12-2014

Direct Torque Control of an Induction Machine Using Multi-Layer Perceptron Network

Adithya Chandrashekharan

DIRECT TORQUE CONTROL OF AN INDUCTION MACHINE USING MULTI-LAYER PRECEPTRON NEURAL NETWORK

by

Adithya Chandrashekharan

A thesis submitted to the Graduate College
in partial fulfilment of the requirements for the
degree of Master of Science in Engineering (Electrical)
Electrical and Computer Engineering
Western Michigan University
December 2014

Thesis Committee:

Johnson Asumadu, Ph.D., Chair
Ralph Tanner, Ph.D.
Masood Zandi Atashbar, Ph.D.

DIRECT TORQUE CONTROL OF AN INDUCTION MACHINE USING MULTI-LAYER PRECEPTRON NEURAL NETWORK

Adithya Chandrashekharan, M.S.E.

Western Michigan University, 2014

In conventional direct torque control (DTC) scheme of induction motor (IM), Proportional Integral Controller (PI) is used as the speed controller. PI controller is more suitable in steady state condition and for linear system but both DTC and IM are mostly nonlinear. Multi-Layer perceptron neural network. (MLPNN) controller is more suitable and performs better than PI controller. The switching table of the conventional DTC is replaced by the MLPNN controller. The MLPNN inputs are, the magnitude of the stator flux, torque and the Voltage sectors. Levenberg-Marquardt back propagation technique has been used to train the MLPNN. The output of the MLPNN are the gate signals to the inverter, which switch to produce the required voltage vectors. The Hysteresis band for torque is 1% and the hysteresis band for the flux is 0.5%.The advantage of using MLPNN is that, in classical DTC look up table is used to select switching states, thus size requirement will increase with some advanced control. MLPNN will take less memory and is more reliable. Despite the fact that the proposed scheme does not eliminate the torque, flux and current ripples to a great extent, the dropping of the stator flux due to the sector changes is eliminated. Also a faster stator flux and Torque response is achieved in the transient state.

ACKNOWLEDGEMENTS

I would like to begin by thanking Dr. Johnson Asumadu, for giving me on opportunity to work on this thesis by supervising my work, servicing as the chair, providing valuable advice and for his guidance during my graduate study. I would also like to thank Dr. Ralph tanner and Dr. Massood Zandi Atashbar, for serving on my thesis committee and teaching the courses that helped me towards my research.

I would also like to thank my colleague, Bryan Ford, for his valuable guidance on neural networks, which is an integral part of this thesis.

A special thanks to all my classmates for having supported me and making my graduate school experience so much more enjoyable.

Finally, I want to extend my deepest thanks and apperception to my parents and my family for their never-ending support and encouragement.

Adithya Chandrashekharan

TABLE OF CONTENTS

Table of Contents-continued

Table of Contents-continued

LIST OF TABLES

LIST OF FIGURES

List of Figures-continued

1. Introduction

Direct Torque control (DTC) was first proposed by [1] in 1986, and it can be summarized as follows:

Stator flux is a time integral of the stator EMF. Therefore, its magnitude strongly depends on the stator voltage. The developed torque is proportional to the sine of the angle between the stator and the rotor flux vectors. The reaction of the rotor flux to the changes in the stator voltage id slower than that of the stator flux [2].

The magnitude of the stator flux and developed torque can be directly controlled by selection of the space vectors of the stator voltage, i.e. selection of the inverter states.

If the misalignment of the voltage vectors with the stator flux is within ±90° the stator flux increases and if it exceeds ±90° the flux increases. The developed torque can be controlled by selecting inverter states such that the stator flux vector is accelerated, stopped or decelerated [2].

The estimated speed is subtracted from the desire speed and this error signal is used to generate the torque reference signal. The estimated speed generates the reference signal for the stator flux linkage which is compared with the estimated stator flux linkages. These errors of the electromagnetic Torque, stator flux magnitude and angular position of the

stator flux, determine the stator voltage space vector that is applied to the motor during each sampling interval. [3]

DTC uses feedback control of torque and stator flux, which are computed using the measured stator voltages and currents.

Stator reference frame model of the induction motor, hence avoiding the trigonometric operations in the co-ordinate transformation of the synchronous reference frames. This is one of the key advantages of DTC. [4]

This paper uses a Multi-Layer perceptron neural network (MLPNN) to replace the switching in order to achieve faster torque response in the transient state.

The MLPNN is trained using Levenberg-Marquardt back-propagation algorithm as described in [5].

This method involves forward propagating the inputs to the number of layers and then back-propagating it till the optimal convergence is achieved. This is achieved by assigning random weights initially.

In [6], the direct torque control using fuzzy logic controlled has been simulated. It is found that by replacing the switching table, with the fuzzy logic controller the distortion in the phase currents have been reduced.

In this paper, the DTC scheme is implemented using a MLPNN. This has been simulated using Matlab/Simulink®. The stator flux, the developed torque and the stator flux angle are computed using the equations in 2. The algorithm used for the training of the neural network in discussed in section 3.3.3. It is seen that by using the MLPNN for the DTC

control scheme, the sector changes do not cause stator Flux drop (Hexagonal Trajectory of Stator Flux). Also in, Transient state, the Stator flux response and the Torque response.

2.Theoretical Background

2.1 Voltage Equations

2.1.1 Stator Voltage Equations

$$V_{sa}(t) = R_s i_{sa} + \frac{d\lambda_{sa}(t)}{dt} \qquad (2.1)$$

$$V_{sb}(t) = R_s i_{sb} + \frac{d\lambda_{sb}(t)}{dt} \qquad (2.2)$$

$$V_{sc}(t) = R_s i_{sc} + \frac{d\lambda_{sc}(t)}{dt} \qquad (2.3)$$

V_{sa} , V_{sb} and V_{sc} are the stator voltage of the phases A, B and C respectively

i_{sa} , i_{sb} and i_{sc} are the stator currents of the phases A, B and C respectively.

λ_{sa} , λ_{sb} and λ_{sc} are the stator fluxes of the phases A, B and C respectively.

R_s is the stator resistance

2.1.2 Rotor Voltage Equations:

$$V_{ra}(t) = R_r i_{ra} + \frac{d\lambda_{ra}(t)}{dt} \qquad (2.4)$$

$$V_{rb}(t) = R_r i_{rb} + \frac{d\lambda_{rb}(t)}{dt} \qquad (2.5)$$

$$V_{rc}(t) = R_r i_{rc} + \frac{d\lambda_{rc}(t)}{dt} \qquad (2.6)$$

V_{ra} , V_{rb} and V_{rc} are the rotor voltage of the phases A, B and C respectively

i_{ra} , i_{rb} and i_{rc} are the rotor currents of the phases A, B and C respectively.

λ_{ra} , λ_{rb} and λ_{rc} are the rotor fluxes of the phases A, B and C respectively.

R_r is the rotor resistance

2.2 3-Phase to 2-phase Transformation

A dynamic model of the 3-phase induction machine can be derived from the two-phase machine if the equivalence between three and two phases is established. The equivalence is based on the MMF produced in the 2-phase and 3-phase windings and equal current magnitude. Assuming that each of the three-phase windings has T_1 turns per phase and

equal current magnitudes the 2-phase windings will have $\frac{3T_1}{2}$turns per phase of mmf equality. [4]

The relationship between dq0 and abc currents and voltages are as shown below.

$$V_{dq0} = [T_{abc}]V_{abc} \quad (2.7)$$

$$i_{dq0} = [T_{abc}]i_{abc} \quad (2.8)$$

Where:

V_{dq0} is the dq axis voltage vector

i_{dq0} is the dq axis current vector

Where T_{abc} is the transformation matrix given by:

$$T_{abc} = \begin{bmatrix} \cos\theta_c & \cos(\theta_c - \frac{2\pi}{3}) & \cos(\theta_c + \frac{2\pi}{3}) \\ \sin\theta_c & \sin(\theta_c - \frac{2\pi}{3}) & \sin(\theta_c + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (2.9)$$

2.3 Flux Equations

2.3.1 Stator Flux Equations

$$\lambda_{sd} = \int (V_{sd} - R_s i_{sd})dt \quad (2.10)$$

$$\lambda_{sq} = \int (V_{sq} - R_s i_{sq})dt \quad (2.11)$$

Where:

λ_{sd} and λ_{sq} are the stator fluxes of the d and q axis respectively

V_{sd} and V_{sq} are the stator voltages of the d and q axis respectively

i_{sd} and i_{sq} are the stator currents of the d and q axis respectively

2.3.2 Rotor Flux Equations

$$\lambda_{rd} = \int (\omega\lambda_{rq} - R_r i_{rd})dt \qquad (2.12)$$

$$\lambda_{rq} = \int (\omega\lambda_{rd} - R_r i_{rq})dt \qquad (2.13)$$

Where:

λ_{rd} and λ_{rq} are the rotor fluxes of the d and q axis respectively

V_{rd} and V_{rq} are the rotor voltages of the d and q axis respectively

i_{rd} and i_{rq} are the rotor currents of the d and q axis respectively

ω is the electrical rotor speed in rad/sec

2.4 Current Equations

2.4.1 Stator Current Equations

$$i_{sd} = \frac{L_m\lambda_{rd} - L_r\lambda_{sd}}{L^2{}_m - L_s L_r} \qquad (2.14)$$

$$i_{sq} = \frac{L_m\lambda_{rq} - L_r\lambda_{sq}}{L^2{}_m - L_s L_r} \qquad (2.15)$$

2.4.2 Rotor Current Equations

$$i_{rd} = \frac{L_m \lambda_{sd} - L_s \lambda_{rd}}{L^2{}_m - L_s L_r} \tag{2.16}$$

$$i_{rq} = \frac{L_m \lambda_{sq} - L_s \lambda_{rq}}{L^2{}_m - L_s L_r} \tag{2.17}$$

Where:

L_m is the magnetizing inductance per phase.

L_s is the stator self-inductance per phase

L_r is the rotor self-inductance per phase referred from the stator

2.5 Reference Frame & Net Electromagnetic Torque Equation

For the purpose of this paper, stator reference frame induction model will be used.

The stator speed is zero. Since the stator is the reference frame, the speed of the arbitrary reference frameω_c is zero i.e. $\omega_c = 0$.

The resulting model is as shown below.

$$\begin{bmatrix} V_{sq} \\ V_{sd} \\ V_{rq} \\ V_{rd} \end{bmatrix} = \begin{bmatrix} R_s + L_s p & 0 & L_m p & 0 \\ 0 & R_s + L_s p & 0 & L_m p \\ L_m & -\omega L_m & R_r + L_r p & -\omega L_r \\ \omega L_m & L_m p & \omega_r L_r & R_r + L_r p \end{bmatrix} \begin{bmatrix} i_{sq} \\ i_{sd} \\ i_{rq} \\ i_{rd} \end{bmatrix} \tag{2.18}$$

Where

p is the differential operator

Since the input variables are well defined, the stator d and q axes voltages and the electromagnetic can be determined easily.

$$V_{sq} = V_{as} \tag{2.19}$$

$$V_{sd} = \frac{V_{cs} - V_{bs}}{\sqrt{3}} \tag{2.20}$$

$$T_{em} = \frac{p}{2} L_m (i_{sq} i_{rd} - i_{sd} i_{rq}) \tag{2.21}$$

2.6 Basic Concept of DTC

Stator flux is a time integral of the stator voltage. Therefore, its magnitude strongly depends on the stator voltage. The Developed torque is proportional to the sine of angle between the stator and rotor flux vectors.

Reaction of rotor flux to changes in stator voltage is slower than that of the stator flux. Consequently, both the magnitude of stator flux and the developed torque can be directly controlled by proper selection of space vectors of stator voltage, that is, selection of consecutive inverter states. Specifically: Nonzero voltage vectors whose misalignment with the stator flux vector does not exceed ±90° cause the flux to increase. Nonzero voltage vectors whose misalignment with the stator flux vector exceeds ±90° cause

The flux to decrease. Zero states, 0 and 7, (in the case of a two level inverter) practically do not affect the vector of stator flux which, consequently, stops moving. The developed torque can be controlled by selecting such inverter states that the stator flux vector is accelerated, stopped, or decelerated [2].

