.2)) DELJ(IJ)=.121668
IF ((TL.GT.-4.2).AND.(TILE.-3.2)) DELJ(IJ)=.048784
IF ((TL.GT.-5.2).AND.(TILE.-4.2)) DELJ(IJ)=.02955
IF (TILE.-5.2) DELJ(IJ)=0.01

DELJ(IJ)=DELJ(IJ)*SUM
*****
DELJ(IJ)=OJ(IJ)*(1.0-OJ(IJ))*SUM

DO 110 II=1,NI
  DWJ(IJ,II)=NNW*DELJ(IJ)*OI(II,IP)+ALP*DWJ(IJ,II)
  WJ(IJ,II)=WJ(IJ,II)+ DWJ(IJ,II)
110 CONTINUE
  DTHJ(IJ)=NNW*DELJ(IJ)+ALP*DTHJ(IJ)
  THJ(IJ)=THJ(IJ)+DTHJ(IJ)
100 CONTINUE

DO 777 IK=1,NK
  ERR=ERR+0.5*(TK(IK,IP)-OK(IK))**2
777 CONTINUE
10 CONTINUE

ITER=ITER+1
WRITE(*,'(I6,2X,3(F12.7,2X))') ITER,NNW,ERR
IF(ERR.LE.EPS.OR.ITER.EQ.NT) THEN
  CLOSE(2)
  WRITE(*,*) 'STORE THE WTS AND THITAS IN FILE ? (1/0)'
  READ(*,*) NY
IF(NY.EQ.1) THEN
  OPEN(UNIT=2,FILE='LINWT.DAT')
  WRITE(2,*) ITER
  WRITE(2,*) ((WJ(J,I),I=1,NI),J=1,NJ)
  WRITE(2,*) (THJ(J),J=1,NJ)
  WRITE(2,*) ((WK(K,J),J=1,NJ),K=1,NK)
  WRITE(2,*) (THK(K),K=1,NK)
  WRITE(2,*) NNW
ENDIF
GOTO 250
ENDIF
NNW=.2

```

```

210 CONTINUE
250 IF(ERR.LE.EPS) THEN
      PRINT *,'CONVERGED'
      ELSE
      PRINT *,'NONCONVERGENT'
      ENDIF

      DO 290 IP=1,NP
      DO 940 IJ=1,NJ
          NETJ(IJ)=0.0
      DO 950 II=1,NI
          NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
950 CONTINUE
*****

      TI=NETJ(IJ)

      IF ((TI.LE.1.2).AND.(TI.GT.-1.2)) STRT=.4475*TI
      IF ((TI.LE.2.2).AND.(TI.GT.1.2)) STRT=.263*TI+.2214
      IF ((TI.LE.3.2).AND.(TI.GT.2.2)) STRT=.121668*TI+.532329
      IF ((TI.LE.4.2).AND.(TI.GT.3.2)) STRT=.048784*TI+.765557
      IF ((TI.LE.5.2).AND.(TI.GT.4.2)) STRT=.02955*TI+.84634
      IF (TI.GT.5.2) STRT=1

      IF ((TI.GT.-2.2).AND.(TI.LE.-1.2)) STRT=.263*TI-.2214
      IF ((TI.GT.-3.2).AND.(TI.LE.-2.2)) STRT=.121668*TI-.532329
      IF ((TI.GT.-4.2).AND.(TI.LE.-3.2)) STRT=.048784*TI-.765557
      IF ((TI.GT.-5.2).AND.(TI.LE.-4.2)) STRT=.02955*TI-.84634
      IF (TI.LE.-5.2) STRT=-1

      OJ(IJ)=STRT
*****

940 CONTINUE

      DO 960 IK=1,NK
          NETK(IK)=0.0
      DO 970 IJ=1,NJ
          NETK(IK)=NETK(IK)+WK(IK,IJ)*OJ(IJ)
970 CONTINUE
      OK(IK)=(NETK(IK)-THK(IK))/THO2

```

```
960  CONTINUE
      OI1=OI(1,IP)
      OI2=OI(2,IP)
      TK1=TK(1,IP)
      ERROR=TK1-OK(1)
      WRITE(3,'(5(E14.7,2X))'OI1,OI2,TK1,OK(1),ERROR)
290  CONTINUE

