

SOFT COMPUTING TECHNIQUES TO CONTROL SYNCHRONOUS MOTOR

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

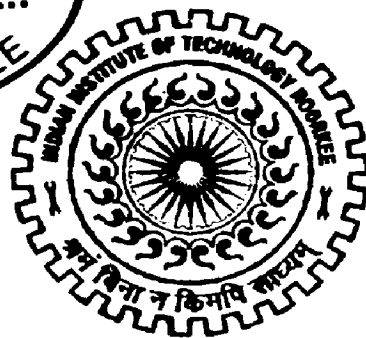
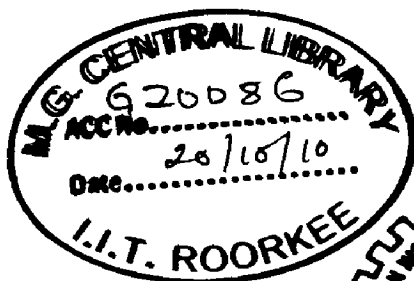
MASTER OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING
(With Specialization in Control and Guidance)

By

VENKATA PRAVEEN BATTULA



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2010**

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this Dissertation entitled "SOFT COMPUTING TECHNIQUES TO CONTROL SYNCHRONOUS MOTOR" submitted for the award of the degree of **Master of Technology** with specialization in **Control & Guidance** in the department of Electronics & Computer Engineering, **Indian Institute of Technology Roorkee**, under the guidance of **Dr. Vijay kumar**, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee.

Date: 28-06-2010

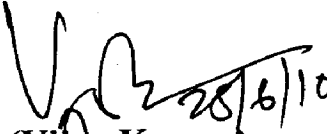
Place: Roorkee

B. V. Praveen

(B.V.PRAVEEN)

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.


(Vijay Kumar)

Department of Electronics & Computer Engineering,

Indian Institute of Technology Roorkee,

Roorkee-247667, India.

ACKNOWLEDGEMENT

I express my foremost and deepest gratitude to **Dr.Vijay Kumar, Associate Professor**, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee, Roorkee for his valuable guidance, support and motivation throughout this work. The valuable hours of discussion and suggestions that I had with him have undoubtedly helped in supplementing my thoughts in the right directions for attaining the desired objective. I consider myself extremely fortunate for having got the opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to all faculty members of Control & Guidance for their constant encouragement, caring words, constructive criticism and suggestions towards the successful completion of this work.

I would like to thank the Head Of the Department for providing lab facilities and the lab technicians for their help in using the lab equipment.

Last but not the least, I am highly indebted to my parents, friends and family members, whose sincere prayers, best wishes, moral support and encouragement have a constant source of assurance, guidance, strength and inspiration to me.

(B.V.PRAVEEN)

ABSTRACT

Permanent Magnet Synchronous Motor Drives are highly efficient, high speed drives which are used extensively in many applications. The main aim of this thesis is to develop an intelligent controller for the speed control of a permanent magnet synchronous motor(PMSM), which reduces the uncertainties and effect of load disturbances while maintaining the performance with respect to speed, torque and stator current. In order to achieve this level of intelligence, this thesis investigates how to unify and hybridize many softcomputing techniques including Fuzzy logic, Neurocomputing, Genetic algorithms. First learning capabilities of neurocomputing are explored in interaction with environment and knowledge acquisition. An adaptive Neural Network based Model reference controller is proposed for the control of the parameters of the PMSM. Secondly knowledge tuning of fuzzy logic systems are developed through knowledge-based and Neuro-fuzzy approaches. An Interval Type-2 Fuzzy Logic Controller(FLC) is proposed and adaptive Neuro Fuzzy Based Inference System(ANFIS) for the control of synchronous motor. Thirdly an hybrid approach in the learning of a fuzzy logic system is explored using Particle Swarm Optimization(PSO). These applications are analyzed for the PMSM using digital simulation. The results of the application show the capabilities of the proposed algorithms, i.e., the type-2 FLC and PSO based optimizers in dealing with complex uncertain problems and robust, simple viable and visible solutions offered by soft computing techniques for such nonlinearities.

Contents

Candidate's Declaration	i
Acknowledgements	ii
Abstract	iii
1 Introduction	1
1.1 Statement of Problem	1
1.2 Brief Review of literature	3
1.3 Organization of Dissertation	4
2 Permanent Magnet Synchronous Motor Drive	5
2.1 Vector Control for PMSM	5
2.1.1 Principle of vector Control	5
2.1.2 PMSM Model	8
2.1.3 Speed Field Oriented Control Scheme for the PMSM	9
2.1.4 The Current-Controlled Voltage Source Inverter	10
3 Adaptive Model Reference Neural Network Controller	13
3.1 Neural network architecture	14
3.1.1 Supervised Learning in Neural Networks	16
3.1.2 The Feed Forward Network	18
3.1.3 Back Propagation Training	20
3.1.4 Neural Network Model reference Controller	22
3.1.5 System Identification stage	22
3.2 Model Reference Neural Network Controller in the control of Permanent Magnet Synchronous Motor	24
4 Fuzzy Logic Based Speed Controller	28
4.1 General Introduction	28
4.2 Fundamentals of Fuzzy Logic Control	29
4.2.1 Fuzzification	30

4.2.2	Database and Rules	31
4.2.3	Inferencing	33
4.3	Defuzzification	34
4.4	Control Of Permanent Magnet Synchronous Motor through Fuzzy Logic Control	34
4.4.1	FLC Structure for the PMSM Drive	35
4.4.2	Tuning Procedure of Fuzzy parameters	37
4.5	Introduction to Particle Swarm Optimization Algorithm	37
4.5.1	Overview of Intelligent Optimization Using Stochastic Search	37
4.5.2	Particle Swarm Optimization	39
4.5.3	PSO Algorithm	41
4.5.4	Objective Function	44
4.6	Type-2 Fuzzy Logic Control Approach	44
4.7	Interval Type-2 Fuzzy Set Theory	46
4.7.1	Type-2 Fuzzy Sets	46
4.7.2	Type-2 Fuzzy Logic controller (FLC)	48
4.7.3	Seven Term Fuzzy Logic Controller	49
4.7.4	FLC Inference and Defuzzification	49
4.8	Description of the system	50
4.9	Synchronous Motor control using Type-2 FLC	51
5	Adaptive Networks: Architectures and Learning Algorithms	56
5.1	Adaptive Neuro Fuzzy Inference System	56
5.1.1	The first layer	57
5.1.2	The second layer	58
5.1.3	The third layer	59
5.1.4	The fourth layer	59
5.1.5	The fifth layer	60
5.1.6	Hybrid Learning Algorithm	60
5.2	Modeling of the synchronous motor using the ANFIS	61
6	Results and Discussion	62
6.1	Model Reference Neural Network Controller	62
6.1.1	Fuzzy Controller	64
6.1.2	Anfis Controller	67
6.1.3	Comparison Between Neural,Fuzzy and ANFIS Controller	68
6.1.4	Comparison between Type-1 Fuzzy and Type-2 Fuzzy Logic	68

7 Conclusion	72
7.1 Conclusion and Scope of Future work	72
References	73
A Appendice	79
A.1 Model reference Parameters	79
A.2 Synchronous Motor Parameters	80

Chapter 1

Introduction

1.1 Statement of Problem

High performance drive systems such as Synchronous Motors must provide fast and accurate speed responses, quick recoveries of reference speed from sudden disturbances of all natures and show insensitivity to parameter variations.[1]. Each of the systems presented to date have shortcomings that require remedy if the synchronous motor is to be practically implemented in high performance drive standards. Thus, it is necessary to further develop control algorithms and approaches to produce this high standard of performance in a practical manner.

While the synchronous motor has many advantages[2] over conventional motors, its operation is strongly affected by motor magnetic saliency, saturation and armature reaction effects. Particularly, the saturation of the iron portion of the rotor around the permanent magnets produce distortion of the air-gap flux that affects the reactance parameters of the motor. These reactance changes with different operating conditions and hence, affect the performance of the drive systems if they are accounted for. This makes the control of Synchronous motor for high performance drive applications an engineering challenge.

The objective of this work is to develop and implement a complete Synchronous motor drive systems to be used in high performance drive applications. The vector control scheme, incorporating a speed controller and a current controller is used because it decouples the torque and flux, thus providing faster transient responses and making the control task easier. An efficient speed controller, incorporating heretofore undeveloped methods is presented for the high performance permanent magnet synchronous motor[3].

A fixed gain Proportional Integral (PI)[4], Model reference adaptive Controller[40] all require the accurate and precise knowledge of system model parameters. Moreover, the fixed-gain PI and PID controllers are especially sensitive to parameter variations, load changes and other systems disturbances. Intelligent controllers, such as the fuzzy logic controller (FLC)[16], Type-2

FLC[24], Adaptive Neuro Fuzzy Inference System[43] ANFIS do not need any information about the system mathematical model, are self-adaptive to uncertainties and can handle any kind of system non-linearity. However FLC based drive systems incorporate complex algorithms that impose such computational burdens that they can only be incorporated by making performance compromises or by use of the latest, most powerful personal computer systems. Therefore, a large part of the work is to achieve the control with minimal complexity and computational burden. Apart from fuzzy logic, neural network control is also well suited for control of complex systems where there is an abundance of experimental data. Also using the stand alone fuzzy logic or neural network based controllers, there is a feasibility of combining the two soft computing methods.

As high performance drive systems[3] are becoming more complex, conventional controllers that utilize the mathematical model are insufficient. This resulted in the evolution of soft computing techniques based on biological processes such as learning and evolutionary development. Soft computing techniques based on tools of fuzzy logic, neural networks and genetic algorithm techniques based on particle swarm intelligence are the main techniques in the artificial intelligence and expert systems design. Fuzzy logic reasoning depends on the human experience and expert knowledge. It has the ability to make decisions when the system is with vague and imprecise information. Neural network reasoning depends on the extraction of hidden relationships in given data sets. It has the ability to learn from examples, drawing conclusions based on past experiences. Particle swarm optimization algorithm[33] is essentially an optimization technique based on the ideas of evolution in biological development. It has the ability to systematically obtain solutions in complex problems. Type-2 fuzzy sets are used for modeling uncertainty and imprecision in a better way. Type-2 FLC have grades of membership function that are themselves fuzzy. At each value of a primary variable (Speed, Torque, Current) the membership is a function. The secondary membership function whose domain is the primary membership interval. Hence, the membership function is a three dimensional and Type-2 uses the third dimension that provides new design degrees of freedom for handling the various disturbances and uncertainties.

Soft computing techniques model human intelligence which provides decisions under imprecise and uncertain information. Appropriate combinations of these tools can make them very powerful to tackle ill-defined systems: i.e., systems whose dynamics or working environments are fully/partially known or poorly understood. In fact, several combinations of these techniques have been successfully integrated in a wide range of applications. Integrating soft computing tools with conventional control methods gave birth to an ever growing research area that has become known in the research community as intelligent control.

1.2 Brief Review of literature

Field Oriented Control methods have been sufficiently discussed in the literature and extensively implemented in AC motor drives[1][2][3]. Though there are many advantages of speed sensorless drives using flux observers are obvious, there always exist some difficulties and uncertainties resulting from model inaccuracy, motor parameter variations, and rotor time constant variation.

So extensive research has been concentrated on current control strategies and also presented some comparative studies in this topic[11][12].These strategies can be classified as a linear controller such as PI controller[4]. In [2], it was reported that the transient response of the hysteresis controller was better and its implementation is simple and fast current control can be considered as a simple and efficient strategy applied for inverters to govern the currents of ac motors.

Although there are several novel control methods, such as Fuzzy logic[23], neural networks[37][40][42][44][48], to show some possibilities of replacing PI type control schemes up to now, the most widely used form of feedback control in industrial motor drives is still based on the PI structure[4].To date much research has been done in self tuning control of PI controllers[4].The development of neural networks was inspired by the studies of understanding of the biological nervous system[36][37][44].This thesis paper laid the ground for supervised training using model reference controller[39]. They showed that their technique for adjusting weights could minimize the sum squared error over all patterns in the training set.Despite all the hype, the development of neural networks slowed at the end of 60's and middle 70's due to development of still novel methods using fuzzy logic techniques.[19][22][47] explained uncertain rule based fuzzy logic systems.Fuzzy Logic Controllers(FLC's) have been reported to be successfully used for a number of complex and nonlinear processes. Simple fuzzy controllers show performance that match classical controllers with adaptive characteristics, but are much easier to implement. The first fuzzy controller was introduced by Mamdani[6][7].Fuzzy controllers[41] are linguistic controllers with human logic and sense.But they have no ability to adapt their structure.

Adaptive Neuro Fuzzy controllers like ANFIS by Jang[43] came in 90's which have linguistic approach like fuzzy and adjustable weight systems like neural controllers.ANFIS controller are simple real time controllers which take training data from the system and using fuzzy rule base control the motor.

Genetic algorithms are powerful search techniques though they are time consuming techniques they guarantee global minimum/maximum search.They provide a systematic approach to control optimization and design.Particle swarm intelligence is a type of genetic algorithm technique found by kennedy[31][33]. In this thesis work swarm intelligence is used to control the fuzzy scaling parameters in the control of synchronous motor.