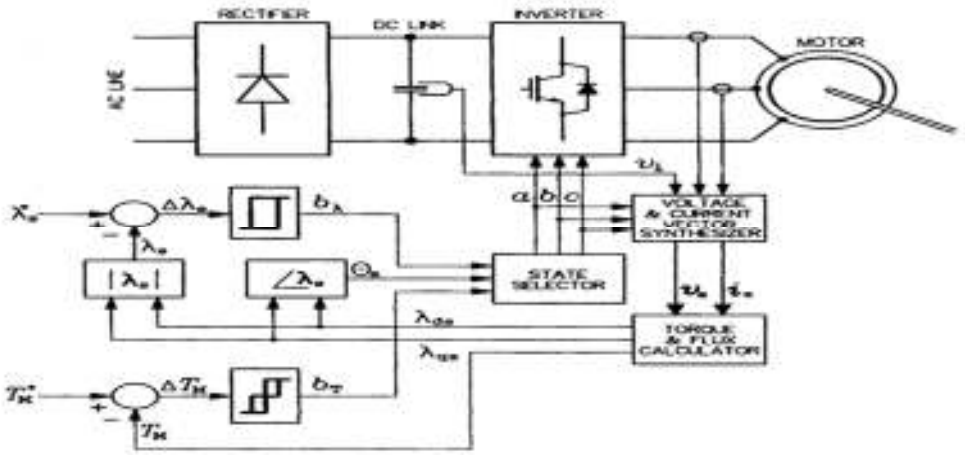

Figure 2.1 Block Diagram of a DTC Drive System [2]

2.6.1 Inverter Switching States

As an example a two level inverter is considered

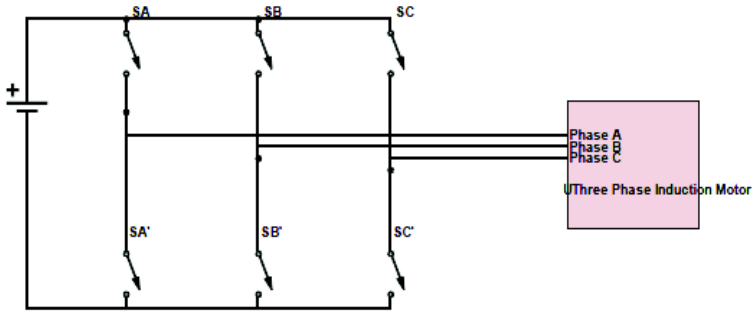

Figure 2.2 Three Phase Inverter with 8 switching states

It is known the at any given instant of time three switches are closed, however two switches on the same leg cannot be on simultaneously as it would result in a short circuit. Each inverter leg can assume two states only, and the number of states of the whole inverter is eight (2^3). Taking as an example phase A, the switching variable is defined to assume the value of 1 if switch SA is on and switch SA' is off. If, conversely, SA is off and SA' is on, assumes the value of 0. Similarly it can be defined for phase B and C. This can be associated with the digital signals.

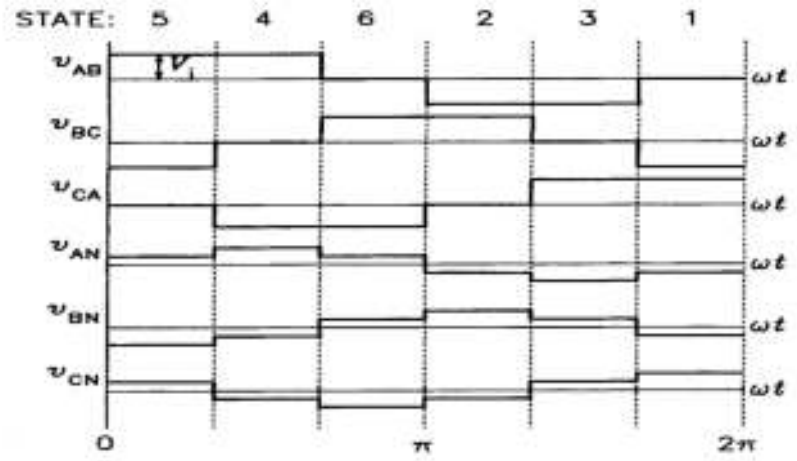

Figure 2.3 Output voltage waveforms in a voltage source inverter in the square-wave mode. [2]

For Example is the switches SA, SB and SC' are on the corresponding states will be 1 1 and 0 respectively.

$110_{(2)} = 6$, this corresponds to the state 6.

The line to neutral voltages can be determines as follows:

$$V_{an} = \frac{2Sa - Sb - Sc}{3} V_{dc} \qquad (2.22)$$

$$V_{bn} = \frac{-Sa + Sb - Sc}{3} V_{dc} \qquad (2.23)$$

$$V_{cn} = \frac{-Sa - Sb + 2Sc}{3} V_{dc} \qquad (2.24)$$

Where

Sa, Sb and Sc are the gate signals the phases A, B and C of the inverter respectively.

V_{dc} is the DC supply voltage to the inverter

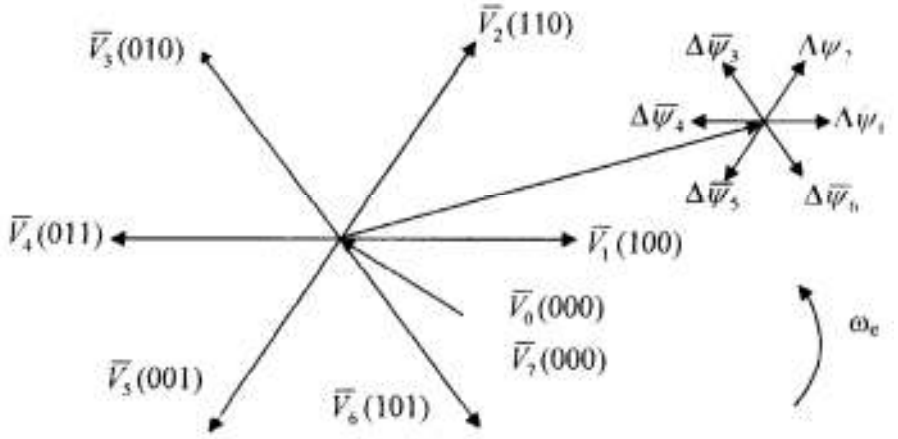

Figure 2.4 Inverter voltage vectors and corresponding stator flux with respect to time [2]

The outputs of the flux and torque comparators are used for the inverter switching table. The electromagnetic torque and the stator flux for the sector selectors are measured as follows

$$\Delta T_{em} = T_{em\,ref} - T_{em} \qquad (2.25)$$

$$\Delta\lambda_s = \lambda_{sref} - \lambda_s \qquad (2.26)$$

The flux controller is governed by the conditions given below

Condition	$\boldsymbol{\lambda_{s,cmd}}$
$\boldsymbol{\lambda_{s,er} > \Delta\lambda_s}$	1
$\boldsymbol{\lambda_{s,er} \leq \Delta\lambda_s}$	0

Table 2-1 Conditions for Flux Controller

The Torque controller is governed by the Conditions given below

Condition	$\boldsymbol{T_{em,cmd}}$
$\boldsymbol{T_{em,er} \geq \Delta T_{em}}$	1
$\boldsymbol{-\Delta T_{em} < T_{em,er} < \Delta T_{em}}$	0
$\boldsymbol{T_{em,er} \leq -\Delta T_{em}}$	-1

Table 2-2 Conditions for Torque Controller

Considering the six sectors shown in Figure (3-4) the stator flux switching sectors can be distributed as follows.

Starting Angle	Ending Angle	Sector Number
0°	60°	1
60°	120°	2
120°	180°	3
-180°	-120°	4

240°	300°	5
300°	360°	6

Table 2-3 Division of angles into sectors

The Vector Voltages V0 to V7 are associated with their binary number equivalent

Digital outputs associated with Switches	Voltage Vector Number
000	V0
001	V1
010	V2
011	V3
100	V4
101	V5
110	V6
111	V7

Table 2-4 Vector Voltages associated with their binary number equivalent

Based on the data above the vector voltages associated with each condition of torque, Flux and the Sector angle a switching table used as a look up table can be developed

λ_s	T_{em}	Sector 1	Sector 2	Sector 3	Sector 4	Sector 5	Sector 6
1	1	V3	V2	V6	V4	V5	V1
1	0	V7	V0	V7	V0	V7	V0
1	-1	V5	V1	V3	V2	V6	V4
0	1	V2	V6	V4	V5	V1	V3
0	0	V0	V7	V0	V7	V0	V7
0	-1	V4	V5	V1	V3	V2	V6

Table 2-5 Switching Table

All the equations mentioned in this chapter, aid with the development of a Simulink model to study the behavior of a direct torque control system.

This is discussed in detail in Chapter 5.

3.Neural Networks

3.1 Introduction to Neural Networks

An artificial neural network (ANN) is an information processing model that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this model is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well [7].

Neural networks have several distinct advantages, like adaptive learning, self-organization and real time operation to name a few. In recent years, neural networks have been a subject of discussion, as they can be applied to any field of study. In electrical engineering, the neural networks have been used widely in power systems, power electronics, communication systems etc. and have produced some amazing results [7]

3.2 Basic Structure of a Neural Network

The starting point for most neural networks is a model neuron. This neuron consists of multiple inputs and a single output. Each input is modified by a weight, which multiplies with the input value. The neuron will combine these weighted inputs and, with reference to a threshold value and activation function, use these to determine its output. This behavior follows closely our understanding of how real neurons work [7] .

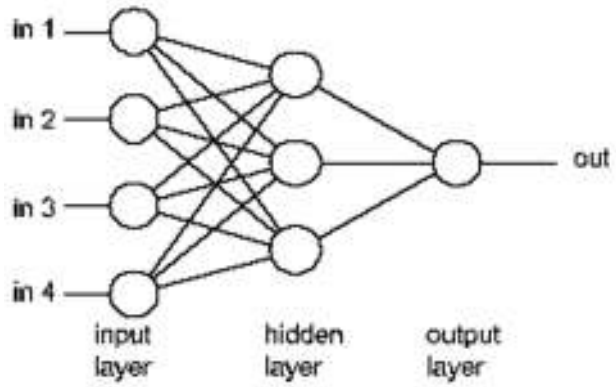

Figure 3.1 Generalized Structure of a Neural Network [5]

3.2.1 Structure of a Neuron

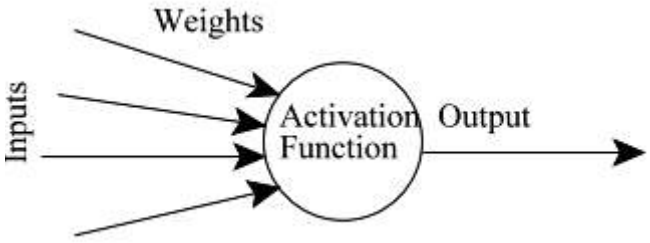

Figure 3.2 An Artificial Neuron [5]

$$TF\ (n) = \sum_1^n x_n w_{nj} \qquad (3.1)$$

$$O_j = \varphi\big(Tf(n)\big) \qquad (3.2)$$

3.3 TYPES OF NEURAL NETWORK

There are several types of neural networks like multilayer perceptron network, probabilistic neural networks, general regression neural networks, radial basis function networks, cascade correlation, functional link networks, kohen networks.

Multilayer perceptron networks (MLPNN) (also known as multilayer feed-forward network) is one the most widely used.

This paper will focus primarily on MLPNN for the purpose of DTC of an induction motor.

3.3.1 Multilayer Perceptron Neural Network Model

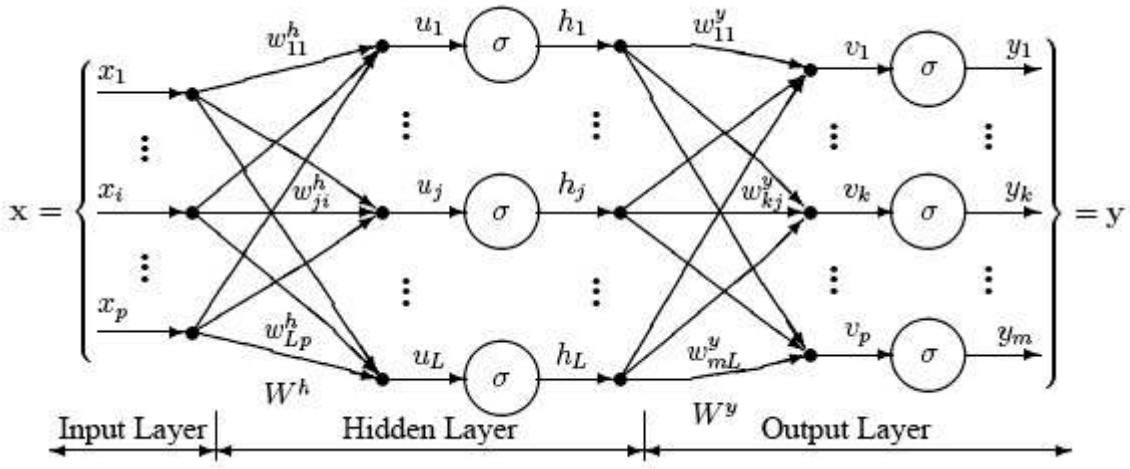

Figure 3.3 Illustration of a Multilayer Perceptron Neural Network Model. [8]

This network has an input layer with three neurons, one hidden layer with three neurons and an output layer (on the right) with three neurons. There is one neuron in the input layer

for each predictor variable. In the case of categorical variables, N-1 neurons are used to represent the N categories of the variable [8].

Input Layer — A vector of predictor variable values (x1,x2,x3,...xp) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values so that the range of each variable is -1 to 1. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron [8].

Hidden Layer — Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight (w_{ji}), and the resulting weighted values are added together producing a combined value u_j. The weighted sum (u_j) is fed into a transfer function, σ, which outputs a value u_j. The outputs from the hidden layer are distributed to the output layer [8].

Output Layer — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight (w_{kj}), and the resulting weighted values are added together producing a combined value v_j. The weighted sum (v_j) is fed into a transfer function, σ, which outputs a value y_k. The y values are the outputs of the network. If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single y value. For classification problems with categorical target variables, there are N neurons in the output layer producing N values, one for each of the N categories of the target variable [8].

3.3.2 Generalized Levenberg-Marquardt Algorithm [29]

Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian Matrix. When the performance function has the form of a sum of squares (as is typical in the training of forward feed Networks), then the Hessian Matrix can be approximates as [9]

$$H = J^T J \tag{3.3}$$

Where J represents the Jacobian Matrix and J^T is its Transpose.

The Gradient is computed as

$$g = J^T e \tag{3.4}$$

Where e is the vector of network errors.