      CLOSE(2)
      CLOSE(3)
      RETURN
      END
```

APPENDIX-6

* CHKSRT.FOR

* PROGRAM FOR CHECKING NEURAL NET SOLUTION WITH STRAIGHT
* LINES

DATA NP,NL,NJ,NK /64,2,6,1/

CALL ALRN(NP,NL,NJ,NK)

STOP

END

SUBROUTINE ALRN(NP,NL,NJ,NK)

REAL NNW,NNW0

INTEGER ITER

REAL OI(2,5000),OJ(6),OK(1),WJ(6,2),WK(1,6)

REAL OAJ(6),OAK(1),NETAK(1)

REAL DWK(1,6),DELJ(6),DELK(1),THJ(6),THK(1)

REAL DTHJ(6),DTHK(1),NETJ(6),NETK(1),DWJ(6,2)

REAL TK(1,5000),KVAR(1,5000)

DATA ALP,NNW,THO1,THO2,EPS/0.0,0.4,1.0,1.0,1E-10/

DATA VOLT/220/

OPEN(UNIT=8,FILE='C:\FORTRAN\NEURAL\LINWT.DAT')

OPEN(UNIT=1,FILE='C:\FORTRAN\NEURAL\OAJ1.DAT')

OPEN(UNIT=2,FILE='C:\FORTRAN\NEURAL\OAJ2.DAT')

OPEN(UNIT=3,FILE='C:\FORTRAN\NEURAL\OAJ3.DAT')

OPEN(UNIT=4,FILE='C:\FORTRAN\NEURAL\OAJ4.DAT')

OPEN(UNIT=5,FILE='C:\FORTRAN\NEURAL\OAJ5.DAT')

OPEN(UNIT=6,FILE='C:\FORTRAN\NEURAL\OAJ6.DAT')

OPEN(UNIT=7,FILE='C:\FORTRAN\NEURAL\TEST.DAT')

IP=1

DO 190 CUR=2.0,7,0.2

DO 190 TH=0,52,1

 OI(2,IP)=CUR/9.6

 OI(1,IP)=TH/98.7

```

*****
      X=(OI(2,IP)*.25)+OI(1,IP)-.2675
      Y=(OI(2,IP)*1)+OI(1,IP)-.53
      IF((X.GT.0).AND.(Y.GT.0))THEN
*****
      TK(1,IP)=(CUR*SIN(TH/57.3)-2.4)/5.0

      KVAR(1,IP)=VOLT*CUR*SIN(TH/57.3)/1000
      IP=IP+1
    ENDIF
190  CONTINUE
      NP=IP-1
    DO 333 IP=1,NP
333  TK(1,IP)=TK(1,IP)-0.2

      READ(8,*) ITP
      READ(8,*) ((WJ(J,I),I=1,NI),J=1,NJ)
      READ(8,*) (THJ(J),J=1,NJ)
      READ(8,*) ((WK(K,J),J=1,NJ),K=1,NK)
      READ(8,*) (THK(K),K=1,NK)

      ITER=ITP

      DO 290 IP=1,NP
      DO 940 IJ=1,NJ
          NETJ(IJ)=0.0
      DO 950 II=1,NI
          NETJ(IJ)=NETJ(IJ)+WJ(IJ,II)*OI(II,IP)
950  CONTINUE

*****

      TI=NETJ(IJ)

      IF ((TI.LE.1.2).AND.(TI.GT.-1.2)) STRT=.4475*TI
      IF ((TI.LE.2.2).AND.(TI.GT.1.2)) STRT=.263*TI+.2214
      IF ((TI.LE.3.2).AND.(TI.GT.2.2)) STRT=.121668*TI+.532329
      IF ((TI.LE.4.2).AND.(TI.GT.3.2)) STRT=.048784*TI+.765557
      IF ((TI.LE.5.2).AND.(TI.GT.4.2)) STRT=.02955*TI+.84634
      IF (TI.GT.5.2) STRT=1

      IF ((TI.GT.-2.2).AND.(TI.LE.-1.2)) STRT=.263*TI-.2214
      IF ((TI.GT.-3.2).AND.(TI.LE.-2.2)) STRT=.121668*TI-.532329

```

```
IF ((TI.GT.-4.2).AND.(TI.LE.-3.2)) STRT=.048784*TI-.765557
IF ((TI.GT.-5.2).AND.(TI.LE.-4.2)) STRT=.02955*TI-.84634
IF (TI.LE.-5.2) STRT=-1
OAJ(IJ)=STRT
```