Type-2 fuzzy sets are used for modeling uncertainty and imprecision in a better way.These

Type-2 fuzzy sets were originally presented by Zadeh in 1975 and are essentially "fuzzy fuzzy" sets where the fuzzy degree of membership is a Type-1 fuzzy set[18][30].The new concepts were introduced by Mendel and Liang[16][20][27] allowing the characterization of a Type-2 fuzzy set with a superior membership function and an inferior membership function;these two functions can be represented each one by a Type-I fuzzy set membership function.The interval between these two functions represents the foot print of uncertainty(FOU),which is used to characterize a Type-2 fuzzy set of the inputs and output of PMSM. Type-2 FLC[24] have grades of membership function that are themselves fuzzy.At each value of a primary variable (Speed, Torque,Current) the membership is a function.A new Interval based Type-2 based FLC[21] is introduced in this thesis work which is a simpler type-2 FLC than general type-2 FLCs in the control of synchronous motors.

1.3 Organization of Dissertation

This thesis consists of six chapters. The introduction of vector control techniques for synchronous motor along by describing the statement of problem and contributions have been covered in this chapter.

Chapter 2 provides a nonlinear model of a permanent magnet synchronous motor drive operating employing vector control method for removing the nonlinearity.

Chapter 3 provides a neural network based approach such as Adaptive Model Reference Neural Network Controller in the control of motor.

Chapter 4 provides an Fuzzy Logic Controller based control of Synchronous motor and it provides a way to tune the scaling parameters using the particle swarm optimization.

Chapter 5 describes an Adaptive Neuro Fuzzy Inference System(ANFIS) which combines the characteristics of fuzzy and neural networks to control the synchronous motor.And also a Type-2 Fuzzy Logic Controller is also described which uses fuzzy fuzzy approaches to control the synchronous motor.

Chapter 6 summarizes all the softcomputing techniques by providing the simulation results

Chapter 7 summarizes the whole dissertation. Recommendations for future work are presented.

Chapter 2

Permanent Magnet Synchronous Motor Drive

2.1 Vector Control for PMSM

2.1.1 Principle of vector Control

Vector control makes it possible for ac drives to behave similarly to DC drives with an independent control of flux and torque, and with good steady-state performance and fast dynamic responses. The main idea of vector control is to project electrical variables from a three-phase time and speed dependent non-rotating reference frame into a two-phase time and speed dependent non-rotating reference frame into a two-phase orthogonal time invariant rotating reference frame, and to control the stator currents represented by a vector. Since the control is achieved in field coordinates, vector control is also named as field oriented control.

The vector control for PMSM resolves the stator current vector into two orthogonal coordinate axes of a d-q synchronously rotating rotor reference frame. Three phase stator currents i_a, i_b, i_c are expressed as follows:

$$i_a = I_s \sin(\omega_e(t)) i_b = I_s \sin(\omega_e(t) - \frac{2\pi}{3}) i_c = I_s \sin(\omega_e(t) + \frac{2\pi}{3}) \quad (2.1.1)$$

where I_s is the amplitude of three-phase stator currents. and ω_e is the electrical angular frequency. Taking into account the mechanical 120° angle between the stator coils in space, the stator current vector represented in the 3-phase non rotating frame is defined as in 2.1.2

$$\bar{I}_s = i_a + e^{j\frac{2\pi}{3}} i_b + e^{j\frac{4\pi}{3}} i_c \quad (2.1.2)$$

In figure 2.1.1, it is represented in the three-phase stationary reference frame.

To implement vector control, the stator current vector needs to be transformed into a time

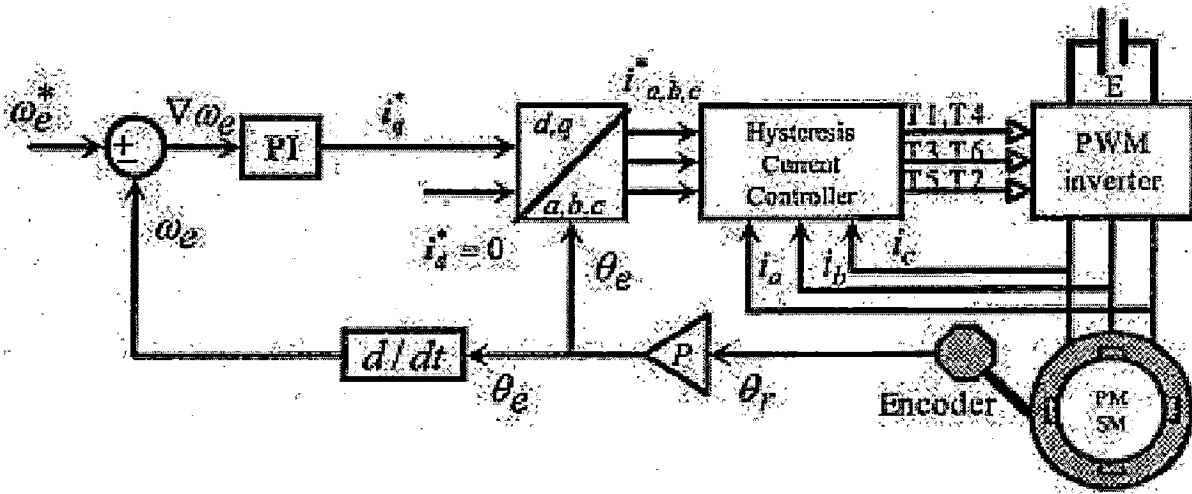


Figure 2.1.1: Three Phase Stationery reference frame

invariant two-coordinate system. Generally, this transformation can be done in two steps using the Clarke transformation and the Park transformation in sequence. Here, for the vector control for PMSM, a transformation of the three-phase variables of stationary reference variables to the d-q synchronously rotating rotor reference frame can be expressed as Eq 2.1.3

$$f_{qd0} = K f_{abc} \quad (2.1.3)$$

where

$$(f_{qd0})^T = [f_q f_d f_0] f_{abc}^T = [f_a f_b f_c] \quad (2.1.4)$$

$$k = 2/3 \begin{pmatrix} -\sin\theta_e & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

In the above equations, the 0-sequence variables as denoted by subscript "0" are independent of θ_e and are normally non-existent; f can represent not only current, but also voltage, flux linkage or electric charge; θ_e is the electrical angle between d-axis of the rotor reference frame and a-axis of the three-phase stationary frame. Furthermore, if rotor flux linkage vector is always aligned with d-axis by the vector control for PMSM, θ_e is also the field angle. It can be derived from equation 2.1.5

$$\theta_e = \omega_e t + \theta_0 \quad (2.1.5)$$

where θ_0 is the initial θ_e at $t=0$. This transformation of the stator current vector can be illustrated in Fig.2.1.2

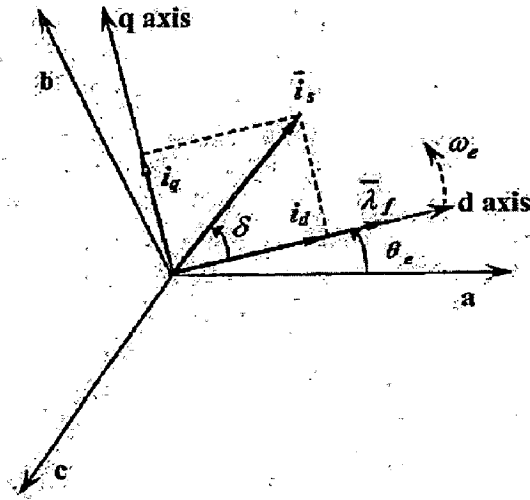


Figure 2.1.2: Stator current vector and its component in d-q reference frame

In the above figure, $\overline{\lambda}_f$ stands for the rotor flux linkage vector. δ is the angle between the rotor field and stator current vector, known as the torque angle, i_q is the torque-producing component of stator current. i_d is the flux-producing component of stator current. They are given by

$$i_q = i_s \sin \delta \quad i_d = i_s \cos \delta \quad (2.1.6)$$

For the PMSM operation at a lower than the base speed, the torque angle can be $0 < \delta \leq 90^\circ$. if $\delta > 90^\circ$, i_d will be negative and the rotor flux linkage will decrease, which is names as flux-weakening. Although the flux-weakening can extend the speed range, it will reduce the torque-to-current ratio. In this project, the examined scope is within the base speed of the PMSM. In vector control, if desired i_q and i_d are already known, an inverse transformation is needed to obtain the stator-phase commands. According to fig. 2.1.2, an inverse transformation is needed to obtain the stator-phase current commands. According to fig. 2.1.2, an inverse transformation of K is expressed as follows:

$$f_{abc} = K^{-1} f_{qdo} \quad (2.1.7)$$

It should be mentioned that the definition of the angle used in the transformation is not unique. Here θ_e is chosen on purpose because it is convenient to realize vector control strategy with the digital simulation using MATLAB.

2.1.2 PMSM Model

The more comprehensive dynamic performance of a synchronous machine can be studied by synchronously rotating d-q frame model known as Park equations. The dynamic model of synchronous motor in d-q frame can be represented by the following equations(1)-(3) [14, 13]:

$$v_{ds} = R_s i_{ds} + \frac{d}{dt} \Phi_{ds} - \omega \Phi_{qs} \quad (2.1.8)$$

$$v_{qs} = R_s i_{qs} + \frac{d}{dt} \Phi_{qs} - \omega \Phi_{ds} \quad (2.1.9)$$

$$v_f = R_s i_f + \frac{d}{dt} \Phi_f \quad (2.1.10)$$

The mechanical equation of synchronous motor can be represented as:

$$J \frac{d}{dt} \Omega = T_e - T_r - B \Omega \quad (2.1.11)$$

Where the electromagnetic torque is given in d-q frame:

$$T_e = p(\Phi_{ds} i_{qs} - \Phi_{qs} i_{ds}) \quad (2.1.12)$$

In which:

$$\Omega = \frac{d}{dt} \theta, \theta = \int \Omega dt, \omega = \frac{d}{dt} \theta_e = p \Omega, \theta_e = p \theta.$$

The flux linkage equations are:

$$\Phi_{ds} = L_{ds} i_{ds} + M_{fd} i_f \quad (2.1.13)$$

$$\Phi_{qs} = L_{qs} i_{qs} \quad (2.1.14)$$

$$\Phi_f = L_f i_f + M_{fd} i_{ds} \quad (2.1.15)$$

Where R_s stator resistance, R_f field resistance, L_{ds} , L_{qs} respectively direct and quadrature stator inductances, L_f field leakage inductance, M_{fd} mutual inductance between inductor and armature, Φ_{ds} and Φ_{qs} respectively direct and quadrature flux, Φ_f field flux, T_e electromagnetic torque, T_r external load disturbance, p pair number of poles, B is the damping coefficient, J is the moment of inertia, ω electrical angular speed of motor. Ω Mechanical angular speed of

motor, θ mechanical rotor position, θ_e electrical rotor position.

2.1.3 Speed Field Oriented Control Scheme for the PMSM

The speed FOC scheme for the PMSM in this thesis is shown in Fig. 2.1.3 In the fig. 2.1.3

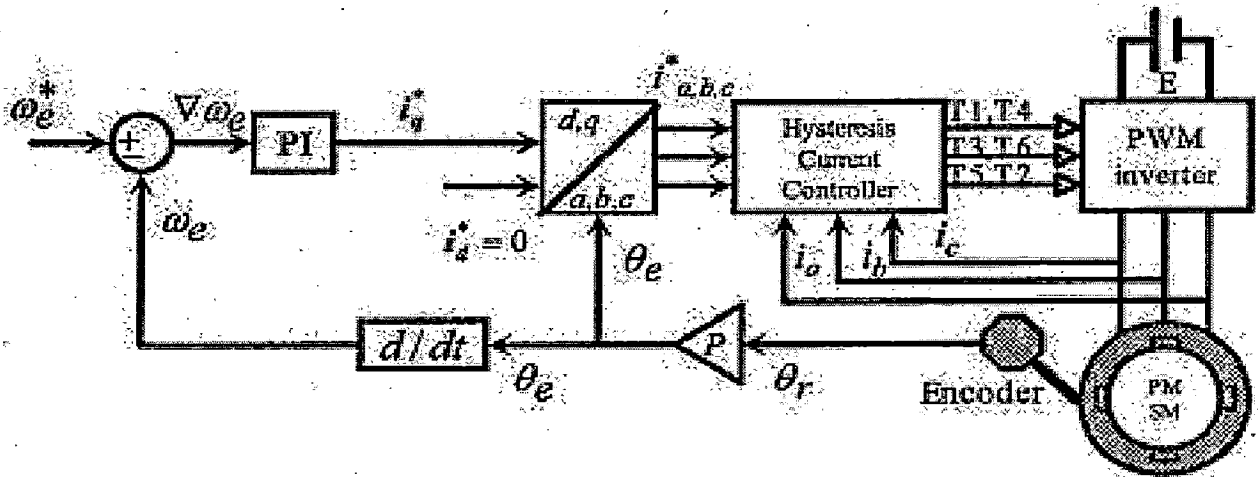


Figure 2.1.3: The Speed Field Oriented Control Scheme

variables with a subscript * stand for the reference variables. The complete system consists of two control loops. The outer loop (Speed loop) is for regulating the motor's speed and the inner loop (Current Loop) is to modulate the stator currents. Using the field-orientation concept, i_d^* is set to zero so that the torque equation becomes linear and the maximization of the torque-to-current ratio is assured. i_q^* serves as the input to the motor torque control. In the hysteresis current controller, stator currents i_a, i_b, i_c are forced to follow the reference currents i_a^*, i_b^*, i_c^* , respectively, within a fixed hysteresis band are set for the motor currents and three hysteresis current loops work independently.