The Jacobian Matrix J, consists of the first order derivatives of the network errors with respect to the weights and biases. The Jacobian matrix can be computes through a standard back-propagation technique that is much less complex than computing the Hessian Matrix. [9]

3.3.3 The Multilayer Perceptron Learning Algorithm using Levenberg-Marquardt Technique

i. Initialize the random weights and calculate the total error
ii. Update the weights
iii. Again calculate the error

iv. If the new error is more than the error from the initial step, increase the co-efficient by a random factor.
v. Go to step 2
vi. If the new error is lesser than the initial error, accept the step.
vii. Repeat step 2 and step 4 or 5 till the new total error is smaller than the required value.

Calculate the net values for all neurons in the first layer

$$\mathrm{net}_j^1 = \sum_{i=1}^{ni} I_i w_{j,i}^1 + w_{j,0}^1 \qquad (3.5)$$

Calculate the Outputs

$$y_j^1 = f_j^1 w_{j,i}^1 + w_{j,0}^1 \qquad (3.6)$$

Calculate the slopes

$$s_j^1 = \frac{\partial y_j^1}{\partial \mathrm{net}_j^1} \qquad (3.7)$$

Where,

$I_i = inputs\ to\ the\ network$

j= the index of neurons in the first layer

y_j^1 = The output node of neuron j for the first layer

f_j^1 = The activation function of the neuron j for the first layer

$w_{j,0}^1$ = the bias weight of neuron j for the first layer

$w_{j,i}^{1}$= The weight of the ith input node of the jth neuron

Superscript 1 represents the first layer

By using the outputs of the previous layers, we can calculate the net values, outputs and the slope for the succeeding n layers. Then calculate the final output of the nth layer.

$$o_j = f_j^n\left(net_j^n\right) \tag{3.8}$$

Calculate the error at the output j and initial δ as the slope of the output j

$$e_j = d_j - o_j \tag{3.9}$$

$$\delta_{j,j}^n = s_j^n \tag{3.10}$$

$$\delta_{j,k}^n = 0 \tag{3.11}$$

Where,

d_j= the desired output at desired j

o_j= the actual output at output j

$\delta_{j,j}^n$ = self-back-propagation

$\delta_{j,k}^n$= back-propagation from the other neurons in the same layer.

Back-propagate δ from the inputs of the nth layer to the outputs of the (n-1)th layer, till (n-1)=1 (i.e., arrives at the starting layer).

$$\delta_{j,k}^{n-1} = w_{j,k}^n \delta_{j,j}^n \tag{3.12}$$

Note: It is necessary, while back-propagating, to compute the back-propagation from all the succeeding layers and the self-back-propagation.

4.Development of the Simulink Model

The induction motor model used for this thesis has been taken from [3]

The overall model of the developed DTC system is given below.

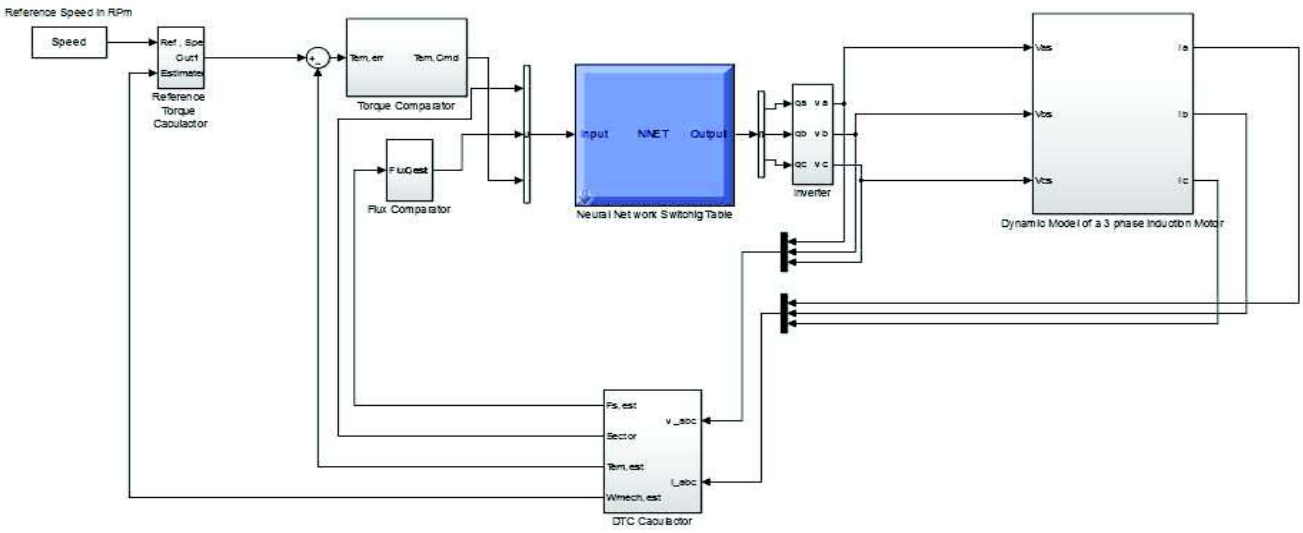

Figure 4.1 DTC model with ANN switching Table

4.1 Comparators

4.1.1 Torque Comparator

A Matlab® user defined function block has been used. Torque error is the input to this block.

The output of this block is 1, 0 or -1 based on the conditions defined in Table 3-2.

The $\Delta T_{em}$is determined to be 0.6322. The calculations for the same are shown in the M-File for the calculation of **Δ**Tem and **Δλ**s

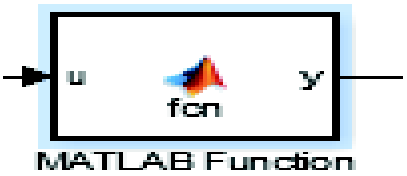

Figure 4.2 Matlab® user defined function block

The code written is as given below

```
function y = fcn(u)
y=0;
if u>=0.6322
    y=1;
elseif u<=-0.6322
    y=-1;
elseif u<0.6322 && u>-0.6322
    y=0;
end
```

The each torque command output serves a distinct function.

If the Tem,cmd=1; then the torque need to be increased

If the Tem,cmd=0; the Torque requires no change

If the Tem,cmd=-1; the Torque need to be decreased

The above conditions are reflected on the space vector plane. The top half represents the torque increase "1" the, the zero Vectors represent the No change "1" and the bottom half of the plane represents the torque decrease "-1"

4.1.2 Flux Comparator

A Matlab® user defined function block (figure 4.2) has been used. Flux error is the input to this block.

The output of this block is 1 or based on the conditions defined in Table 3.1

The $\Delta\lambda_s$is determined to be 0.0073. The calculations for the same are shown in the M-File for the calculation of $\boldsymbol{\Delta}$Tem and $\boldsymbol{\Delta\lambda}$s

The code written is as given below

```
function y = fcn(u)

if u>=0.0073
    y=1;
elseif u<= -0.073
    y=0;
end
```

Each torque command output serves a distinct function.

If the Fs, cmd=1; then the Flux needs to be increased

If the Fs, cmd=0; the Flux needs to be Decreased

The above conditions are reflected on the space vector plane. The Right half represents the Flux increase “1” and the left half of the plane represents the torque decrease “0”.

4.2 DTC Calculator

DTC calculator, makes use of the Induction motor currents ,the inverter output voltages (input to the Induction motor) and the machine parameters to determine the Magnitude of the Stator flux, the angle of the stator flux and the Electromagnetic torque developed by the Induction machine. This is achieved by the substituting the instantaneous values of the Induction motor currents and the inverter output voltages in the equations in chapter 3.

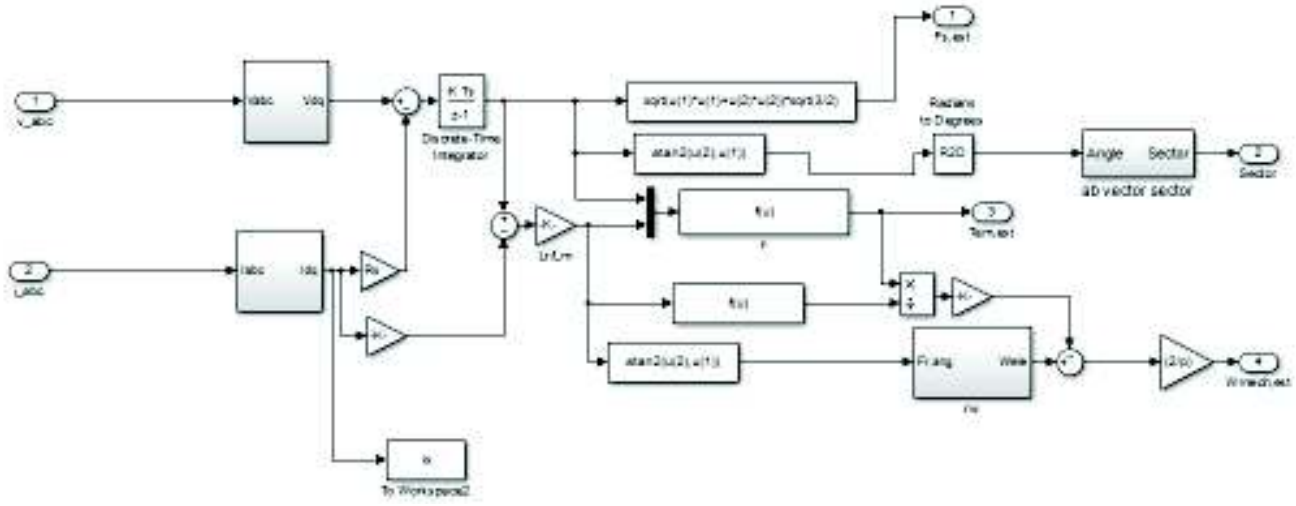

Figure 4.3 Top-level Simulink file for the DTC Calculator

4.2.1 ABC to DQ Transformation

In order to work with the dynamic model of an induction motor, it is necessary to transform the 3 phase voltages and currents to 2 phase. These are governed by (3.7), (3.8) and (3.9).

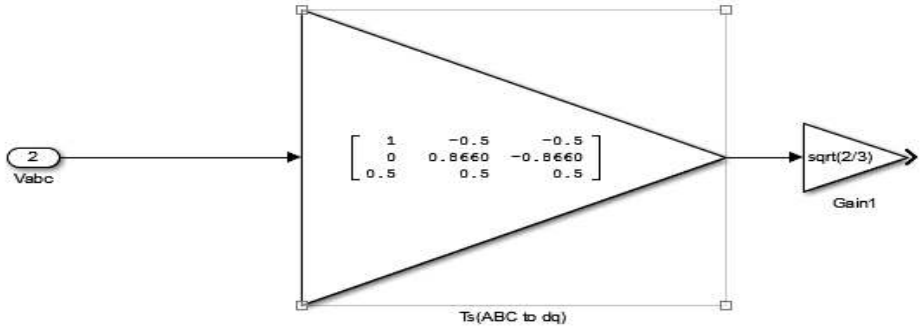

Figure 4.4 ABC to dqo transformation using Simulink blocks

4.2.2 Estimation of Stator Flux Magnitude ($|\lambda_s|$)

Equations (3.10) and (3.11) can be used to estimate the magnitude of stator flux.

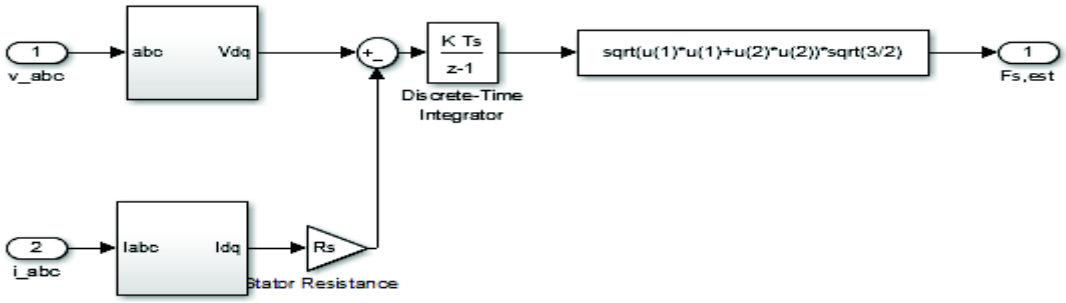

Figure 4.5 Simulink Model in the Estimation of Stator Flux Magnitude

4.2.3 Estimation of Stator Flux Angle.

Using (3.10) and (3.11) the flux which is in complex form is determine. From this, the magnitude and angle can be derived.

Figure 4.6 Simulink Model to Estimate Stator Flux Angle

4.2.4 Placing the Stator Flux Angles into Sectors

From table 3-3, where the angle for the sectors have been developed, it is possible to create a Simulink block. This outputs are the sector numbers. The output of the stator flux angle estimator (figure 5.5) is in radians. For simplicity, it is converted to degrees using the inbuilt Simulink block "R2D".

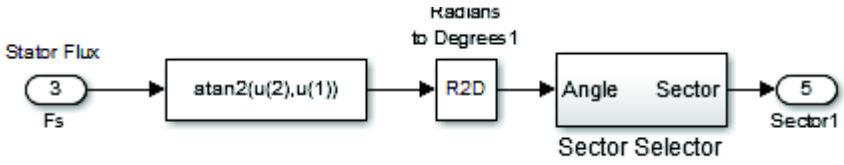

Figure 4.7 Top-level diagram of the sector selector

Due to the change in data type, form Boolean to double, a convert block is used. At the end a saturation block is used. The limits of the saturation block from 1 to six which corresponds to the sector numbers, in order to avoid the output of any number other than the desired.