```
*****
```

```
940 CONTINUE
```

```
DO 960 IK=1,NK
  NETAK(IK)=0.0
  DO 970 IJ=1,NJ
    NETAK(IK)=NETAK(IK)+WK(IK,IJ)*OAJ(IJ)
```

```
970 CONTINUE
```

```
OAK(IK)=(NETAK(IK)-THK(IK))/THO2
```

```
960 CONTINUE
```

```
ERROR=TK(1,IP)-OAK(1)
WRITE(1,'(2(E14.7,2X))')NETJ(1),OAJ(1)
WRITE(2,'(2(E14.7,2X))')NETJ(2),OAJ(2)
WRITE(7,'(4(E14.7,2X))')TK(1,IP),OAK(1),ERROR,KVAR(1,IP)
WRITE(3,'(2(E14.7,2X))')NETJ(3),OAJ(3)
WRITE(4,'(2(E14.7,2X))')NETJ(4),OAJ(4)
WRITE(5,'(2(E14.7,2X))')NETJ(5),OAJ(5)
WRITE(6,'(2(E14.7,2X))')NETJ(6),OAJ(6)
```

```
290 CONTINUE
```

```
CLOSE(2)
CLOSE(3)
RETURN
END
```

APPENDIX-7

The neural network for the reactive power controller has two inputs, six hidden and one output nodes performing a continuous mapping of $X = I \cdot \sin\theta$. Range of input 'I' varied from 2 to 7 and input ' θ ' from 0° to 55° . 'I' and ' θ ' were normalized by dividing the corresponding inputs by 9.6 and 98.7 respectively. Gradients chosen for the PWL function were given through Eq. 2.23a to 2.23f

$m_1 = 0.4475$	$X_1 < 1.2$
$m_2 = 0.263$	$1.2 \leq X_2 < 2.2$
$m_3 = 0.12167$	$2.2 \leq X_3 < 3.2$
$m_4 = 0.04878$	$3.2 \leq X_4 < 4.2$
$m_5 = 0.02955$	$4.2 \leq X_5 < 5.2$
$m_6 = 1$	$5.2 < X_6$

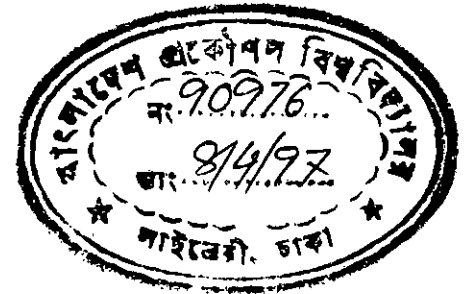
After learning, the weights settled as,

1. Total iteration cycle: 80800
2. Final RMS error: .00099
3. Weights for node θ : $W_{11} = -1.644$, $W_{21} = -1.69$, $W_{31} = 4.77$,
 $W_{41} = 4.305$, $W_{51} = -8.416$, $W_{61} = 2.38$.
4. Weights for node I: $W_{12} = -1.2808$, $W_{22} = 2.155$, $W_{32} = -5.376$,
 $W_{42} = 3.525$, $W_{52} = 3.2767$, $W_{62} = -0.11$
5. Weights for output node: $W_1 = -1.0811$, $W_2 = 1.241$, $W_3 = -0.192$,
 $W_4 = -2.0332$, $W_5 = -0.269$, $W_6 = 1.88$

To make the implementation of the break points easier and more precise, the break points are increased to 3 times than the training values. This modification requires the inputs or the weights to be amplified 3 times to achieve the same target output. The slopes of the equations are kept unaltered. To impose the condition of Eq. (3.16) it is observed that 5th node of the hidden layer has the greatest value of $|W_{51}| + |W_{52}|$. Hence Eq. (15) for node no. 5 will be

$$0.4475 \times (8.416 + 3.2767) = 5.23$$

So, as explained earlier, the inputs should be increased more than 5.23 times to satisfy the condition. Therefore, the overall amplifying factor for the input stands $5.23 \times 3 \approx 16$. The values of resistances obtained through Eq. (3.9) to Eq. (3.15) are given in Table A.7. Node 1, 2 and 6 have responses limited within the first two sections of the PWL function. The only one output node is implemented by an analog adder circuit.



BREAKING POTENTIALS : 1.611, 2.4, 2.765, 2.911, and 3.

RESISTANCE VALUES :

Node Number	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇
1	5.085	6.527	0.929	1.00			
2	4.947	3.88	1.035	1.00			
3	13.728	12.182	36.963	7.833	2.78	1.00	
4	16.529	20.187	17.406	8.511	3.02	1.086	1.00
5	7.781	19.98	291.60	7.833	2.78	1.00	
6	3.512	76.157	0.886	1.00			

Table A.7 The resistance values for different nodes