In vector control method [14], an alternative current machine is controlled like a separately excited direct current machine, the principle is to maintain the armature flux and the field flux in an orthogonal or decoupled axis.

For an optimal function with a maximal torque, the simple solution in a synchronous motor is to maintain the direct component of stator current $i_{ds} = 0$, and control the speed by the quadrature component of the stator current i_{qs} . In order to have an optimal functioning, the direct current i_{ds} is maintained equal to zero [13]. Substituting Eqn. 2.1.14, 2.1.15 in Eqn. 2.1.11, the electromagnetic torque can be rewritten for $i_f = \text{constant}$ and $i_{ds} = 0$ as follow:

$$T_e(t) = \lambda i_{qs}(t) \quad (2.1.16)$$

where $\lambda = pM_{fd}i_f$.

In the same conditions, it appears that the v_{ds} and v_{qs} equations are coupled. We have to introduce a decoupling system, by introducing the compensation terms emf_d and emf_q in which

$$emf_d = \omega L_{qs}i_{qs} \quad (2.1.17)$$

$$emf_q = -\omega L_{ds}i_{ds} - \omega M_{af}i_f \quad (2.1.18)$$

2.1.4 The Current-Controlled Voltage Source Inverter

The structure of a typical 3-phase Voltage Source Inverter(VSI) with six switches is shown in Fig. 2.1.4

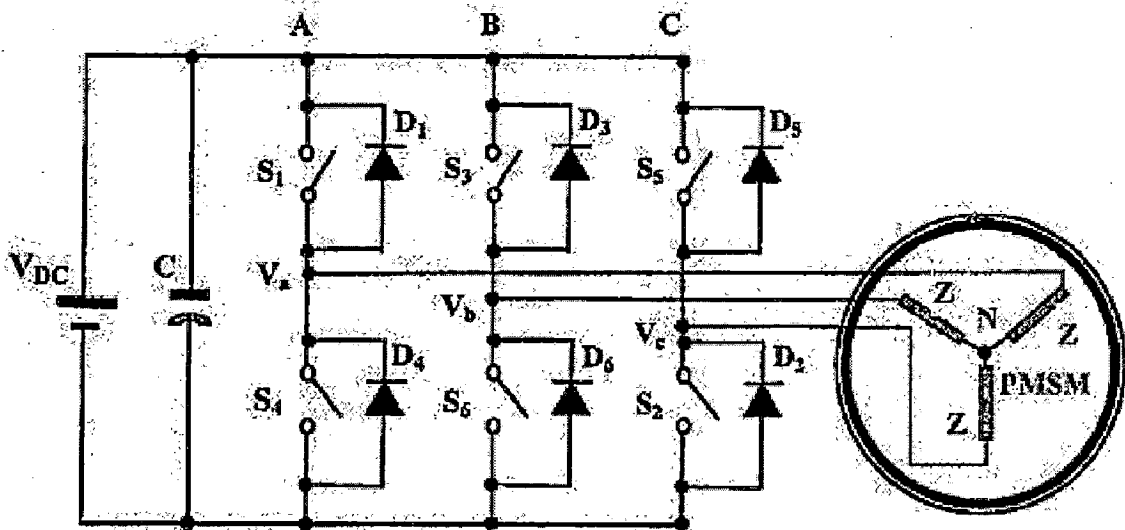


Figure 2.1.4: The Current Controller Voltage Source Inverter

S_1 - S_6 are six power switches, which are IGBTs in this project. V_a , V_b , V_c are the phase voltages applied to the motor windings. Each stator phase of the PMSM is connected to each leg of the VLSI. At any time, three power switches are on and the other power switches are OFF. Also, the upper switch and the lower switch in the same bridge are driven with two complementary firing signals so as to avoid conducting through(short circuit) faults. According to the ON-OFF switching states of power switches, there are eight possible switching combinations, corresponding to eight phase voltage configurations, respectively[12]. The switching patterns of

a three-phase VSI under current control are given in Fig 2.1.5. The voltage vectors correspond-

State order	A	B	C	V_{ab}	V_{bn}	V_{cn}	V_{ab}	V_{bc}	V_{ca}	Operation modes
V_0	0	0	0	0	0	0	0	0	0	Freewheeling
V_4	0	0	1	$-V_{dc}/3$	$-V_{dc}/3$	$2V_{dc}/3$	0	$-V_{dc}$	V_{dc}	Active
V_2	0	1	0	$-V_{dc}/3$	$2V_{dc}/3$	$-V_{dc}/3$	-	V_{dc}	0	Active
V_6	0	1	1	$-2V_{dc}/3$	$V_{dc}/3$	$V_{dc}/3$	-	0	V_{dc}	Active
V_1	1	0	0	$2V_{dc}/3$	$-V_{dc}/3$	$-V_{dc}/3$	V_{dc}	0	-	Active
V_5	1	0	1	$V_{dc}/3$	$-2V_{dc}/3$	$V_{dc}/3$	V_{dc}	$-V_{dc}$	0	Active
V_3	1	1	0	$V_{dc}/3$	$V_{dc}/3$	$-2V_{dc}/3$	0	V_{dc}	-	Active
V_7	1	1	1	0	0	0	0	0	0	Freewheeling

Figure 2.1.5: Switching Patterns of a 3-phase VSI

ing to the active states are shown in Fig 2.1.6. There are six active voltage vectors from V_1 to V_6 . The remaining two vectors, i.e., V_0 to V_7 are the zero voltage vectors, which correspond to the freewheeling states.

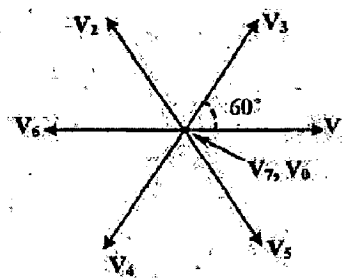


Figure 2.1.6: VSI Voltage Vectors

Current control strategies are important to high-performance drive applications. The efficiency of current controllers determines characteristics of motor drive systems. The hysteresis

current controllers provide fast responses and good accuracy due to their quick actions. Here, a fixed band hysteresis current controller is used to control the VSI. The switching signals are derived from the comparison of the current error with this band. The control target is to limit the current error within a desired band, which is shown in Fig 2.1.7

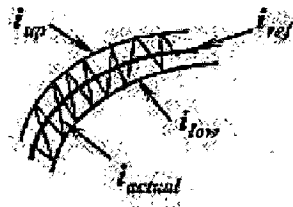


Figure 2.1.7: Fixed Band Hysteresis Current Control

The upper and lower boundaries of the hysteresis band are set for the PMSM stator currents and the hysteresis loops work independently.

Chapter 3

Adaptive Model Reference Neural Network Controller

A neural network is an information-processing system that has been developed as generalizations of mathematical models matching human cognition. They are composed of a large number of highly-interconnected processing units (neurons) that work together to perform a specific task. According to Haykin[44], a neural network is a massively parallel-distributed processor that has a natural propensity for storing experimental knowledge. It resembles the brain in three respects:

1. Knowledge is acquired by the network through a learning process
2. Inter-connected connection strengths known as synaptic Weights are used to store the knowledge
3. Each neuron has an internal state called its activation function (or transfer function) used for classifying vectors.

Neural classification generally comprises of four steps:

1. Pre-processing, e.g., noise suppression, Principal Component Analysis, etc.
2. Training - selection of the particular features which best describe the pattern;
3. Decision - choice of suitable method for comparing the input with the target
4. Assessing the accuracy of the classification. The fundamental model of an artificial neuron that closely matches a biological neuron is given by an op-amp summer like configuration shown in figure 3.0.1.

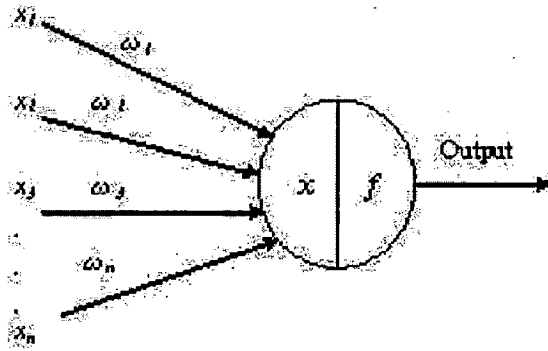


Figure 3.0.1: Structure of Artificial Neuron

Where x_1, x_2, x_3, \dots are input signals, each of the input signal flows through a gain called synaptic weight. The weight can be positive (excitatory) or negative (inhibitory) corresponding, respectively, to acceleration or inhibition [8]. The summing nodes accumulate all the input weighted signals and then pass to the output through the transfer function which is usually nonlinear. The transfer function can be step or threshold type, signum type, or linear threshold type. The transfer function can also be nonlinear continuously varying type, such as sigmoid, inverse-tan, hyperbolic, or Gaussian type. The sigmoidal transfer function is most commonly used, and it is given by Eqn. 3.0.1

$$Y = \frac{1}{1 + e^{-\alpha \cdot x}} \quad (3.0.1)$$

Where α is the coefficient or gain which adjusts the slope of the function. With high gain, this function approaches a step function. The sigmoidal function is nonlinear, monotonic, differentiable, and has the largest incremental gain at zero signal, and these properties are of particular interest.

3.1 Neural network architecture

An artificial neural network is an electrical analogue of a biological neural network. The cell body in an artificial neural network is modeled by a linear activation function. The activation function, in general, attempts to enhance the signal contribution received through different neurons. The synapse in the artificial neural net is modeled by a non-linear inhibiting function, for limiting the amplitude of the signal processed at cell body.

The most common non-linear functions used for synaptic inhibition are: Sigmoid function, Tanh function, Signum function, Step function as shown in Fig. 4.4.10. Sigmoid and tan hyperbolic (tanh) functions are grouped under soft non-linearity, whereas signum and step functions are under hard type non-linearity.

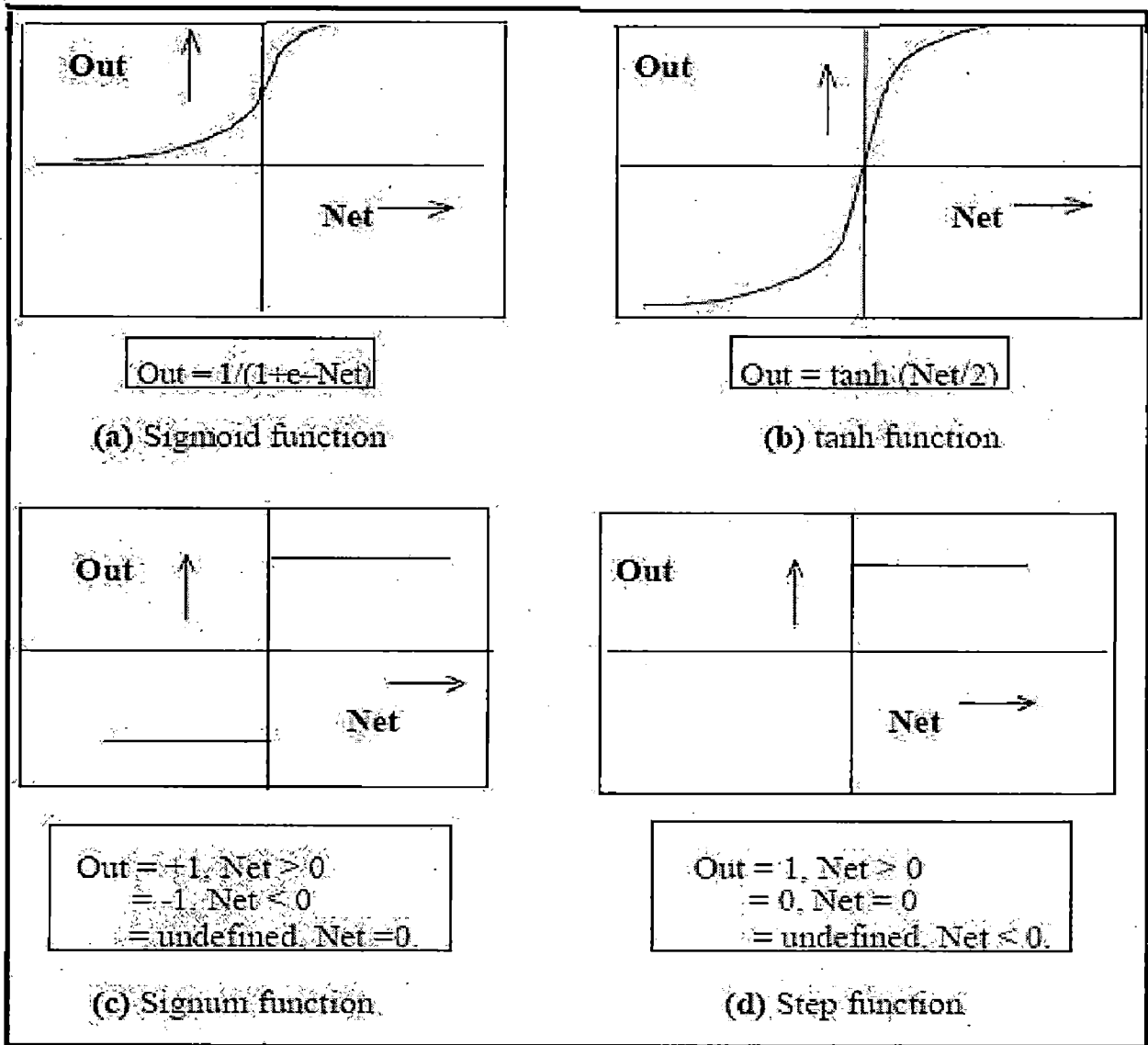


Figure 3.1.2: Common Non-linear Function used for synaptic Functions

Depending on the nature of problems, the artificial neural net is organized in different structural arrangements.