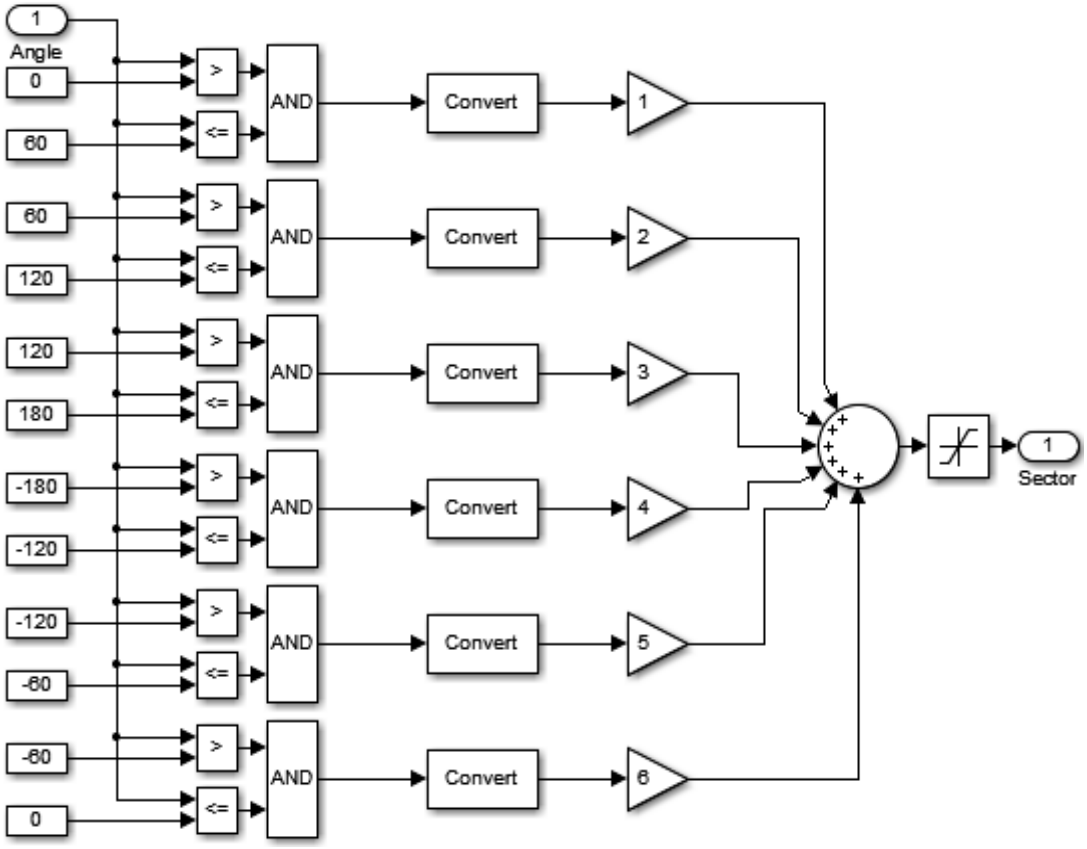

Figure 4.8 Boolean implementation of the sector selector block

4.2.5 Electromagnetic Torque Estimator

Using (3.21) Electromagnetic torque is determined. This is the estimated electromagnetic torque

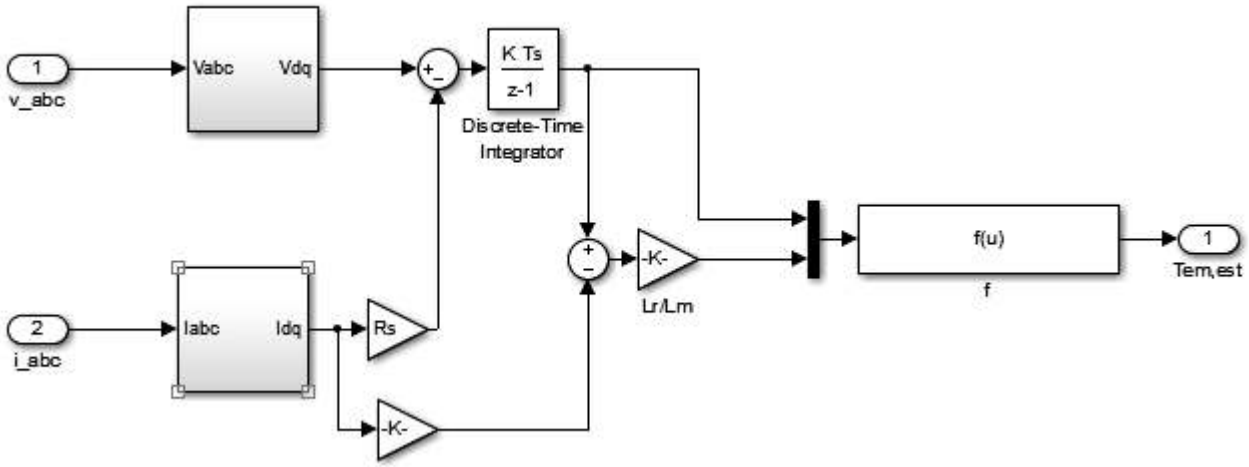

Figure 4.9 Electromagnetic Torque Estimator

4.3 Inverter

A 3-phase inverter is an integral part of the of the DTC control system. The output 3-phase voltages of the inverter is the input to the 3-phase induction motor. The output of the switching table (the neural network in this case), are the gate signals to the switching devices of the inverter. Care should be taken that two switches on the same phase should not operate at the same time.

In Simulink, however, it is not possible to build a 3 phase inverter unless a specific toolbox is used. Due to the absence of this tool box in the version used in this thesis, a mathematical model of a 3 phase inverter is used.

The output of the inverter is given in the table 4.1

Gate 1	Gate 2	Gate 3	Va	Vb	Vc	Vab	Vbc	Vca	Vas	Vbs	Vcs
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	Vdc	0	-Vdc	Vdc	-Vdc/3	-Vdc/3	(2*Vdc)/3
0	1	0	0	Vdc	0	-Vdc	Vdc	0	-Vdc/3	(2*Vdc)/3	-Vdc/3
0	1	1	0	Vdc	Vdc	-Vdc	0	Vdc	-(2*Vdc)/3	Vdc/3	Vdc/3
1	0	0	Vdc	0	0	Vdc	0	-Vdc	(2*Vdc)/3	-Vdc/3	-Vdc/3
1	0	1	Vdc	0	Vdc	Vdc	-Vdc	0	Vdc/3	-(2*Vdc)/3	Vdc/3
1	1	0	Vdc	Vdc	0	0	Vdc	-Vdc	Vdc/3	Vdc/3	-(2*Vdc)/3
1	1	1	0	0	0	0	0	0	0	0	0

Table 4-1 Inverter Switching states, Inverter output Voltages and Machine Voltages

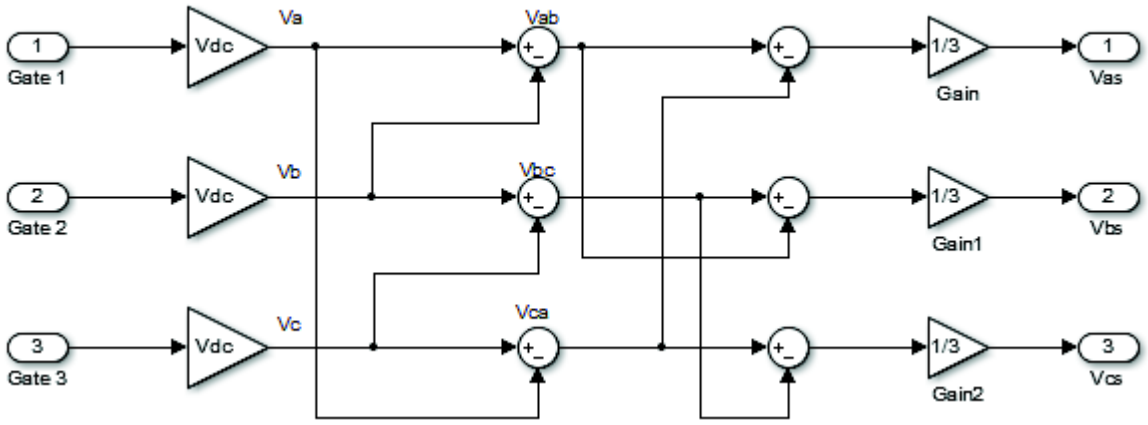

Figure 4.10 Mathematical Model of a 3 phase inverter

4.4 The Neural Network Switching Table

A multi-layer perceptron neural network has been used. This network has one input layer, two hidden layers and one output layer. The switching table discussed in Table 3.5 are

given as the inputs and targets for the training of the neural network. The neural network is implemented using the Neural Network Tool box in Simulink®. The Matlab® file for the switching table is given in M-File for the Switching Table (Table4.1)

4.4.1 Training the Multi-layer Perceptron Neural Network

In Simulink®, the neural network tool box can be started by using "nnstart" command.

Once the command window opens, the inputs and the required targets form the switching table developed earlier as shown in Table 3.5 are given as inputs. The required gate signals to the inverter is given as the required target to the neural network.

The neural network curve fitting tool has been used to achieve the switching table.

The network is trained using Levenberg-Marquardt back-propagation algorithm as described in Section 4.3.4. Care should be taken to make sure that there is enough memory available.

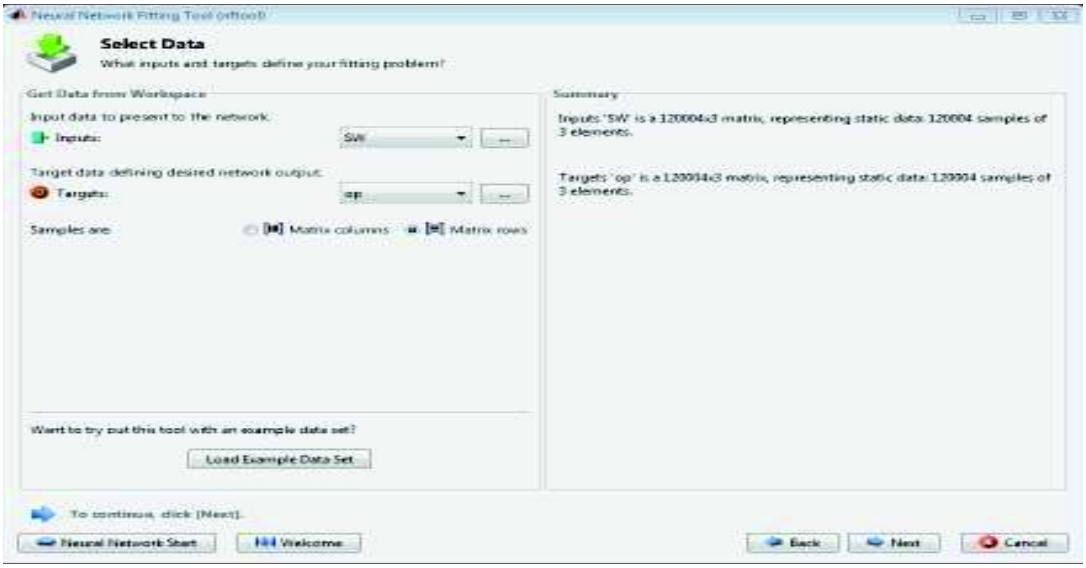

Figure 4.11 Input Data and the Target Data Selection

For best accuracy during training, it is important to validate the data as close to the desired output. Also care must be take not to over train the network.

Figure 4.12 Setting Limits for curve fitting

Based on the data given, the neural network training tool, trains the network. The Matlab file for training is given in M-file for the Neural Network

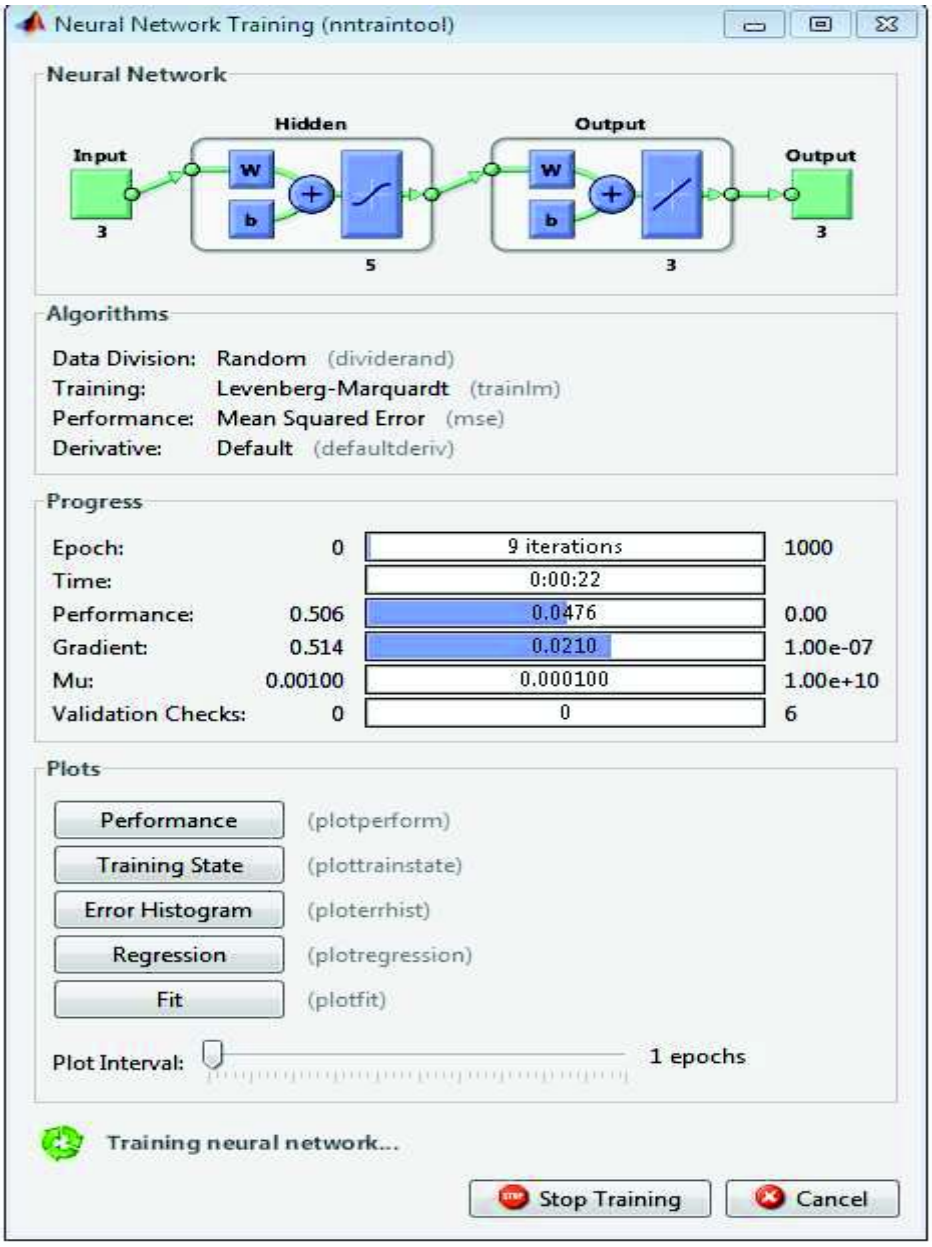

Figure 4.13 Neural Network Training Tool in Simulink®

The neural network training tool in Matlab/Simulink® simplifies the implementation of the training algorithm. It also eliminates the need to calculate the standard deviation in this case.

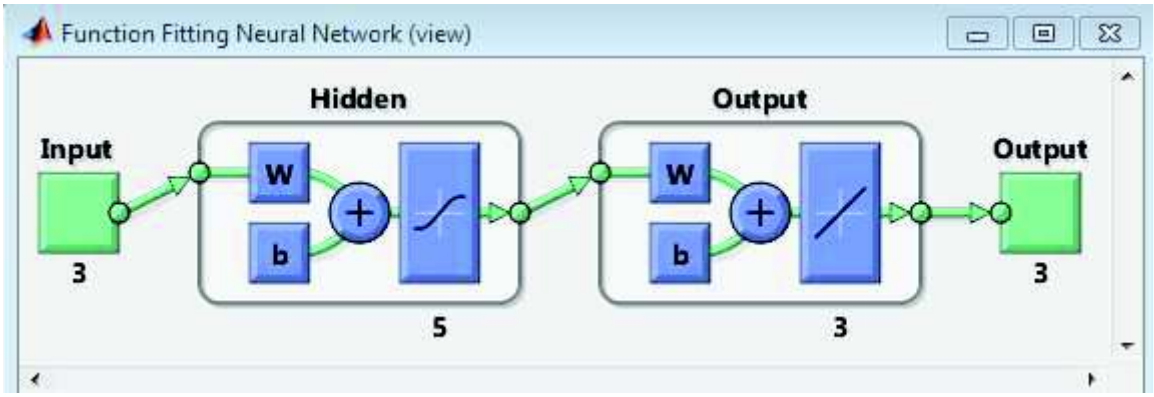

Figure 4.14 The Top level neural network in Simulink®

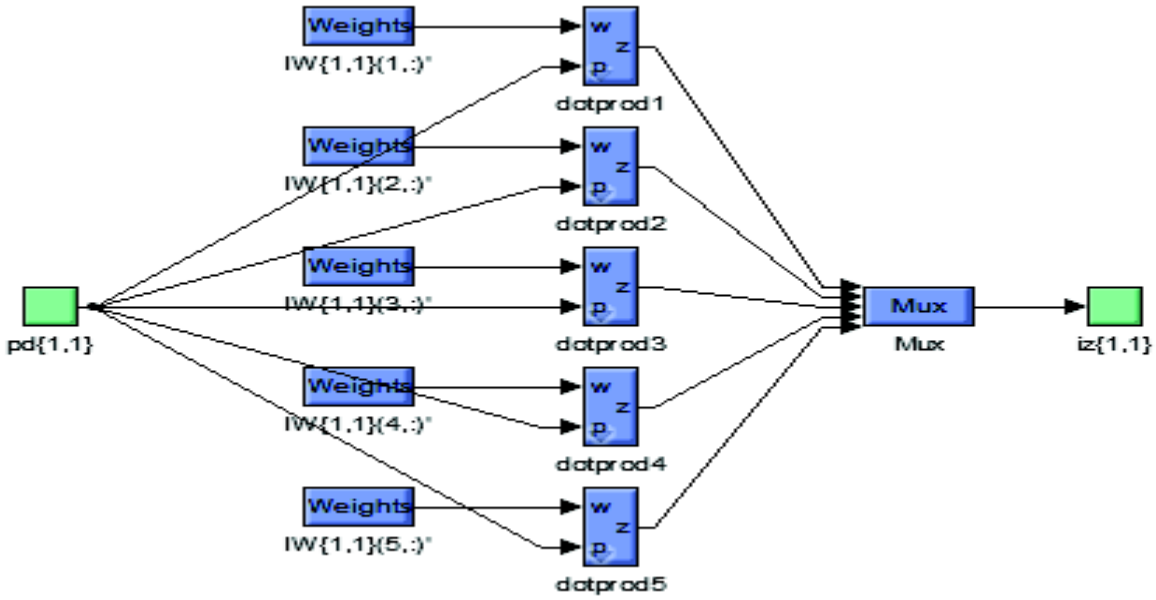

Figure 4.15 Construction of the Hidden layer of the neural network

5.Results

5.1 Training Outcomes of the Neural Network

5.1.1 Training Performance Plot

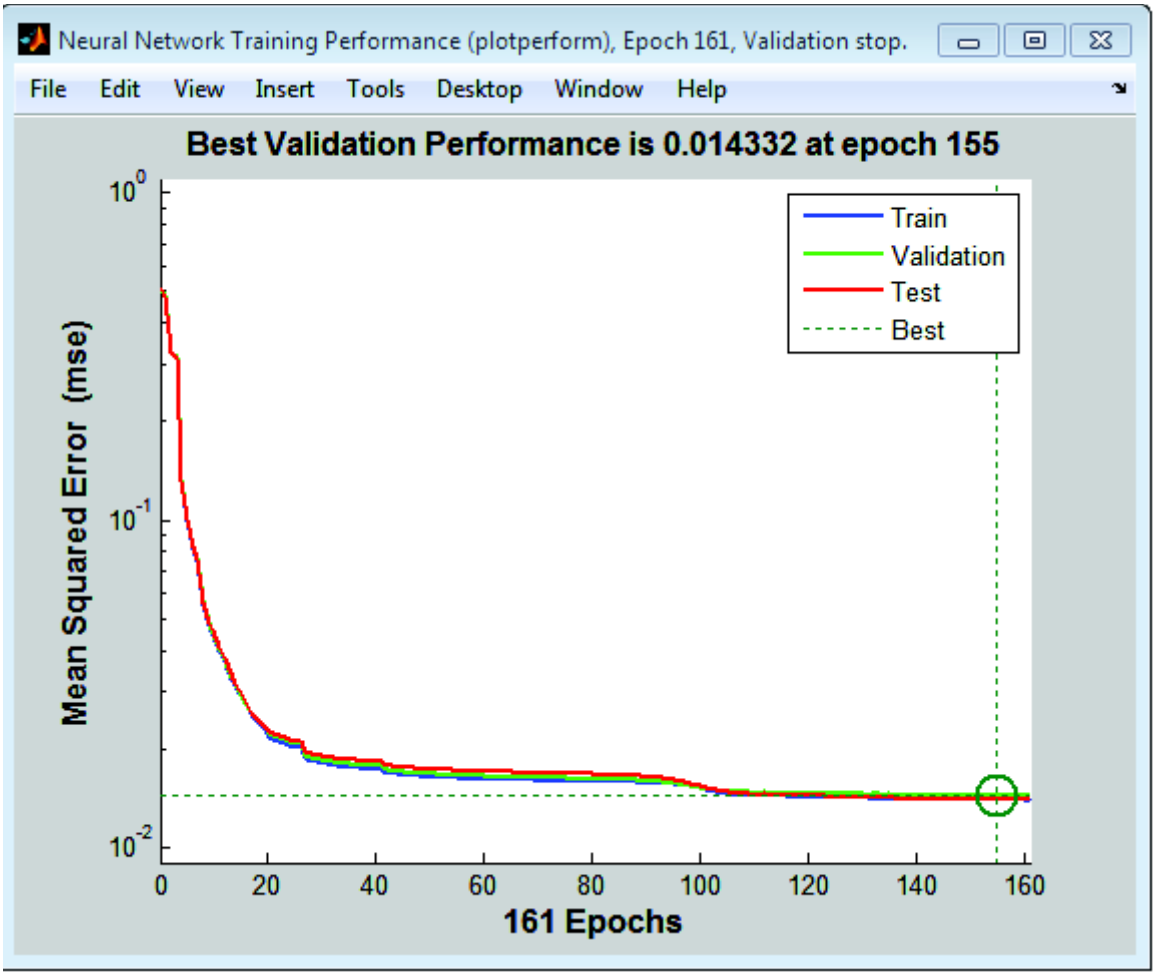

Figure 5.1 Training, validation and Test performance of the Neural Network

The graph represents the Training, validation and Test performance of the neural network during training. From the graph, it can be seen that the best validation performance is at 155 epoch with an error of 0.014332.

5.1.2 Error Histogram

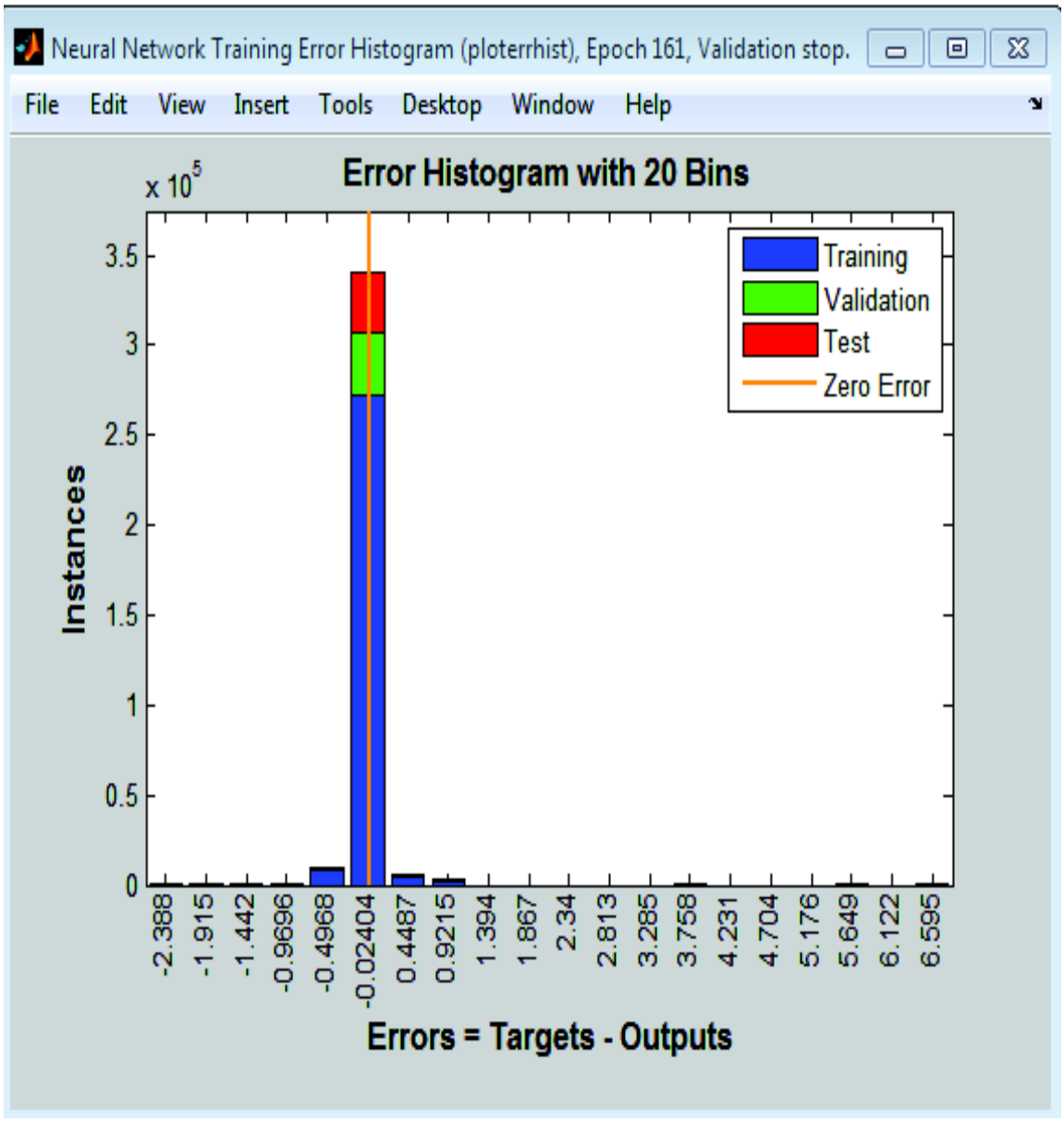

Figure 5.2 Error Histogram of the Neural Network post training

Blue indicates the training data, the green indicates validation data and the red indicates test data. The histogram indicates the data points that have a significantly poorer fit than the majority of the data. In this case, the majority of the errors fall between -0.49 and 1.