- *Feedforward network*
- *Recurrent network*.

Fig. 4.4.11 Common topologies of Neural Network The single layered recurrent network topology was proposed by Grossberg, which has successfully been applied for classifying analog patterns. The feed forward structured neurals are the most common structures for the well-known backpropagation algorithm.

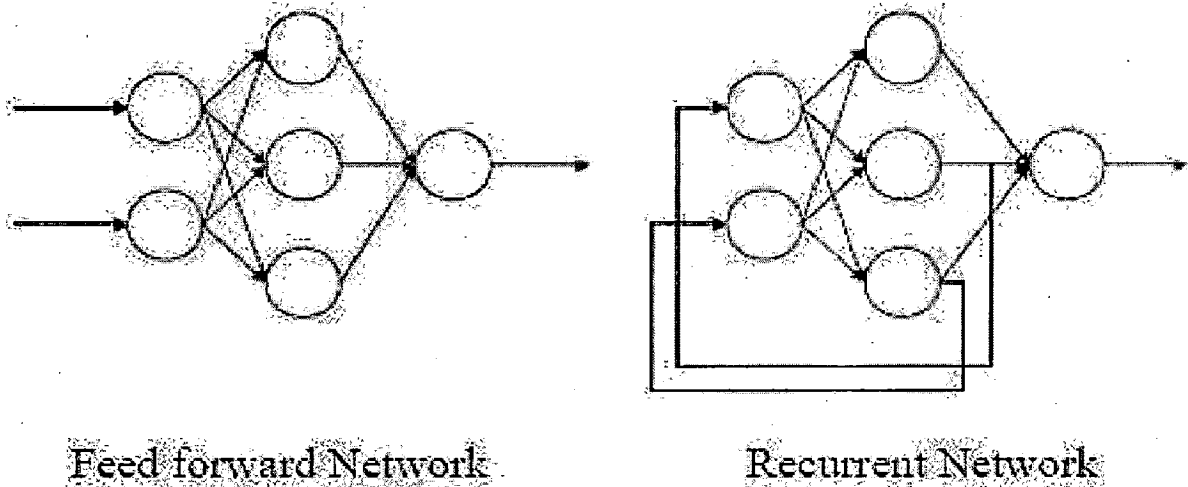


Figure 3.1.3: Common Non-linear Function used for synaptic Functions

3.1.1 Supervised Learning in Neural Networks

Supervised learning is a process of approximating a set of "labelled" data, that is, each datum (which is a data point in the input-output problem space) contains values for attributes (features, independent variables) labelled by the desired value(s) for the dependant variables, for example, the set of Iris examples, each labelled by the class label. Supervised learning can be viewed as approximating a mapping between a domain and a solution space of a problem: $X \rightarrow Y$, when samples (examples) of (input vector, output vector) pairs (x, y) are known, $x \in X, y \in Y$, $x = (x_1, x_2, \dots, x_n)$, $y = (y_1, y_2, \dots, y_m)$. How to achieve an approximation F' of labelled data by using a neural network such that it can generalize on new inputs x is the problem supervised learning is concerned with. Supervised learning in neural networks is usually performed in the following sequence:

1. Set an appropriate structure of a neural network, having, for example, $(n + 1)$ input neurons (n for the input variables and 1 for the bias, x_0) and m output neurons and set initial values of the connection weights of the network.
2. Supply an input vector x from the set of the training examples X to the network.
3. Calculate the output vector o as produced by the neural network.
4. Compare the desired output vector y (answer, from the training data) and the output vector o produced by the network; if possible, evaluate the error.

5. Correct the connection weights in such a way that the next time x is presented to the network, the produced output o becomes closer to the desired output y .
6. If necessary, repeat steps 2 to 5 until the network reach a convergence state.

Evaluating an error of approximation can be done in many ways; the most used being instantaneous error:

$$Err = (o - y), \text{ or } Err = |o - y|; \quad (3.1.2)$$

Mean-square error (MSE):

$$Err = (o - y)^2 / 2 \quad (3.1.3)$$

a total MSE sums the error over all individual examples and all the output neurons in the network:

$$Err = \left(\sum_{k=1}^p \sum_{j=1}^m (o_j^{(k)} - y_j^{(k)})^2 \right) / p.m \quad (3.1.4)$$

Where:

- $o_j^{(k)}$ Is the output value of the j th output of the network when the k th training example is presented;
- $y_j^{(k)}$ is the desired result (the desired output) for the j th output (j th independent variable) for the k th training example;
- p is the number of training examples in the training data;
- m is the dimension of the output space (the number of independent variables equal to the number of the output neurons in the neural network); and root-mean-square error (RMS), the root of the MSE.

Depending on how an error is calculated, two types of error can be evaluated for a neural network. The first, called apparent error, estimates how well a trained network approximates the training examples. The second, called test error, estimates how good a trained network can generalize, that is, react to new input vectors. For evaluating a test error we obviously have to know the desired results for the test examples. Supervised learning is a very useful learning paradigm for solving problems like classification, or for learning a certain prescribed behavior, when the classes, labels, or desired behavior patterns are known. Supervised learning can be used for learning "micro rules" of stimulus-reaction type, element-class type, source-destination type, etc. The above general algorithm for supervised learning in a neural network has different implementations, mainly distinguished by the way the connection weights are changed through training. Some of the algorithms are discussed in this section-perceptron learning (Rosenblatt 1958); ADALINE (Widrow and Hoff 1960); the backpropagation algorithm (Rumelhart et al. 1986b; and others); and (learning vector quantization) LVQ1, 2, 3 algorithms (Kohonen 1990).

Supervised learning uses as much of the information and knowledge as given in the data, but it is considered by many authors not to be plausible at a low, synaptic level. It is obviously plausible at a psychological level because people do learn by being supervised in one way or another, as well as through their own experience (which sometimes can be painful).

3.1.2 The Feed Forward Network

Feed forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of neurons. Multiple layers of neurons with nonlinear transfer functions allow network to learn nonlinear and linear relationships between input and output vectors. A two-layer feed forward network is shown below in Fig. 3.1.4

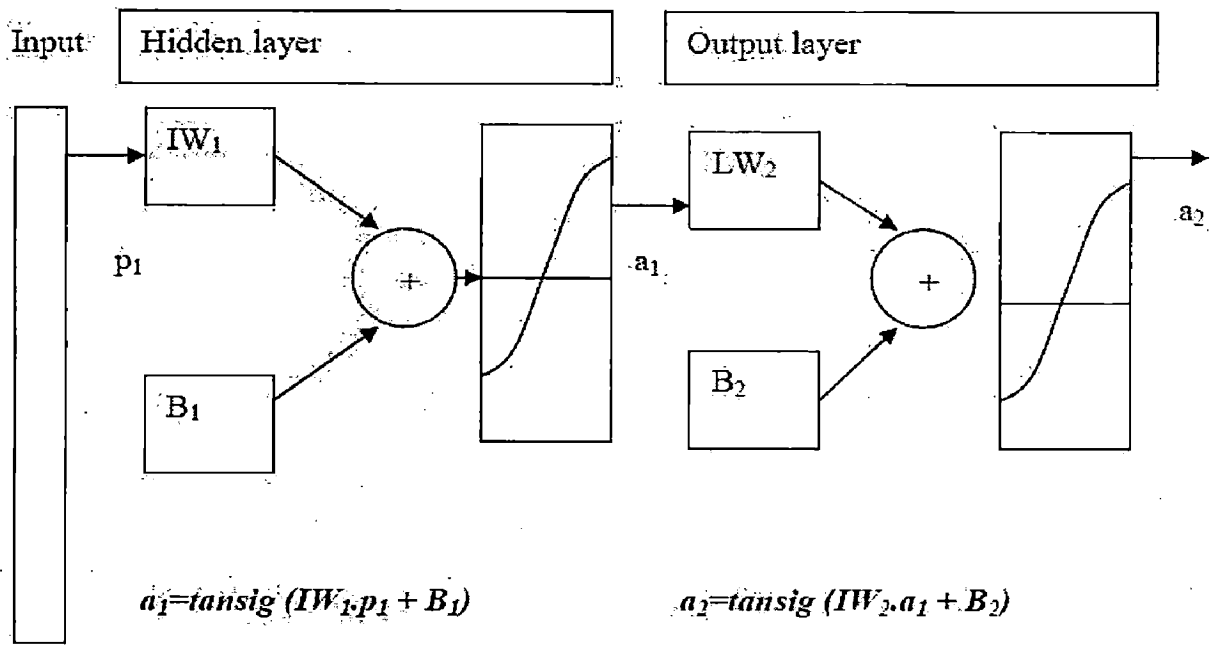


Figure 3.1.4: A two layer feed forward network

The feedforward network is used in present work because it can deal with nonlinear classification problems. Both the hidden layer and the output layer use a continuous network based on the nonlinear sigmoid discriminant function. In a multilayer feedforward neural network, the number of nodes in the input layer is determined by the dimension of the feature space (i.e. the dimension of the input vector). The number of nodes in the output layer is determined by the problem's desired response (i.e. the size of the target vector). Physically, the hidden layers are inaccessible while the output layer provides the user with learning responses after training. However, by adding more hidden layers, the neural network is able to extract higher-order

statistics in order to perform more complex tasks . A multilayer feedforward network produces a response to the input signal by propagating in the forward direction only. There is no feedback. A backpropagation network is a feedforward network with feedback function added.

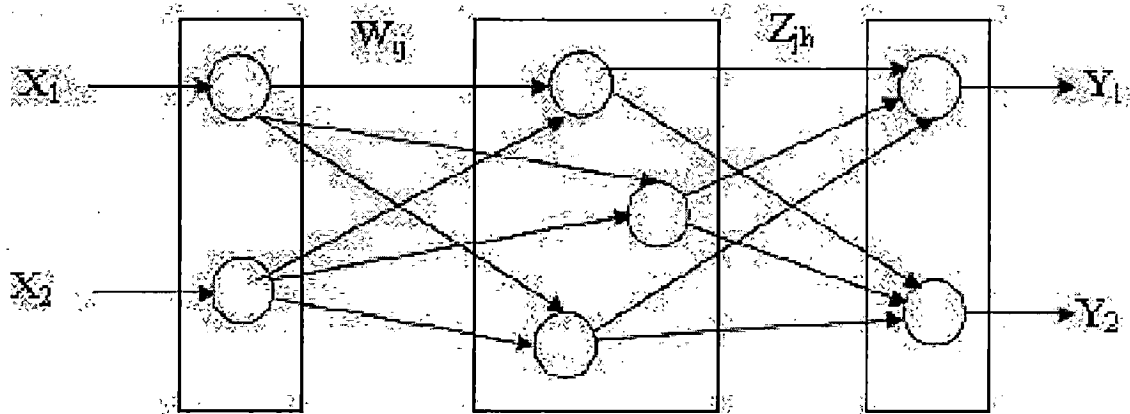


Figure 3.1.5: A Multi layer feed forward network

In general, neural networks can be classified as feedforward and feedback types depending on the interconnection of the neurons. At present, the majority of the problems use feedforward architecture, and it is of direct relevance to power electronics and motion control applications. Figure 3.1.5 shows the structure of a feedforward multilayer network with two input and two output signals. The topology is based on Perceptron which was proposed by Rosenblatt in 1958.

The circles represent neurons and the dots in the connections represent the weights. Fig 3.1.5 shows that Feed Forward Multi-Layer Neural Network The network has three layers, defined as input layer (a), hidden layer (b), and output layer (c). The hidden layer functions as a connection between the input and the output layers. The input and output layers have neurons equal to the respective number of signals. The input layer neurons do not have transfer functions, but there are scale factors, as shown, to normalize the input signals. The number of hidden layers and the number of neurons in each hidden layer depend on the network design considerations. The input layer transmits the signals to the hidden layer, and the hidden layer, in turn, transmits the signals to the output layer, as shown. The network can be fully connected or partially connected.

3.1.3 Back Propagation Training

Back-Propagation training algorithm is most commonly used in a feedforward neural network as mentioned before. For this reason, a feed forward network is often defined as "back-prop" network. Figure 3.1.6 shows the principle of back propagation training.