5.1.3 Regression Plots

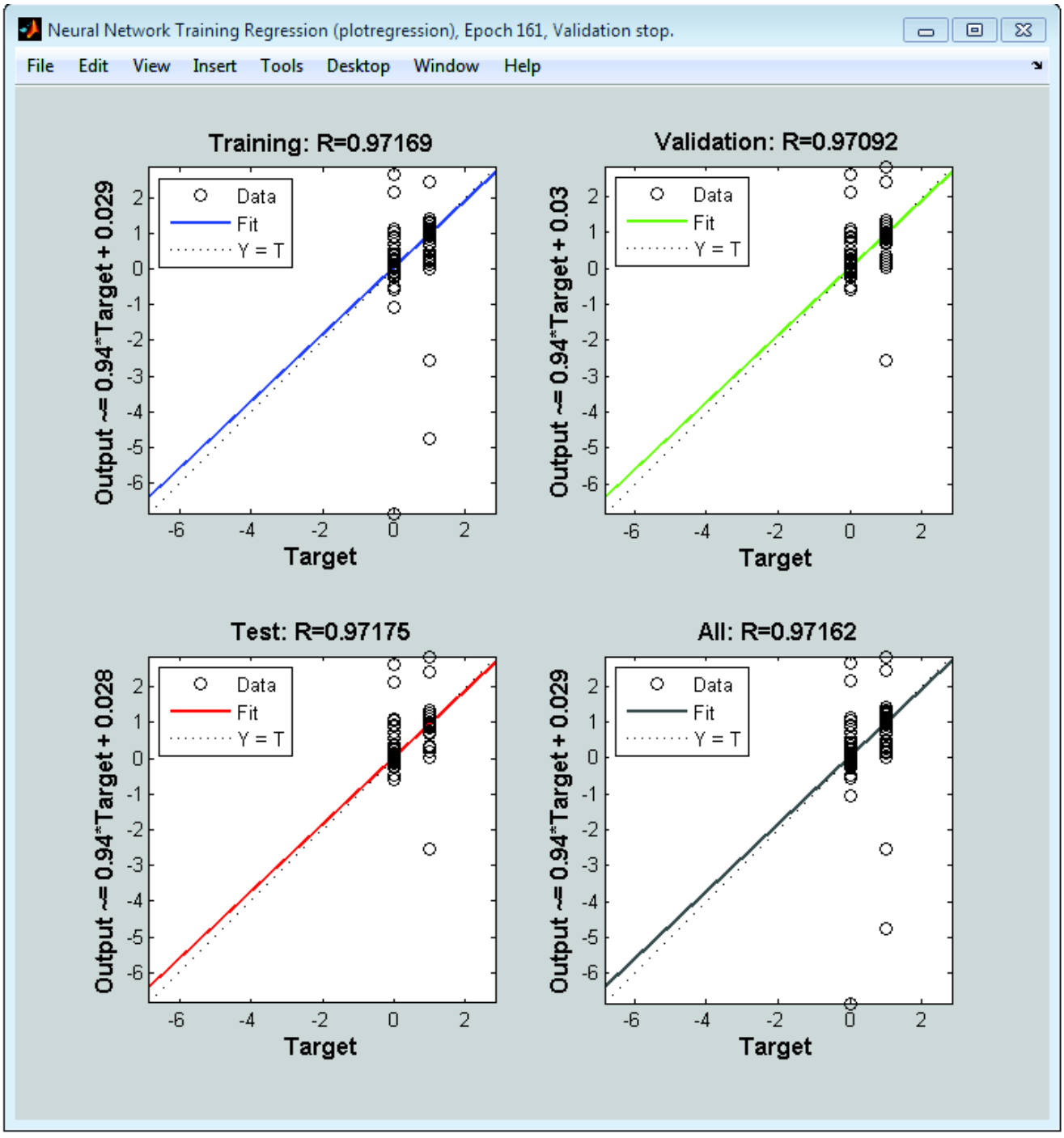

Figure 5.3 Regression Plots of the Training, Validation, Testing and over all Convergence

The Regression Values measure the relationship between the output and targets. The regression value closer to 1 represents a close correlation, (more accuracy) while a zero indicates random correlation.

5.2 Results for DTC System

5.2.1 Speed Response

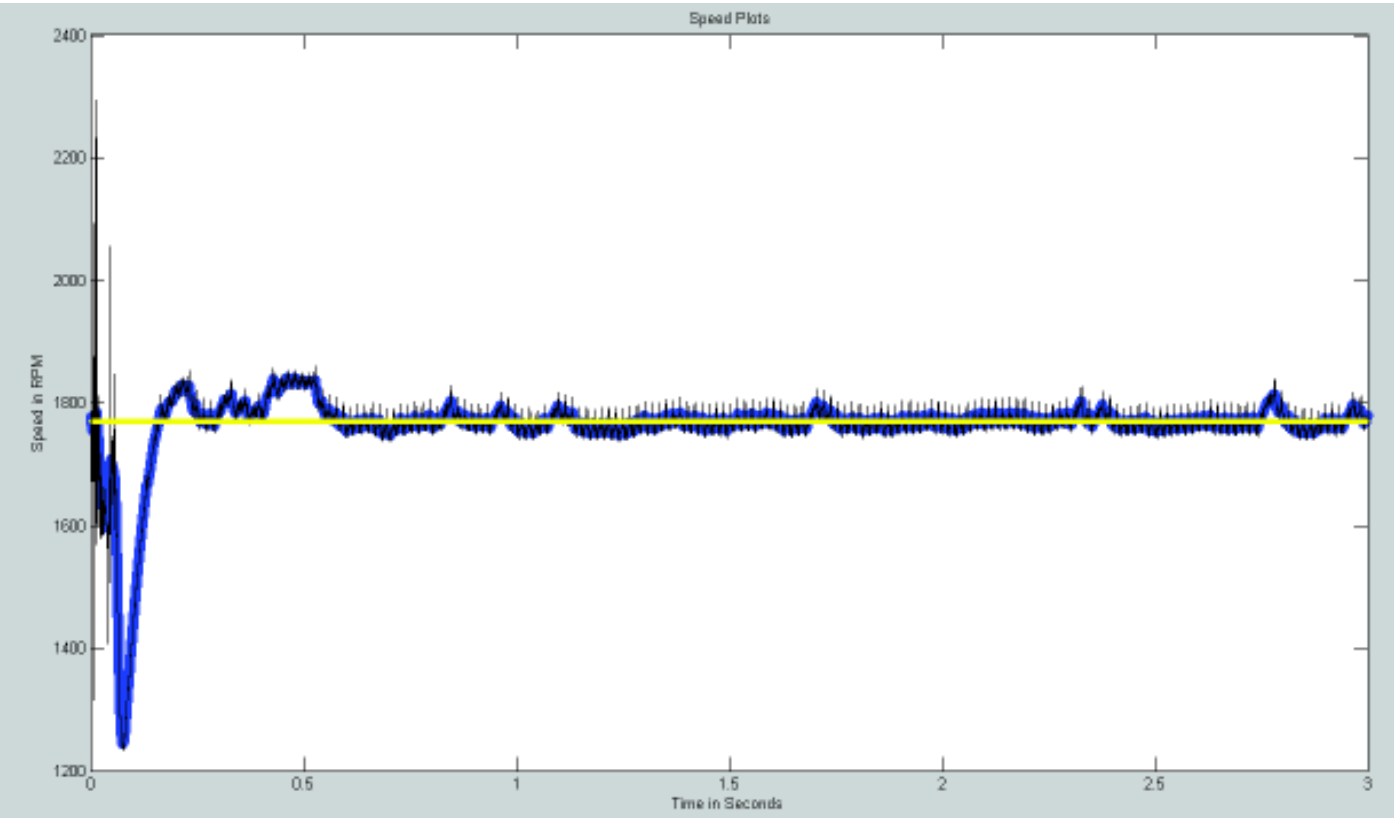

Figure 5.4 Speed Response curves;

(Blue)Actual mechanical Speed of the Induction Machine in RPM, (Black)Estimated Mechanical Speed in RPM, (Yellow) Reference Mechanical Speed in RPM

The Speed response of the system is show in the figure 5.5. It can be see that the estimated speed converges to the steady state value rather quickly. The bumps along the graph of the estimated mechanical speed occur when there is a sudden change in the load torque.

5.2.2 Torque Response

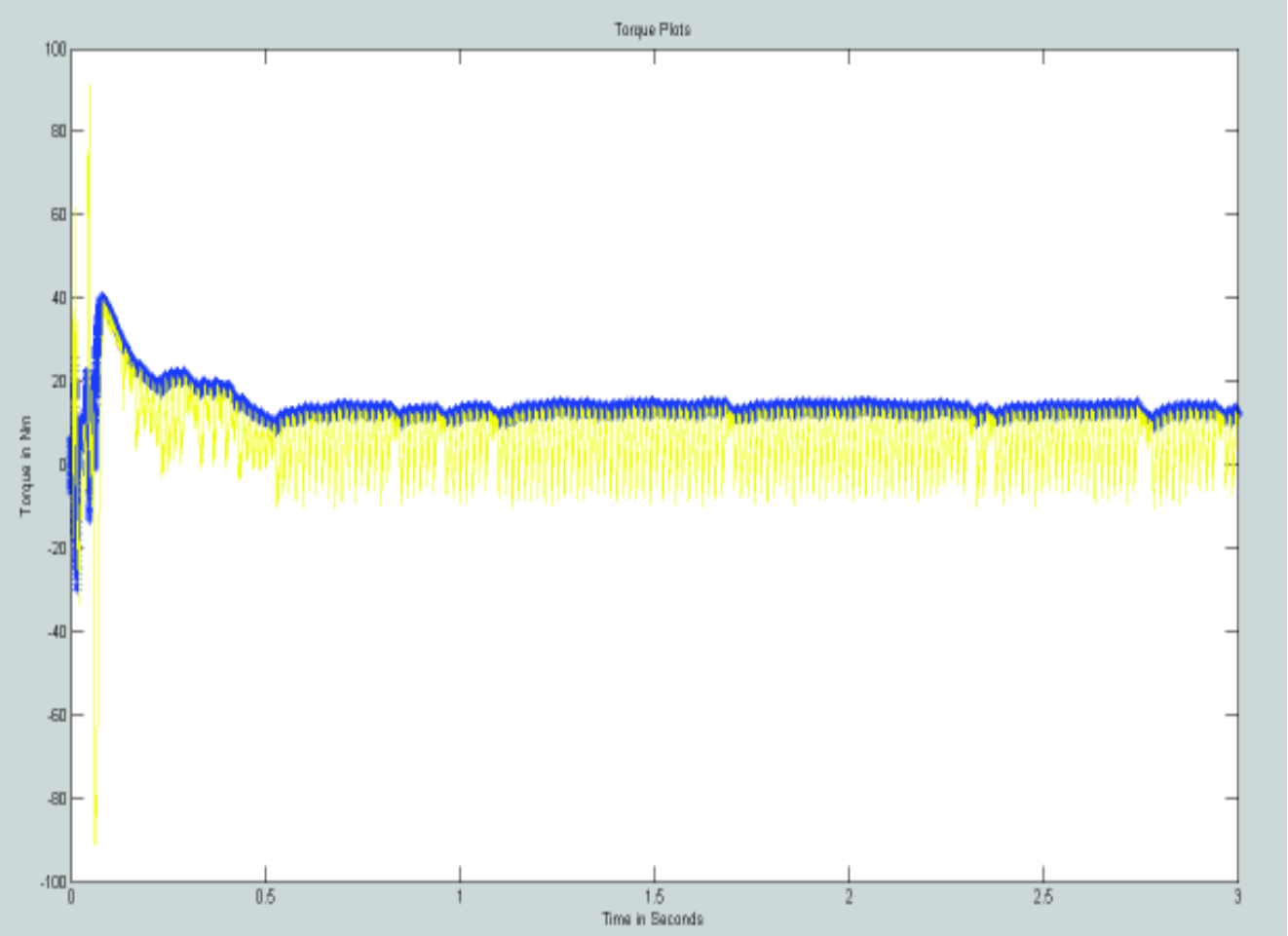

Figure 5.5 Torque Response; (Blue) Reference Torque in Nm, (Yellow) Estimated Torque in Nm

The Torque response of the system to its steady state value is fast. Due to the constant switching a significant amount of ripples can be seen.

5.2.3 Stator Flux Response

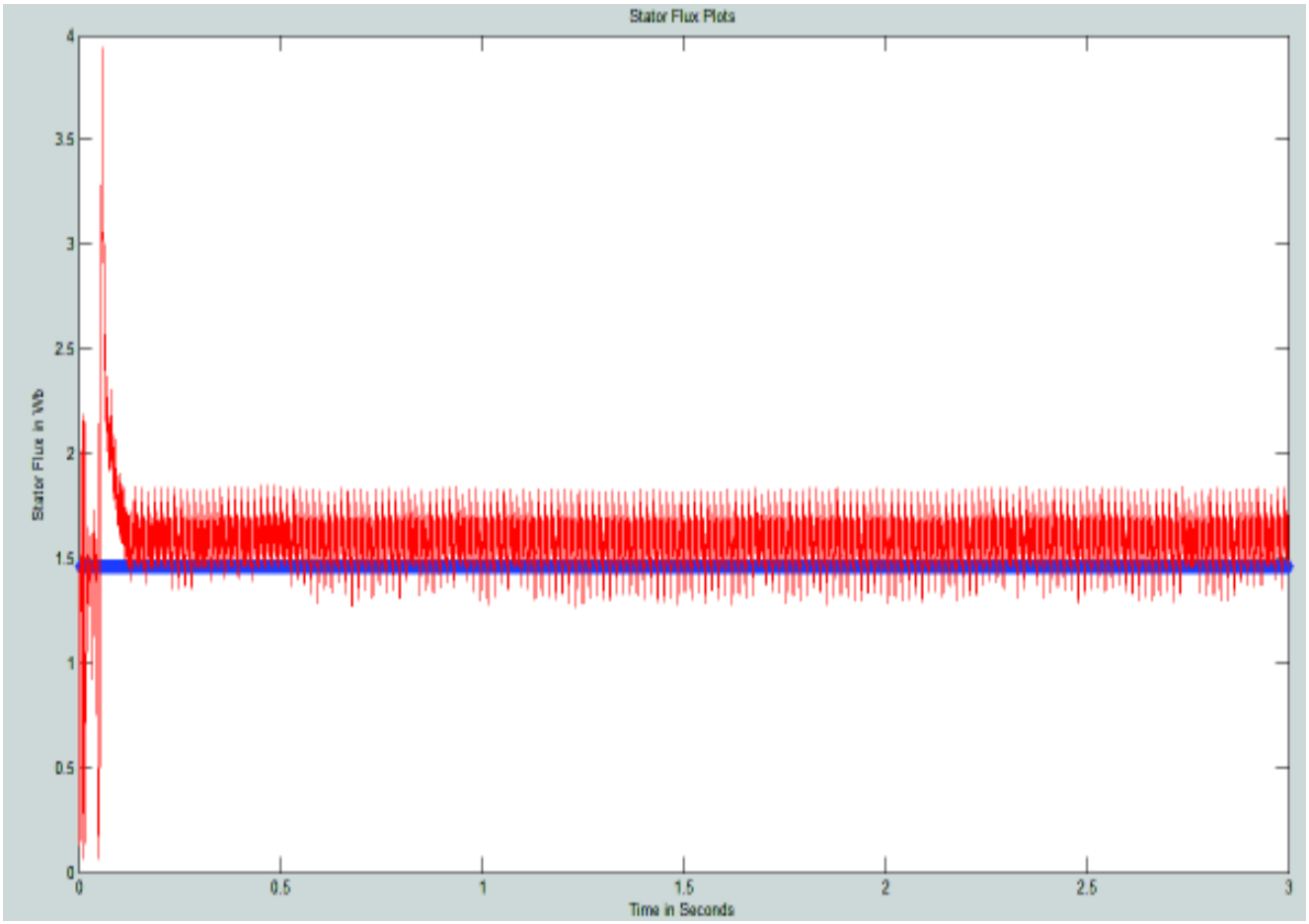

Figure 5.6 Flux Response; (Blue),Reference Flux in Wb, (red)Estimated Stator Flux in Wb

Keeping in mind that the stator flux reacts faster to the change in stator voltage, it can be seen that, the flux reaches steady state at approximately at 0.05 ms. Due to the virtue of constant switching the ripples are reflected.

Figure 5.8, shows the stator and the rotor flux paths. It can be seen that the stator trajectory is hexagonal along the sectors that are 60° apart.

5.2.4 Stator and Rotor Flux Paths

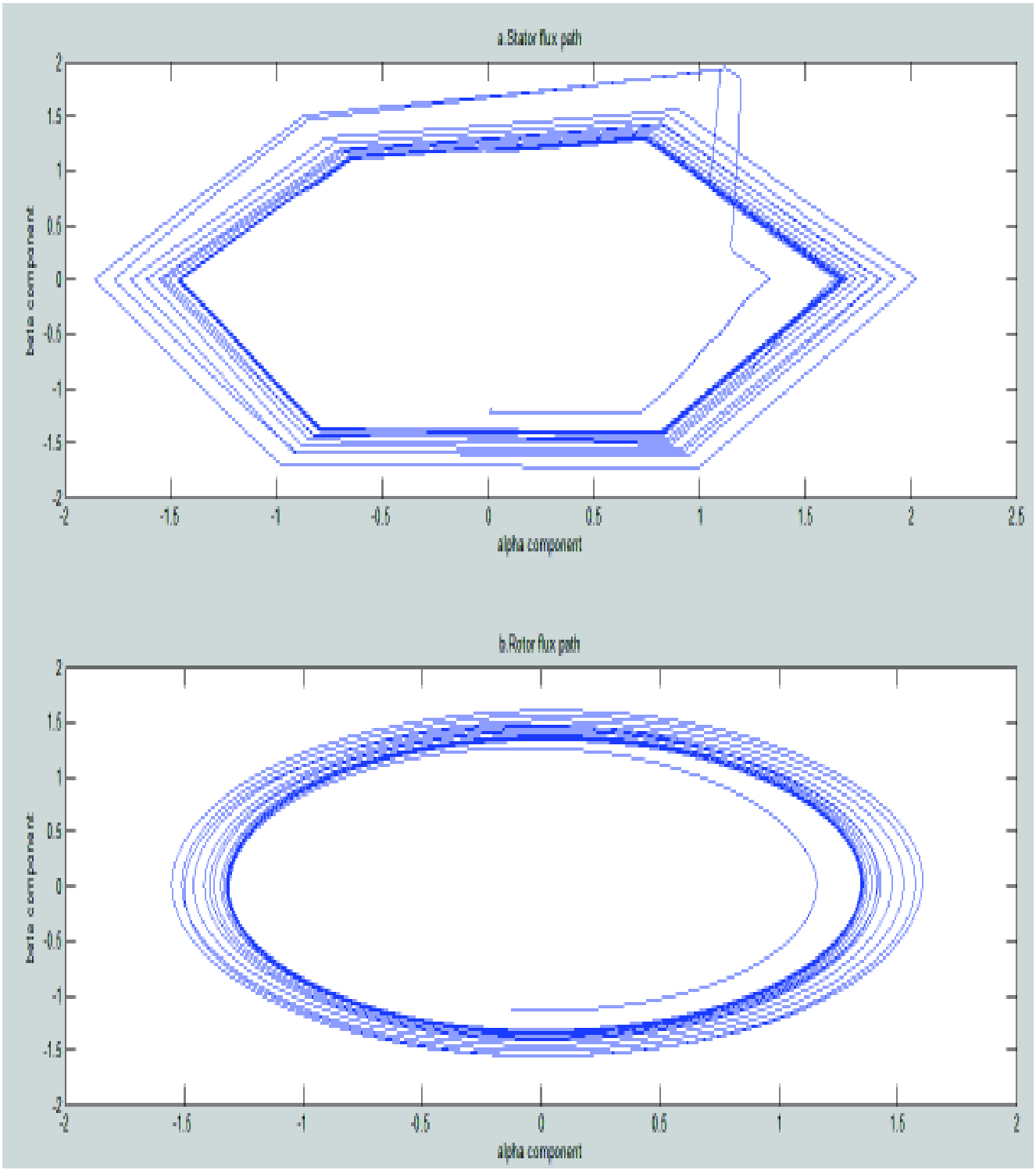

Figure 5.7 a. Stator Flux Path, b. Rotor Flux Path

5.2.5 Inverter Output Voltages

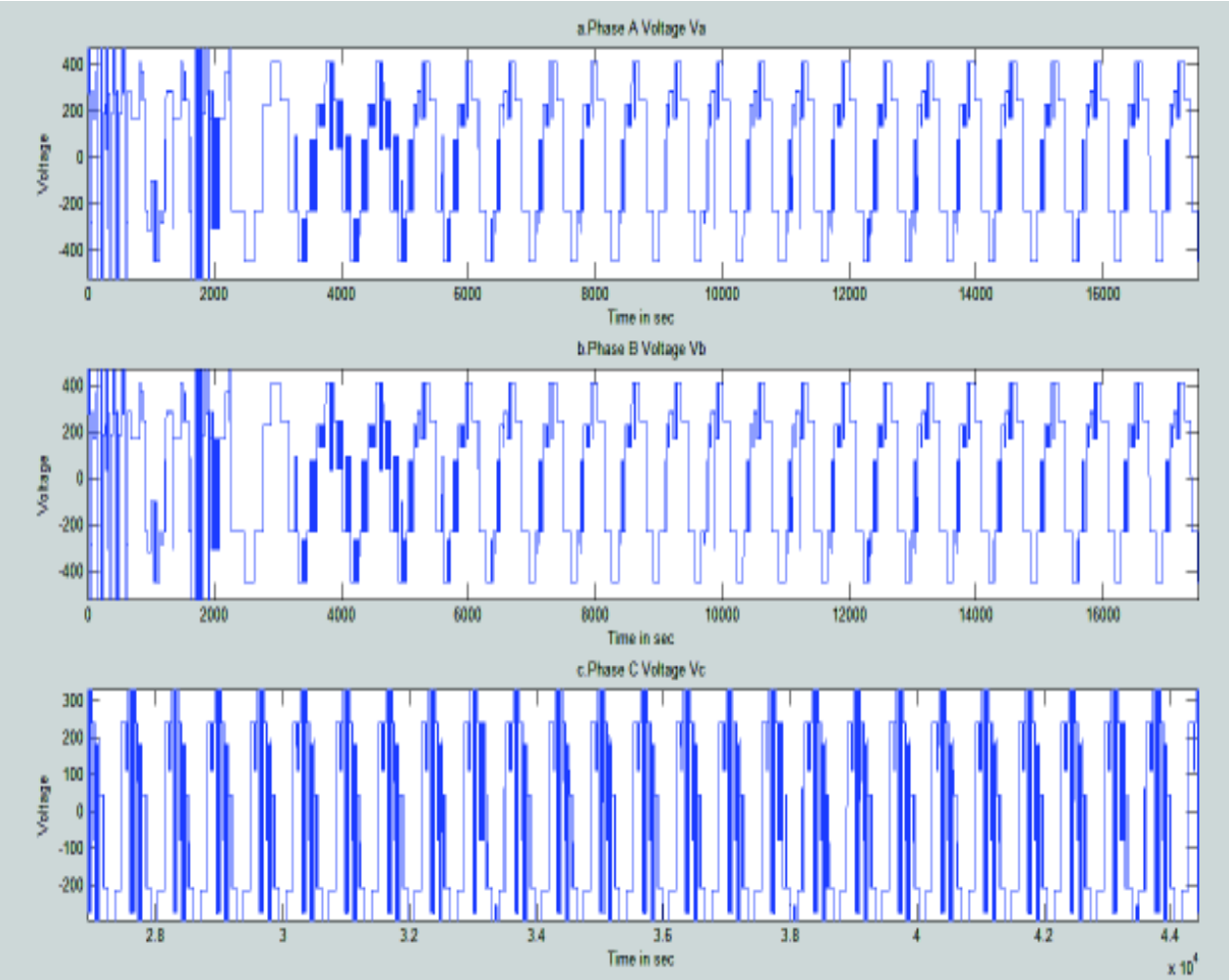

Figure 5.8 Output Voltages of the Inverter in Volts

The inverter output voltages follow the equations of the inverter voltage outputs as indicated by equations 2.2, 2.3 and 2.4