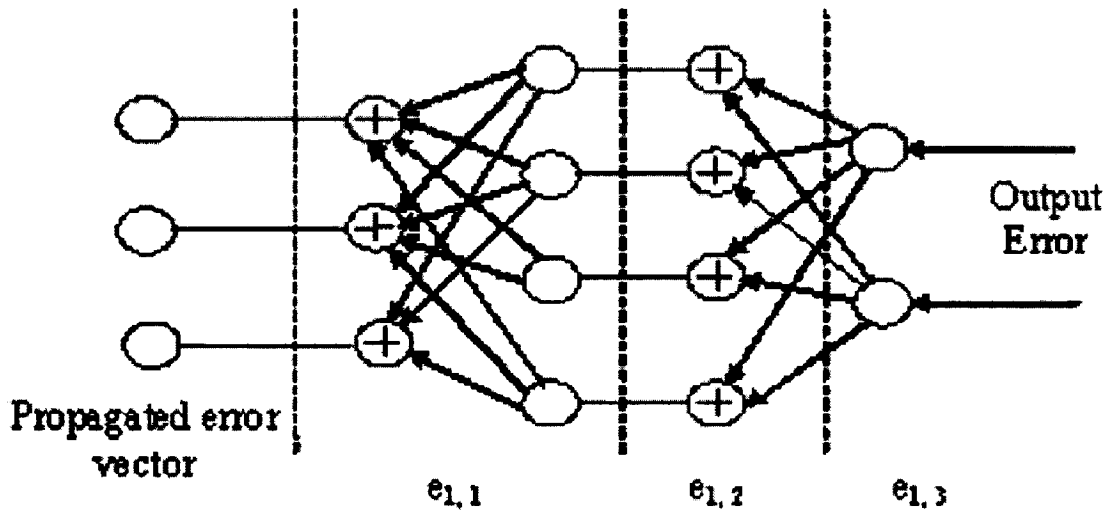
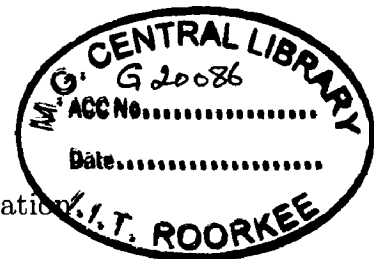


Figure 3.1.6: Principle of Back Propagation



In the beginning, the network is assigned random positive and negative weights. For a given input signal pattern, step by step calculations are made in the forward direction to derive the output pattern. A cost functional given by the squared difference between the net output and the desired net output for the set of input patterns is generated and this is minimized by gradient descent method altering the weights one at a time starting from the output layer. The equations for the output of a single processing unit are given as

$$Net_j^p = \sum_{i=1}^N W_{ij} X_i \quad (3.1.5)$$

$$Y_j^p = f_j(Net_j^p) \quad (3.1.6)$$

Where,

- j is the processing unit under consideration
- p is the input pattern number

- X_i is the output of the neuron connected to the neuron
- W_{ij} is the connection weight between the and neurons.
- Net_j^p is the output of the summing node, i.e., the neuron activation signal.
- N is the number of the neurons feeding the neuron.
- f_j is the nonlinear differentiable transfer function (usually sigmoid), and
- y_j^p is the output of the corresponding neuron.

For the input pattern p , the squared output error for all the output layer neurons of the network is given as

$$E_p = \frac{1}{2}(d^p - y^p)^2 = \frac{1}{2}\sum_{j=1}^S (d_j^p - y_j^p)^2 \quad (3.1.7)$$

Where,

- d_j^p is the desired output of the neuron in the output layer,
- y_j^p is the corresponding actual output,
- S is the dimension of the output vector,
- y^p is the actual net output vector, and
- d_p is the corresponding desired output vector.

The total squared error E for the set of P patterns is then given by

$$E = \frac{1}{2}\sum_{p=1}^P = \frac{1}{2}\sum_{p=1}^P \sum_{j=1}^S (d_j^p - y_j^p)^2 \quad (3.1.8)$$

The weights are changed to reduce the cost functional E in a minimum value by gradient descent method, as mentioned. The weight update equation is then given as

$$W_{ij}(t+1) = W_{ij}(t) + \eta \left(\frac{E_p}{\delta W_{ij}(t)} \right) \quad (3.1.9)$$

Where

- η is the learning rate,
- $W_{ij}(t+1)$ is the new weight and
- $W_{ij}(t)$ is the old weight. The weights are updated for all the P training patterns. Sufficient learning is achieved when the total error E summed over the patterns falls below a prescribed threshold value. The iterative process propagates the error back-propagation [8, 46, 45]

3.1.4 Neural Network Model reference Controller

ANN is one of the branches of artificial intelligence finding widespread application in controlling the power flow of transmission and distribution networks using FACTS technology. ANN has been successfully applied in the identification and control of dynamic systems. Neural network controllers[40] have been implemented in the field of electrical power in many applications such as electric load forecasting, transient ability assessment, harmonic source identification, inverter current controllers, speed control of synchronous motor drives, etc. The self-adapting and superfast computing features of ANN make them well suited to handle nonlinearities, uncertainties, and parameter variations that can occur in power electronic systems. In learning process, neural network adjusts its structure such that it is able to output the same signals as the supervisor. The learning is repeated until the difference between network output and supervisor is enough.

This research article introduces the application of one of the popular neural network controllers, namely, model reference controller implemented in neural network toolbox of MATLAB. MRC could be used to regulate the speed by controlling the armature voltage. There are two stages involved in the implementation of this controller, namely, system identification stage and controller training stage. Figure 3.1.7 shows the block diagram representation of the system identification stage. Plant identification GUI is an interactive environment for developing a neural network capable of modeling a given plant. In the control design stage, the developed neural network plant model is used to train the controller. It has been observed that well trained controllers will be able to keep track of the reference signal.

3.1.5 System Identification stage

The first phase of plant identification process is to generate input/output data to train a neural network to represent the forward dynamics of the plant. This could be achieved by either generating the training data from simulink plant model or by importing the training data from a valid data file with input and output values. The LevenbergMarquadt algorithm [42] is used for training the plant model. Once the training data is acceptable, a neural network could be trained to identify the function of the plant identification model described as:

$$y(k+d) = N(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)) \quad (3.1.10)$$

where $u(k)$ is the system input, $y(k)$ is the system output and d is the system delay. To determine the control input that causes the plant output to follow a specific reference, the controller could

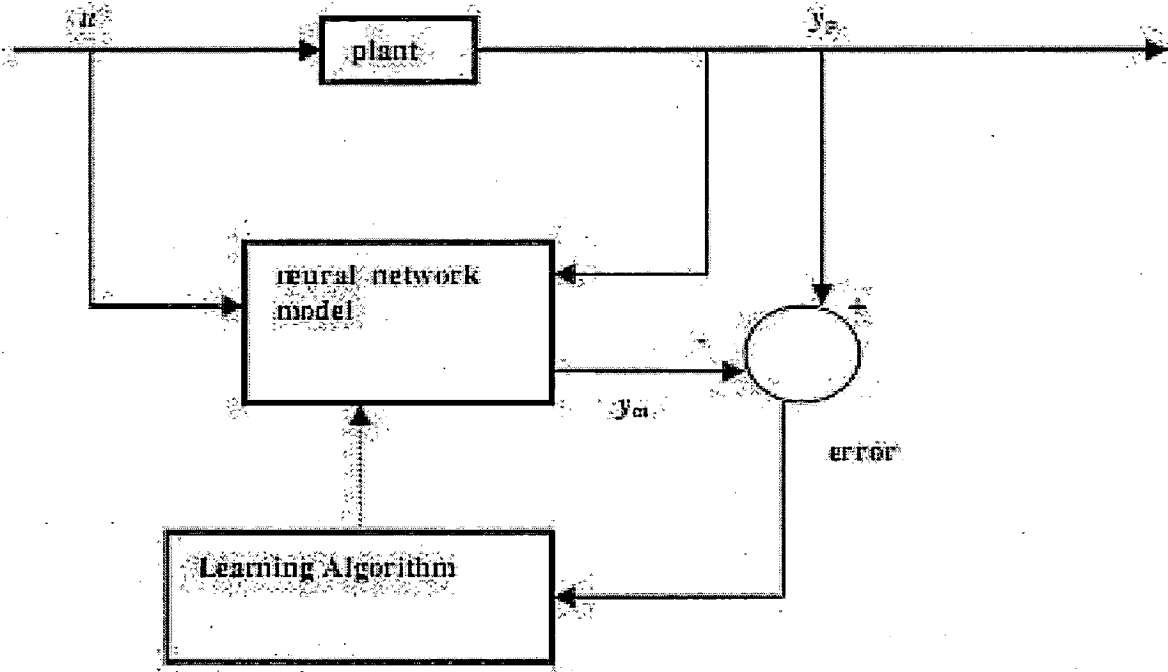


Figure 3.1.7: Block diagram representation of system identification stage

be identified using the expression:

$$u(k) = \frac{y_r(k+d) - f(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1))}{g(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1))} \quad (3.1.11)$$

However, determination of the control input based on the output at the same time is not realistic and hence uses the model.

$$y(k+d) = f(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)) + g(y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)) * u(k+1) \quad (3.1.12)$$

The SIMULINK plant model shown in Fig. 3.1.8 is used to generate the input/output pattern required in the controller training stage. The plant model must have an input port and output port. One may note that the input of the plant is the driving voltage to the armature and the output is the corresponding speed of the motor. Plant model specifications are tabulated in Appendix 1. As specified in appendix 1, the maximum and minimum input voltages to the armature have been treated as 240 and 0 V, respectively. Random inputs in between 240 and 0 V will be applied to the SIMULINK plant model[40] to generate the training data. The maximum plant output has been taken as the rated speed of the motor. Performance graph and training data obtained for neural network MRC at the system identification stage are illustrated

in Fig 3.1.10, respectively. The neural network simulation results are:

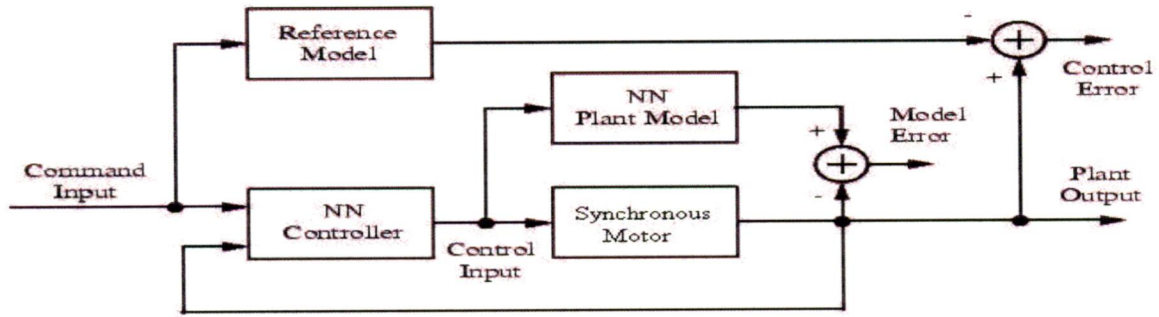


Figure 3.1.8: Block diagram representation of Model Reference Controller

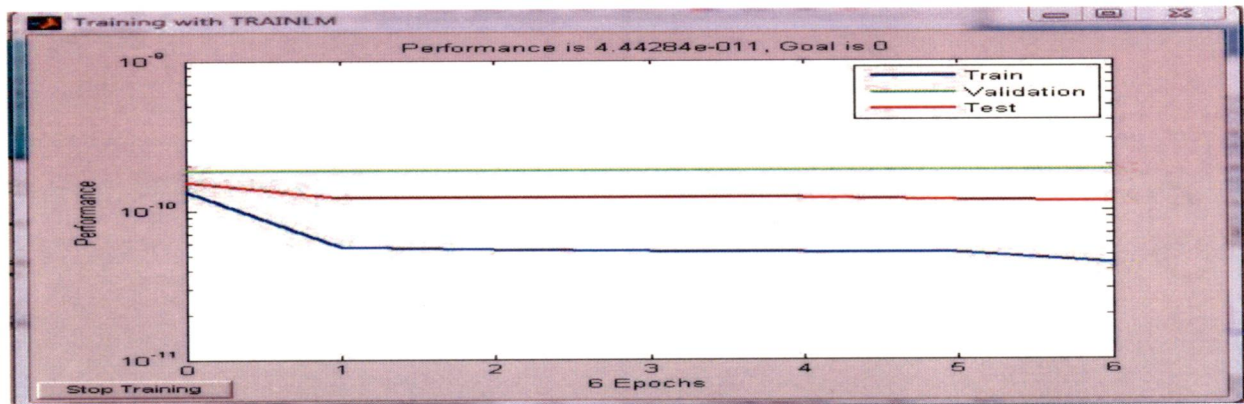


Figure 3.1.9: Performance curve for the plant identification model

The testing, validation and training data are shown in the Figures 3.1.11

3.2 Model Reference Neural Network Controller in the control of Permanent Magnet Synchronous Motor

To investigate the effectiveness of the proposed adaptive neural speed controller, a second-order system transfer function with the following prescribed characteristics: 0.3 s rise time, no steady-state error and unity damping ratio to avoid overshoot is chosen as the reference mode l for the periodic step command. Therefore, in accordance with [39], the following transfer function is chosen in this study:

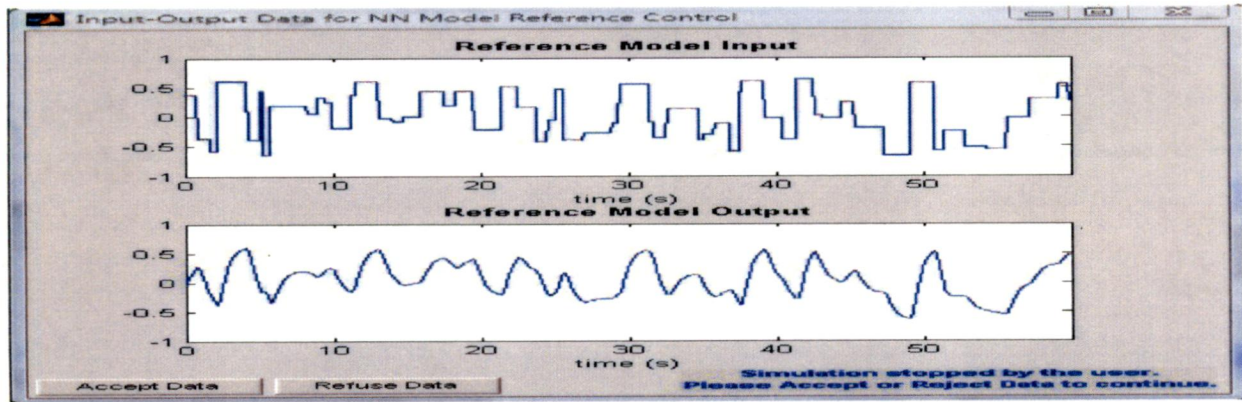


Figure 3.1.10: performance curve and input-output curve for NN model

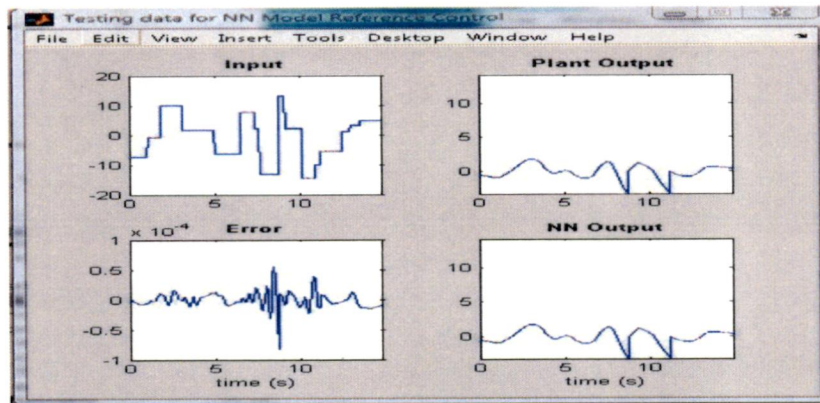


Figure 3.1.11: Testing Data

$$G(s) = \frac{168.11}{s^2 + 25.93s + 168.11} \quad (3.2.13)$$

The PMSM used in this drive system is a three-phase four-pole 750 W 3.47 A 1500 rpm type is shown in fig 3.2.14

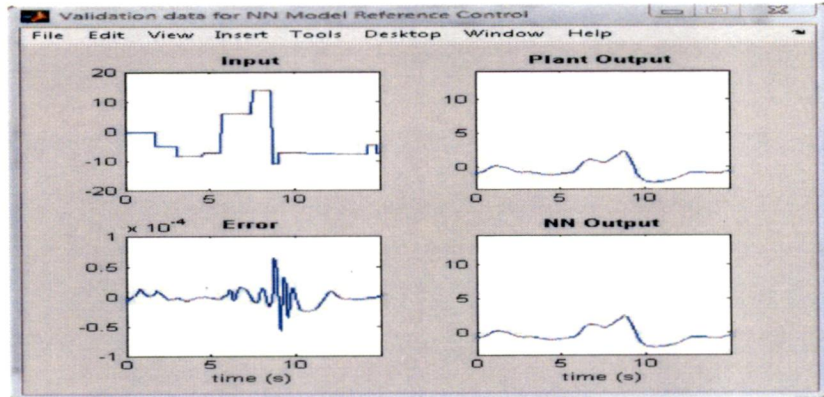


Figure 3.1.12: Validation Data

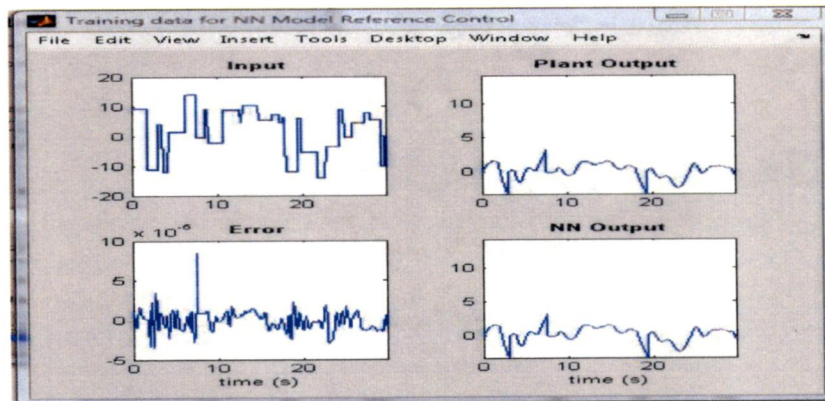


Figure 3.1.13: Training Data

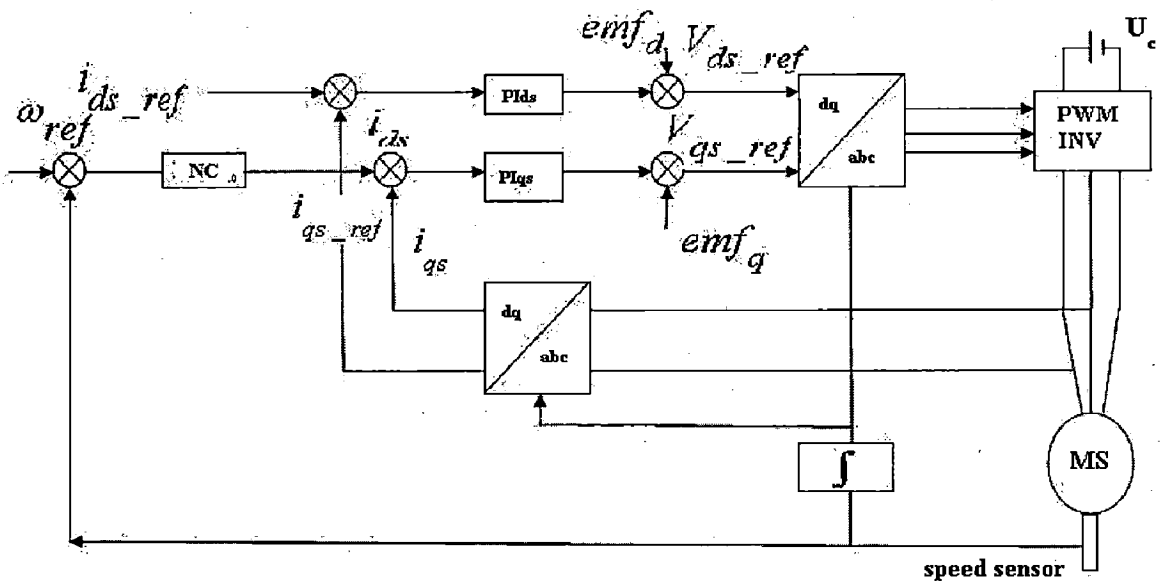


Figure 3.2.14: Permanent Magnet Synchronous Motor Using Neural Network Controller

Chapter 4

Fuzzy Logic Based Speed Controller

4.1 General Introduction

The two most commonly implemented command current generating algorithms for controlling the current-controlled VSI are PI and PID schemes. In these schemes the speed error (Command speed-actual speed) is used to generate the command torque necessary in order to return the rotor to the command speed. The stator currents that must be applied to the motor in order to produce this desired speed are obtained from the required torque, and then the VSI is used to apply these currents, under the influence of the command currents, the rotor speed is made to track the command speed.

The calculations of command torque, however, rely upon mathematical modeling equations of the PMSM that are dependent upon the internal motor parameters of d- and q- axis reactances. This leads to problems because, in the PMSM, the rotor magnetic saliency, saturation and armature reaction vary during operation under different speed and loading conditions and thus affect the air gap flux and reactance parameters. Therefore, under operating conditions, the PMSM model contains unknown dynamics. This affects the performance of PI and PID based control systems at different operating conditions, because in these controllers the d-q axis parameters are assumed to be constant. In addition, conventional PI and PID controllers are very sensitive to step changes of command speed and load disturbances. Thus, the effective control of PMSM needs a complex structure for high performance applications, where rapid speed response, fast and precise handling of load changes and parameter variations, overload capacity, maintenance free operation, size, weight and robustness are all of primary concern.

The use of fuzzy logic controllers (FLC) eliminates much of these problems and makes the control systems more generic. The FLC has the advantage over conventional controllers because it does not need the exact systems mathematical model, and therefore it does not rely upon knowledge of dynamically changing parameters, such as reactances. Thus it can handle nonlinear

functions of any arbitrary complexity, and it is easily expanded and modified.

4.2 Fundamentals of Fuzzy Logic Control

Fuzzy logic is an extension of Boolean logic that is designed to handle the concept of partial truth-truth values between "completely true" and "completely false"- between 0 and 1 [5] For instance, in fuzzy logic a statement may be true to a degree of 0.7, not just 1 or 0.

The fuzzy set(subset) A on the universe(set) X is defined by a membership function, μ_A from X to the real interval[0,1], which associates a number $\mu_A(x) \in [0, 1]$ to each element x of universe X. $\mu_A(x)$ represents the degree of membership of the element x to the fuzzy set A. For example, the equation $\mu_A(x) = 0.5$ means x has A-ness of about 50A fuzzy singleton $S(x_o) = \mu_A(x)$ is a fuzzy set that supports only one element x_o . Therefore, the fuzzy set(A in this discussion) is the union of the fuzzy singletons of all its constituent elements(here, all the elements x of universe X). However, in fuzzy set theory, the boundaries of the fuzzy sets can be vague and ambiguous, making it useful for appropriate systems. Fuzzy sets are represented graphically buy means of their membership functions. The four most popularly use membership functions are shown in Fig. 4.2.1 These membership functions can be defined as,

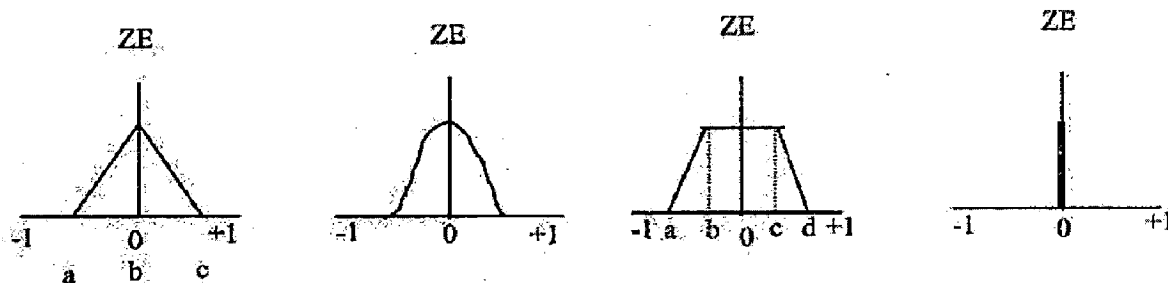


Figure 4.2.1: Membership functions of linguistic value
ZE: Triangular, Gaussian, Trapezoidal, Singleton

$$\begin{aligned}
 \text{Triangular : } f(x; a, b, c) &= \{0, x \leq a \\
 &= \frac{x - a}{b - a}, a \leq x \leq b \\
 &= \frac{c - x}{c - b}, b \leq x \leq c \\
 &= 0, x \geq c\}
 \end{aligned}$$

$$\text{Gaussian Function : } f(x; \sigma, c) = e^{-\frac{(x-c)^2}{2\sigma^2}} \quad (4.2.1)$$

$$\begin{aligned} \text{Trapezoidal : } f(x; a, b, c) &= \{0, x \leq a \\ &= \frac{x-a}{b-a}, a \leq x \leq b \\ &= 1, b \leq x \leq c \\ &= \frac{c-x}{c-b}, c \leq x \leq d \\ &= 0, x \geq d\} \end{aligned}$$

$$\text{Singleton : } f(x) = \{1, x = x_o :: 0, x \neq x_o\} \quad (4.2.2)$$

Fig. 4.2.1 shows some possible choices of membership functions for a fuzzy set associated with the linguistic value ZE in the universe $X = [-1, 1]$. In these examples we see that the number 0 fully belongs to the fuzzy sets while the numbers -1 and +1 do not. This need not necessarily be the case for all possible choices of membership functions. The choice of fuzzy logic membership functions depends on the expert.

The complete process of formulating the mapping from a given input to an output using fuzzy logic is known as fuzzy inference. There are two types of fuzzy inference methods: Mamdani and Sugeno types [6]. The difference between the two methods is only in the way the output is defined. In control applications, Mamdani type fuzzy inference is the most commonly used method and is the one utilized for this work. The process of fuzzy inference consists of three main components. These are given as follows:

1. Fuzzification
2. Rule Base Evaluation
3. Defuzzification

4.2.1 Fuzzification

The general structure of a complete fuzzy control system is given in Fig. 4.2.2. The plant control u is inferred from the two state variables, error (e) and change in error Δe [7].