5.2.6 Stator Currents

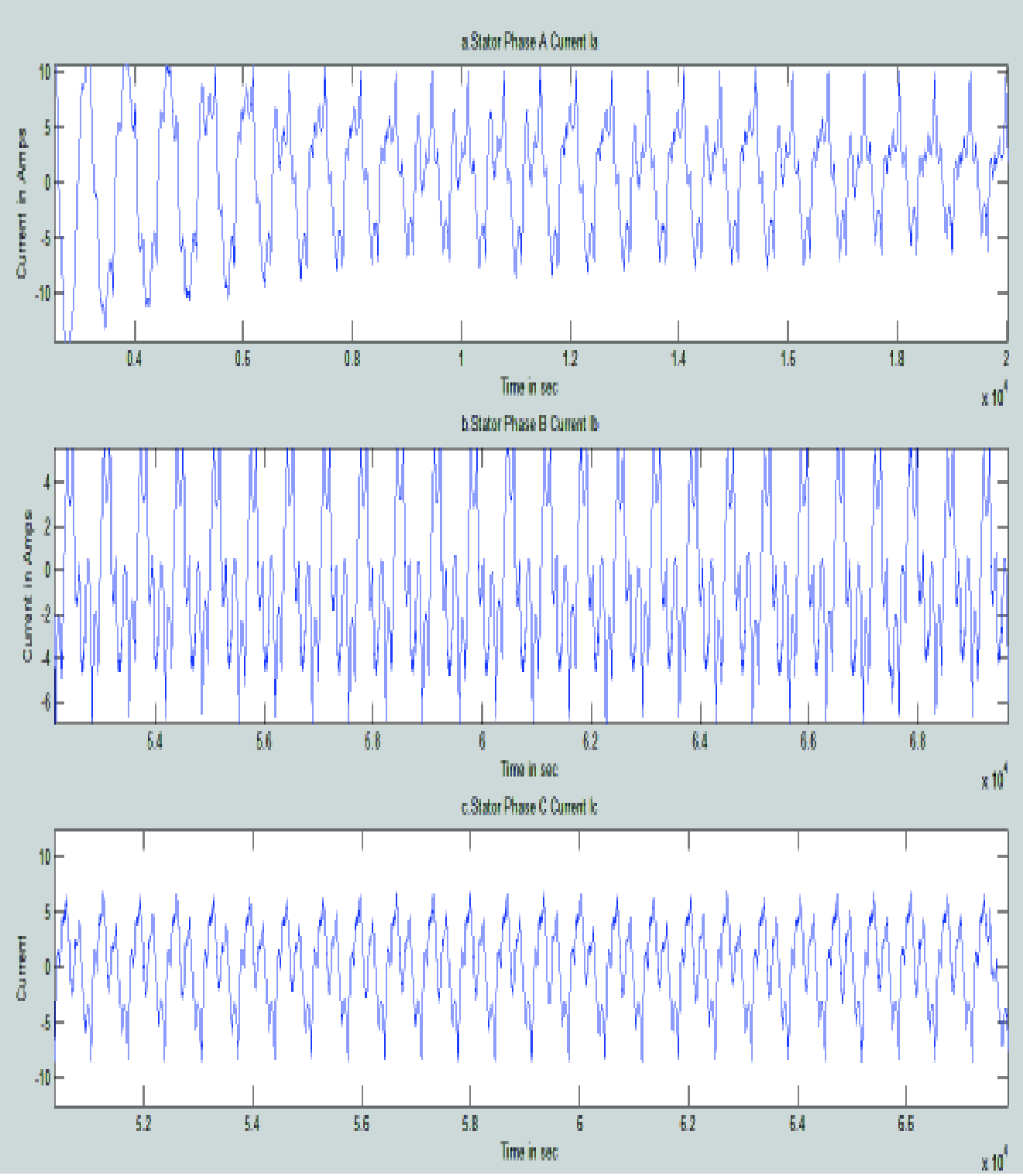

Figure 5.9 Stator Currents of each of the Phases

5.3 Conclusions and future work:

In this paper, a Simulink Model of Direct Torque Control of an Induction Motor was constructed and simulated. An improved flux and Torque Response has been achieved.

The performance of the system has been tested by simulations under different load conditions.

The Neural network is trained has been developed in correlation to the switching table proposed.

The Main Improvements shown are

1. The sector changes do not cause stator Flux drop (Hexagonal Trajectory of Stator Flux)
2. In, Transient state, the Stator flux response is faster
3. In, Transient state, the torque response if faster.

Though the MLPNN yielded faster speed and torque response, it is seen that there is significant ripples. One obvious effort in the future is to try and work on the reduction of the ripples in the torque and stator flux responses.

Multi-level inverters have been known to reduce harmonic distortion. Also multi-level inverters have more switching states. Effort in the future will be to try and incorporate the multilevel inverter for the DTC scheme.

Appendix A:

Induction Machine Parameters [3]

Power	2400 KW
Voltage	460 V
Frequency	60 Hz
Phases	3
Full-Load Current	4A
Full Load Speed	1750 RPM
Full Load Efficiency	88.5%
Power Factor	0.8
Number of Poles	4
Stator Resistance (Rs)	1.77 Ω
Rotor Resistance (Rr)	1.34 Ω
Stator Self Inductance L_{ls}	13.92mH
Rotor Self Inductance L_{lr}	12.12mH
Mutual Inductance L_m	368.7mH
Full Load Slip (S)	1.72%
Total System Inertia	0.025kgm^2

Appendix B:

M-File for the calculation of ΔTem and $\Delta\lambda$s

```
% Induction Motor Parameters
Rs=1.77;
Rr=1.34;
Xls=13.92;%mH
Xlr=12.12;%mH
Xm=368.7;%mH
Jeq=0.025;
p=4;
f=60; VLLrms= 460; s=(1.72/100);
Ws=2*pi*60;
Wd=Ws;
Wslip= s *Ws;
Wm=(1-s)*Ws;


Va = VLLrms * sqrt(2/3);
Vs = (3/2) * Va;
Vsang = angle(Vs);
Vsd = (2/3)^0.5 * abs(Vs) * cos(Vsang)
Vsq = (2/3)^0.5 * abs(Vs) * sin(Vsang)


A = [Rs -Wd*Ls 0 -Wd*Lm;...
     Wd*Ls Rs Wd*Lm 0;...
     0 -Wslip*Lm Rr -Wslip*Lr;...
     Wslip*Lm 0 Wslip*Lr Rr];
Ainv = inv(A);
Isdq=inv(A)*Vsdq;
Irdq=inv(A)*(Vrdq);

Tem = (p/2) * Lm * ((Isq* Irq) - (Isd* Irq));
Tem_Hys=(1/100)*Tem;

M = [Ls 0  Lm 0 ;0  Ls 0 Lm; Lm 0  Lr 0; 0  Lm 0  Lr];
Fsdq=M*Isdq
Fsdq_Hys=(0.5/100)*Fsdq;
```

Appendix C:

M-File for the Switching Table

```
function [a,b,c] = fcn(sector,flux,Torque)
a=0;
b=0;
c=0;

%for Sector 1
if sector==1 && flux==1 && Torque==1
    a=0;b=1;c=1;  %V3
elseif sector==1 && flux==1 && Torque==0
    a=1;b=1;c=1;  %V7
elseif sector==1 && flux==1 && Torque==-1
    a=1;b=0;c=1;  %V5
elseif sector==1 && flux==0 && Torque==1
    a=0;b=1;c=0;  %V2
elseif  sector==1 && flux==0 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==1 && flux==0 && Torque==-1
    a=1;b=0;c=0;  %V4
end

    % for sector 2
    if sector==2 && flux==1 && Torque==1
    a=0;b=1;c=0;  %V2
elseif sector==2 && flux==1 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==2 && flux==1 && Torque==-1
    a=0;b=0;c=1; %V1
elseif sector==2 && flux==0 && Torque==1
    a=1;b=1;c=0;  %V6
elseif  sector==2 && flux==0 && Torque==0
    a=1;b=1;c=1;  % V7
elseif sector==2 && flux==0 && Torque==-1
    a=1;b=0;c=1; % V5
    end

      % for sector 3
    if sector==3 && flux==1 && Torque==1
    a=1;b=1;c=0;  %V6
elseif sector==3 && flux==1 && Torque==0
    a=1;b=1;c=1;  %V7
elseif sector==3 && flux==1 && Torque==-1
    a=0;b=1;c=1;  %V3
elseif sector==3 && flux==0 && Torque==1
    a=1;b=0;c=0;  %V4
elseif  sector==3 && flux==0 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==3 && flux==0 && Torque==-1
    a=0;b=0;c=1;  %V1
```

```
    end


     % for sector 4
    if sector==4 && flux==1 && Torque==1
    a=1;b=0;c=0;  %V4
elseif sector==4 && flux==1 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==4 && flux==1 && Torque==-1
    a=0;b=1;c=0;  %V2
elseif sector==4 && flux==0 && Torque==1
    a=1;b=0;c=1;  %V5
elseif  sector==4 && flux==0 && Torque==0
    a=1;b=1;c=1;  %V7
elseif sector==4 && flux==0 && Torque==-1
    a=0;b=1;c=1;  %V3
    end


      % for sector 5
    if sector==5 && flux==1 && Torque==1
    a=1;b=0;c=1;  %V5
elseif sector==5 && flux==1 && Torque==0
    a=1;b=1;c=1;  %V7
elseif sector==5 && flux==1 && Torque==-1
    a=1;b=1;c=0;  %V6
elseif sector==5 && flux==0 && Torque==1
    a=0;b=0;c=1;  %V1
elseif  sector==5 && flux==0 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==5 && flux==0 && Torque==-1
    a=0;b=1;c=0;  %V2
    end


    % for sector 6
    if sector==6 && flux==1 && Torque==1
    a=0;b=0;c=1;  %V1
elseif sector==6 && flux==1 && Torque==0
    a=0;b=0;c=0;  %V0
elseif sector==6 && flux==1 && Torque==-1
    a=1;b=0;c=0;  %V4
elseif sector==6 && flux==0 && Torque==1
    a=0;b=1;c=1;  %V3
elseif  sector==6 && flux==0 && Torque==0
    a=1;b=1;c=1;  %V7
elseif sector==6 && flux==0 && Torque==-1
    a=1;b=1;c=0;  %V6
    end
```

Appendix D:

M-file for the Neural Network

```
inputs = SW';
targets = op';
% Create a Fitting Network
hiddenLayerSize = 5;
net = fitnet(hiddenLayerSize);
% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};
% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 10/100;
% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};
% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
% View the Network
view(net)
```

Bibliography

[1] T. N. Takahashi, "A New Quick Response and High-Effeiency control Statergy of Induction Motor," *IEEE Transcations on Industrial Applications,* vol. 22, no. 5, pp. 820-827, Sept/Oct 1986.

[2] A. M. Trzynadlowski, Control of Induction Motors, Academic Press, 2001.

[3] N. Mohan, Advance Electric Drives,Analysis,control and Modelling using Simulink(R), Mnpere, 2001.

[4] R. Krishnan, Electric Motor Drives, Modelling Analysis and Control, Prentice Hall, 2001.

[5] C. E. Corporation, "Basic Concepts for Neural Networks," 2003. [Online]. Available: http://www.cheshireeng.com/Neuralyst/nnbg.htm.

[6] S. Mathew and B. K. Mathew, "Direct Torque Control of Induction Motor using Fuzzy Logic Controller," *International Journal of Adcanced Research in Electrical, Electronics and Instrumentation Engineering,* vol. 2, no. 1, Dec 2013.

[7] H. N. Koivo, "Neural Network Basics Using MATLAB(R)," Feb 2008.

[8] DTREG, "Multilayer Perceptron Neural Networks," DTREG, [Online]. Available: ww.dtreg.com/mlfn.htm.

[9] O. Kisi and E. Uncuoglu, "Conparission of three back propogation training algorithms for two case studies," *Indian Journal of Engineering and Material Science,* vol. 12, no. 1, pp. 434-442, 2005.

[10] M. Depenbrock, "Direct Self control (DSC) of Inverter-fed induction Machine," *IEEE Transactions on Power Electronics,* vol. 3, no. 4, pp. 420-829, 1998.

[11] V. Kostic, M. Petronijevic, N. Mitrovic and B. Bankovic, ""Experimental Verification of Direct Torque Control Methods for Electric Drive Application," *FACTA UNIVERSITATIS series: Automatioc Control and Robotics,* vol. 8, no. 1, pp. 111-126, 2009.

[12] J. A. Momoh and M. E. El-Hawary, Electric Systems, Dynamics and Stability with Artifical Intelligence, Marcel Dekker, 2000.

[13] Z. Alnasir and A. Almarhoon, "Design of Direct Torque Controller of Induction Motor," *International Journal of Engineering and Technology,* vol. 4, no. 2, Apr/May 2012.

[14] D. Casadei and G. Serra, "Implementation of Direct torque control Algorithm for Induction Motor based on Discreet Space Vector Modulation," *IEEE Transactions on Power Electronics,* vol. 15, no. 4, July 2002.

[15] J.-Q. Yang and J. Huang, "A New Fill-speed Range Direct torque Control statergy for Induction machine," *Proceedings of the third international conference on machine learning and cybernetics,* August 2004.

[16] R. Toufouti, S. Meziane and S. Benalla, "Direct Torue Control of an Induction Motor using Fuzzy Logic," *ICGST Transactions on ACSE,* vol. 6, no. 2, pp. 17-24, 2006.

[17] J.-Q. Yang and J. Huang, "Direct torque control system for Induction Motors with Fuzzy Speed PI Regulator," *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,* 2005.

[18] H.-H. Xaio, S. Li, P.-W. Wan and M.-F. Zhao, "Study on Fuzzy Direct torque Control system," *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,* 2002.

[19] J.-S. Ryu, I.-S. Yoon, K.-S. Lee and S.-C. Hong, "Direct torque Control of Induction Motor using fuzzy variables switching sector," *IEEE International Symposiumn on Industrial Electronics,* vol. 2, no. 2, pp. 901-906, 2001.

[20] S. A. Mir, M. E. Elbuluk and D. S. Zinger, "Fuzzy Implementation of Direct Self Control of Induction Machines," *IEEE Transctions on Industry Applications,* vol. 30, no. 3, pp. 94-129, 1994.

[21] M. Wlas, J. Krzemin`ski, H. Abu-Ru and H. A. Toliyat, "Artifical Neural Network based Sensorless Non-linear Control of induction Motors," *IEEE Transactions on Energy Conversion,* vol. 20, no. 3, 2005.

[22] X. Ma and Z. Na, "Neural Network Speed Identification scheme for speed sensor-less DTC induction motor drive system," *Power Electronics and Motion Control Conference ,* vol. 3, no. 2, pp. 1242-1245, 2000.