The fuzzification module converts the crisp values of the control inputs into fuzzy values. A fuzzy variable has values which are defined by linguistic variables (fuzzy sets or subsets) such as

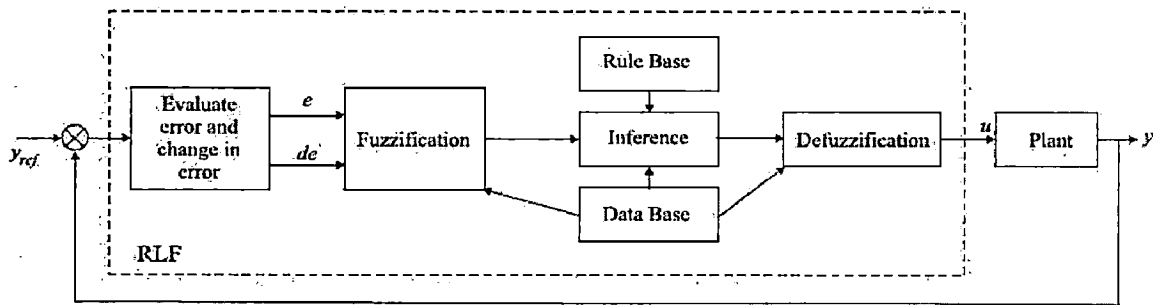


Figure 4.2.2: Basic Structure of Fuzzy Control System

low, Medium, high, big, slow where each is defined by a gradually varying membership function [7, 8].

4.2.2 Database and Rules

The data base and the rules form the knowledge base which is used to obtain the inference relation R . The data base contains a description of input and output variables using fuzzy sets. The rule base is essentially the control strategy of the system. It is usually obtained from expert knowledge or heuristics; it contains a collection of fuzzy conditional statements expressed as a set of IF-THEN rules, such as: $R(i)$:

If x_1 is F_1 and x_2 is F_2 and x_n is F_n THEN Y is $G(i)$, $i=1, \dots, M$

where : x_1, x_2, \dots, x_n is the input variables vector, Y is the control variable, M is the number of rules, n is the number fuzzy variables, (F_1, F_2, F_n) are the fuzzy sets. For the given rule base of a control system, the fuzzy controller determines the rule base to be fired for the specific input signal condition and then computes the effective control action (the output fuzzy variable) [7, 9]. The mathematical procedure of converting fuzzy values into crisp values is known as 'defuzzification'. This operation can be performed by several methods of which center of gravity (or centroid) and height methods are common [9, 10]. The actual crisp input are approximates to the closer values of the respective universes of discourse. Hence, the fuzzyfied inputs are described by singleton fuzzy sets. The elaboration of this controller is based on the phase plan. The control rules are designed to assign a fuzzy set of the control input U for each combination of fuzzy sets of e and Δe [11]. Fig. 4.7.15 shows one of possible control rules. Fig. 4.7.15 shows the rules base. The rows represent the rate of the error change Δe and the columns represent the error e . Each pair $(e, \Delta e)$ determines the output level NB to PB corresponding to control input U .

du		dE_n				
		NB	NM	ZR	PM	PB
E_n	NB	NB	NB	NM	NM	ZR
	NM	NB	NM	NM	ZR	PM
	ZR	NM	NM	ZR	PM	PM
	PM	NM	ZR	PM	PM	GP
	PB	ZR	PM	PM	GP	GP

Figure 4.2.3: Basic Structure of Fuzzy Control System

Here NB is negative big, NM is negative medium, ZR is zero, PM is positive medium and PB is positive big, are labels of fuzzy sets and their corresponding membership functions are depicted in Figures ??, respectively. The continuity of input membership functions, reasoning method, and defuzzification method for the continuity of the mapping $\mu_{fuzzy}(e, \dot{e})$ is necessary.

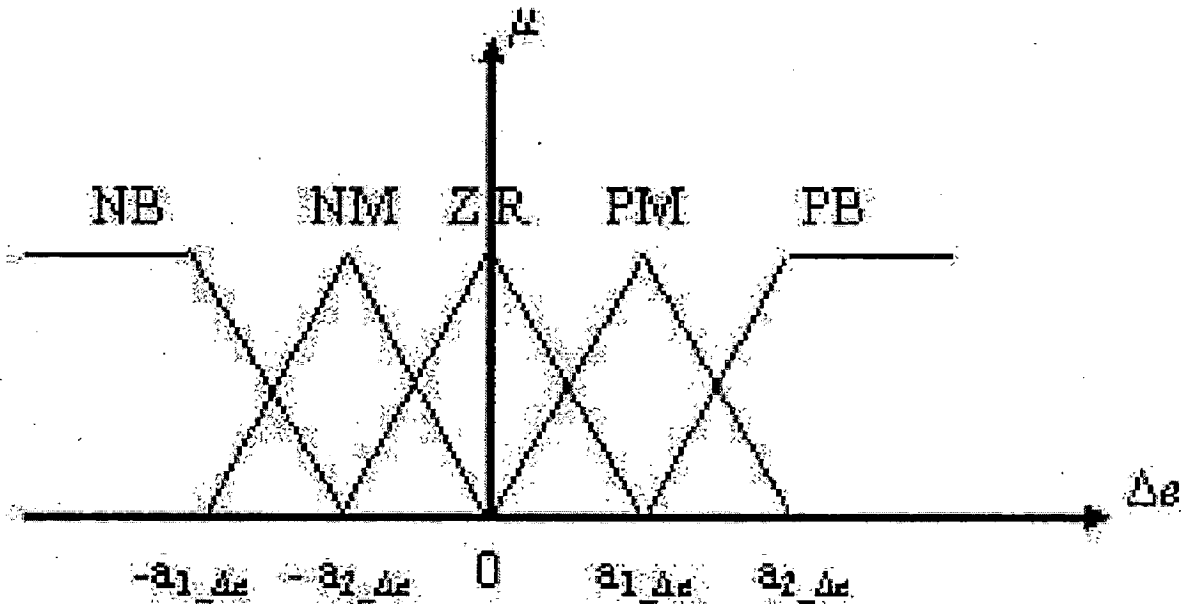


Figure 4.2.4: Membership function of input Δe

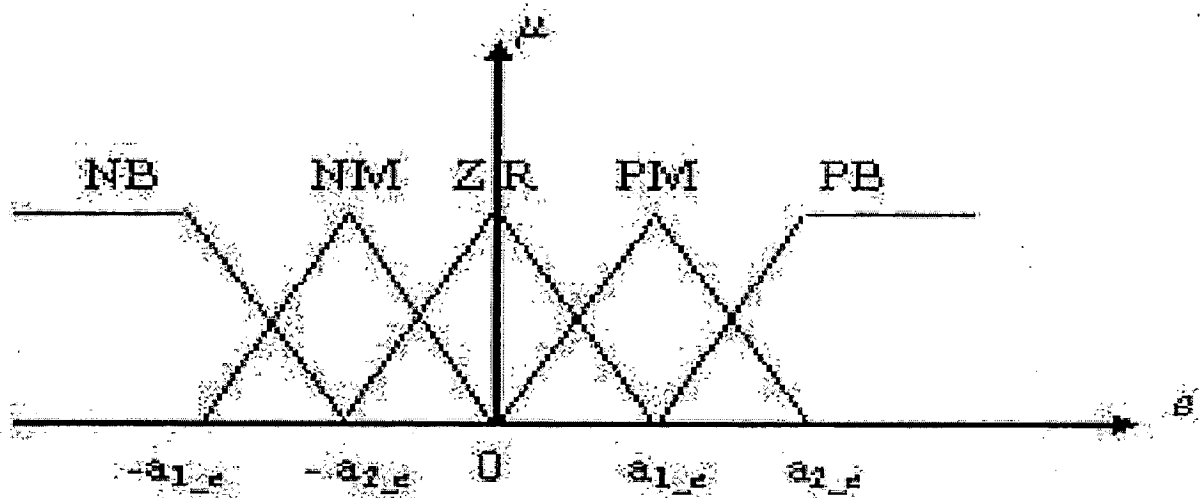


Figure 4.2.5: Membership function of input e

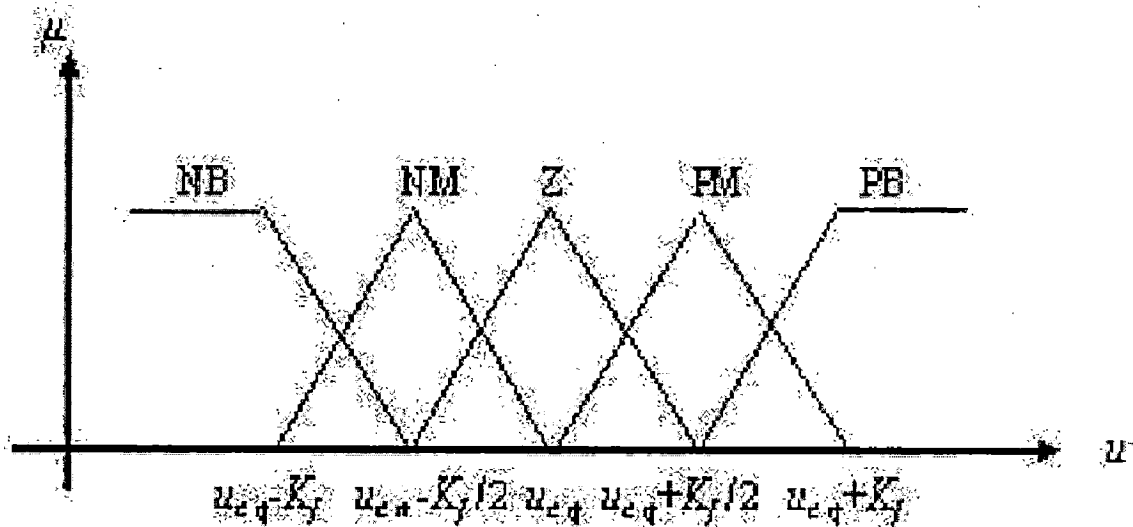


Figure 4.2.6: Membership function of output U

4.2.3 Inferencing

Max-Min reasoning Method: The MAX-MIN method tests the magnitudes of each rule and selects the highest one. The horizontal coordinate of the "fuzzy centroid" of the area under that function is taken as the output. This method does not combine the effects of all applicable rules but does produce a continuous output function and is easy to implement.

We shall consider the following multiple fuzzy reasoning form: RULE 1: $A_1 \text{ and } B_1 \Rightarrow C_1$

RULE 2: $A_2 \text{ and } B_2 \Rightarrow C_2$

.....

.....

RULE n: $A_n \text{ and } B_n \Rightarrow C_n$

FACT: X AND Y CONS: C'

The inference result C_i ($i=1,2,3,\dots,n$) which is inferred from the fact [X and Y] and the fuzzy rule is given as below:

$$\mu_{C_i'}(z) = \mu_{A_i}(X_o) \wedge \mu_{B_i}(Y_o) \wedge \mu_{C_i}(Z_o) \quad (4.2.3)$$

The final consequence C' is aggregated by taking the union(U) of C_1', C_2', \dots, C_n' obtained by

$$\mu_{C'}(z) = \mu_{C_1'}(X_o) \vee \mu_{C_2'}(Z) \vee \mu_{C_3'}(Z) \quad (4.2.4)$$

This fuzzy reasoning method is known as Mamdani's method and called "MIN-MAX GRAVITY METHOD".

4.3 Defuzzification

The tasks of defuzzification module are to: convert the set of controller output values into a single point wise value, perform output re-normalization that maps the point wise value of the controller output into its physical domain. The choice of defuzzification operator is the most important consideration in the defuzzification stage. Some of the most widely used defuzzifiers are:

The center of gravity defuzzifier specifies the z^* as the center of the area (COA) covered by the membership function of C' and is given by equation of the form.

$$Z^* = \frac{\int_w \mu_c(z) dz}{\int_w \mu_c(z) z . dz} \quad (4.3.5)$$

Where \int_w is the convolution integral. The advantage of the center of gravity defuzzifier lies in its intuitive plausibility. The disadvantage is that it is computationally intensive.

4.4 Control Of Permanent Magnet Synchronous Motor through Fuzzy Logic Control

The schematic diagram of the speed control system under study is shown in Fig. 4.4.7. The power circuit consists of a continuous voltage supply which can provided by a six rectifier

thyristors and a three phase GTO thyristors inverter whose output is connected to the stator of the synchronous machine. The field current of the synchronous machine, which determines the field flux level is controlled by voltage. The self-control operation of the inverter-fed synchronous machine results in a rotor field oriented control of the torque and flux in the machine as mentioned earlier. The flux in the machine is controlled independently by the field winding and the torque is affected by the fundamental component of armature current Fig. 4.4.7 shows the schematic diagram of the speed control of synchronous motor using the FLC.

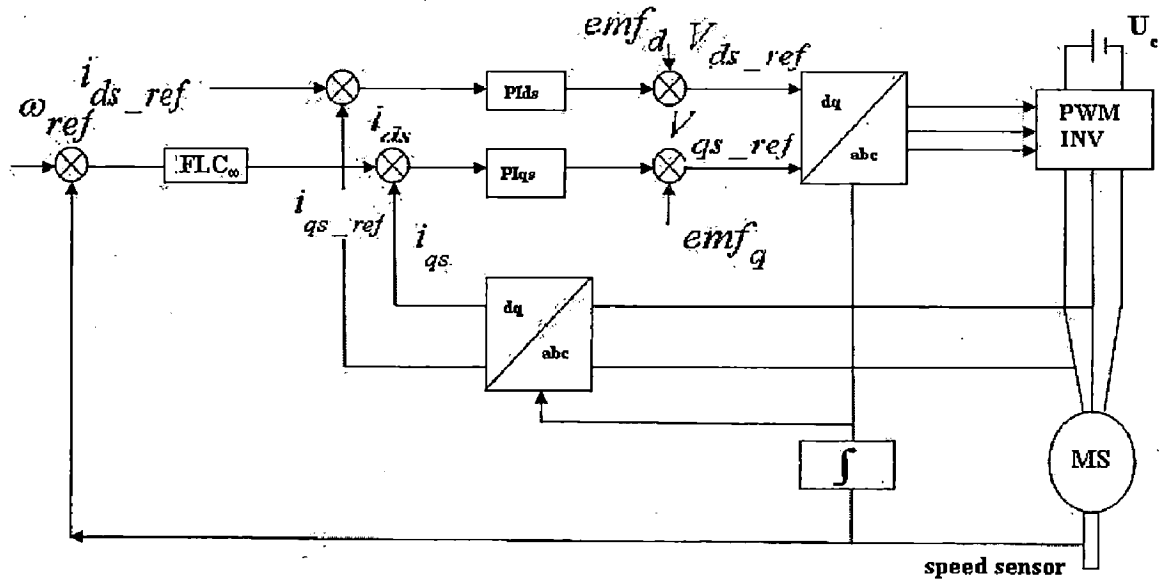


Figure 4.4.7: Center of Area Defuzzifier Method

The blocks FLC_{ω} , NC_{ω} , PI_{id} , PI_{iq} are regulators, the first one is the fuzzy controller for speed and the third is the proportional integral (PI) regulator. For direct current and the fourth is the PI regulator for the quadrature current. To avoid the appearance of an inadmissible value of current, a saturation block is used.

4.4.1 FLC Structure for the PMSM Drive

The motor dynamics can be represented by the following equation:

$$T_e = T_L + B_m \omega_r + J_m p \omega_r \quad (4.4.6)$$

where T_e is the electrical torque, T_L is the load torque, B_m is the friction damping coefficient, J_m is the rotor inertia constant, p is the differential operator and ω_r is the rotor speed.

The dynamic model of the PMSM can be rewritten from the synchronous motor dynamic equation and Eqn. 4.4.6 as

$$L_q p i_q + L_d P \omega_r i_d = v_q - r_s i_q - K_b \omega_r p \omega_r = (T_e - T_L - B_m \omega_r) / J_m \quad (4.4.7)$$

where $K_b = P \Psi_m$. As the FLC can handle any non-linearity, one can consider the load as having unknown nonlinear mechanical characteristics. The load can be modeled using the following equation as

$$T_L = A \omega_r^2 + B \omega_r + C \quad (4.4.8)$$

where A,B,C are arbitrary constants. To make the control task easier, the equations of an PMSM are expressed as a single input and single output system by combining equations ?? in continous time domain form as,

$$J_m \frac{d\omega_r}{dt} = T_e - (B_m + B) \omega_r - A(\omega_r + \Delta\omega_r)^2 - C \quad (4.4.9)$$

$$J_m \frac{d\omega_r + \Delta\omega_r}{dt} = (T_e + \Delta T_e) - (B_m + B)(\omega_r + \Delta\omega_r) - A(\omega_r + \Delta\omega_r)^2 - C \quad (4.4.10)$$

Subtracting Eqn. 4.4.9 from Eqn. 4.4.10 gives,

$$J_m \frac{d(\Delta\omega_r)}{dt} = (\Delta T_e) - (B_m + B + 2A\omega_r)(\Delta\omega_r) - A(\Delta\omega_r)^2 \quad (4.4.11)$$

By replacing all the continous quantities of Eqn. 4.4.11 by their finiter differences, the discrete time small signal model of the simplified PMSM with nonlinear load can be given as

$$\Delta T_e(n) = \frac{J_m}{t_s} \Delta e(n) + (B_m + B + 2A\omega_r(n))(\Delta\omega_r(n)) + A(\Delta\omega_r(n))^2 \quad (4.4.12)$$

Hence,

$$T_e(n) = \oint_{discrete} \Delta T_e(n) = f(\Delta e(n), \Delta\omega_r(n), \omega_r(n)) \quad (4.4.13)$$

where $\Delta e(n) = \Delta\omega_r(n) - \Delta\omega_r(n-1)$ is the change of speed error, $\Delta\omega_r(n) = \omega_r^*(n) - \omega_r(n)$ is the present sample of speed error, $\Delta\omega_r(n-1)$ is the past sample of speed error, $\omega_r(n)$ is the present sample of actual speed, $\omega_r^*(n)$ is the present sample of command speed, t_s is the sampling time interval and f denotes the nonlinear function. Thus the purpose of using the FLC is to map the nonlinear functional relationship between electrical torque T_e and the rotor speed ω_r .

In real-time, the command q-axis and d-axis currents i_q^* and i_d^* are used to get the motor command phase currents i_a^*, i_b^*, i_c^* by using Park's Transformation.

4.4.2 Tuning Procedure of Fuzzy parameters

There are two categories of Fuzzy Logic metering parameters that can be tuned: dynamic range limits and rule weights. The dynamic ranges, affect controller behavior by defining the linguistic variables. The rule weights, affect controller behavior by defining the relative importance of each rule. Rule weights can be thought of as a way to balance objectives. Two dynamic range parameters, LL, representing the low limit, and HL, representing the high limit, are associated with both inputs and outputs of the FL ramp metering algorithm.

Essentially, these dynamic range parameters can be adjusted to redefine each fuzzy class, such as, NB, NM, ZR, PM, PB. Adjusting these parameters changes both the shape and range of the fuzzy class, which changes the behavior of the controller as a result. Within the fuzzification process of the controller, the the following scaling equation normalizes the crisp variables from the (LL, HL) range to the [0, 1] range (Taylor and Meldrum, 1995, 1997, 2000a; Taylor et al., 1998), as shown.

$$\text{Normalized Variable} = \frac{\text{OriginalCrispvariable} - LL}{HL - LL}$$

4.5 Introduction to Particle Swarm Optimization Algorithm

4.5.1 Overview of Intelligent Optimization Using Stochastic Search

Here, intelligent optimization refers to a broad category of population based stochastic optimization algorithms, such as Differential Evolution (DE), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), etc. Intelligent optimization algorithms are considered advantageous compared with classic optimization methods if the optimization problem is complex, stochastic, or highly nonlinear with multiple local optima. Specifically, the advantages of intelligent optimization lie in their intrinsic parallelism, ability to solve huge and complex problems, minimum requirement on domain specific knowledge, etc. Details about these advantages are discussed as follows.

First, these intelligent optimization algorithms are intrinsically parallel. Most classic algorithms are serial and can only explore the searching space in one direction at a time. This difference between classic algorithms and intelligent algorithms is illustrated in Figure 4.5.8. From this figure, it can be seen that if the solution discovered turns out to be suboptimal, there is nothing to do but abandon the current search and start over. Instead, the intelligent optimization algorithms can explore the solution space in multiple directions simultaneously. If one path does not work, they can easily eliminate that path and continue work on more promising ones. This provides a greater chance to find the optimal solution.

Due to the parallelism, intelligent optimization algorithms are particularly suitable for huge

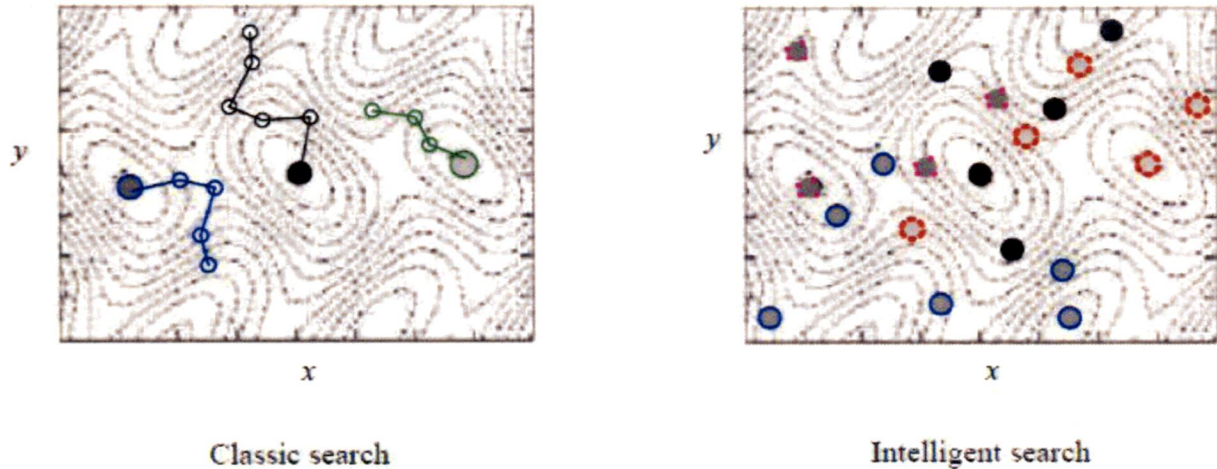


Figure 4.5.8: Difference Between Classic and Intelligent Algorithms

problems where the solution space is too vast to search exhaustively in reasonable time. For example, for the defensive islanding problem of a power system consisting of n transmission lines described in [?], the number of possible solutions is equal to 2^n . Thus, the search space is huge even for a medium scale power system with 41 lines because of the existence of $2^{41} = 2.199 \times 10^{12}$ possible solutions. The implicit parallelism of the intelligent optimization algorithms allow them to successfully find optimal or very good results in a short time after exploring a small region of the searching space.

Another notable strength of the intelligent optimization algorithms is that they perform well for complex problems with multiple local optima. Figure 4.5.9 illustrates the existence of local optima in a two-dimensional searching space. Most practical problems are much more complex than this example and may have a vast searching space that is impossible to search thoroughly. Many classic search algorithms can be trapped in local optima. Intelligent optimization algorithms have been proven to be effective at escaping from local optima and discovering the global optimum in complex searching spaces. It should be noted that, sometimes, there is no way to tell whether a solution is the global optimum or just a very good local optimum. However, even when the intelligent optimization algorithms cannot find the global optima, it can usually find a good local optimum. The last advantage worth mentioning here is that these intelligent optimization algorithms do not require detailed domain specific knowledge as other optimization algorithms do. This advantage is very important for their possible application to the fault diagnosis problem here. Instead of using domain specific information to guide the search, these algorithms make random changes to their candidate solutions and then use a fitness function, often multivalued, to determine whether those solutions are good or not.

As a new development of the intelligent optimization algorithms, Particle Swarm Optimiza-

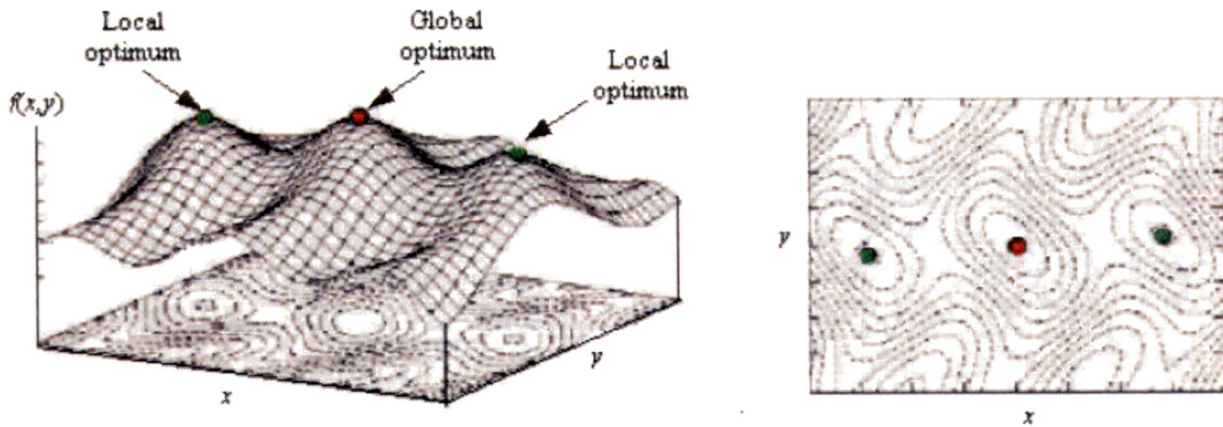


Figure 4.5.9: Existence of local optima in a two-dimensional searching space

tion is simple in concept and highly computationally efficient. It has been proved to be a very powerful algorithm and has been applied to a lot of practical problems in the past years. For the above mentioned defensive islanding problem, the PSO algorithm can find efficient solutions by investigating just a very small percent of the candidate solutions, i.e. 400 instead of $2.199 \cdot 10^{12}$ solutions [?]. This work explores the application of PSO to the PMSM speed control problem under dynamic conditions.

4.5.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic search algorithm. It was first introduced by Kennedy and Eberhart in 1995 [31]. Since then, it has been widely used to solve a broad range of optimization problems. The algorithm was presented as simulating animals social activities, e.g. insects, birds, etc. It attempts to mimic the natural process of group communication to share individual knowledge when such swarms flock, migrate, or hunt. If one member sees a desirable path to go, the rest of this swarm will follow quickly. In PSO, this behavior of animals is imitated by particles with certain positions and velocities in a searching space, wherein the population is called a swarm, and each member of the swarm is called a particle. Starting with a randomly initialized population, each particle in PSO flies through the searching space and remembers the best position it has seen. Members of a swarm communicate good positions to each other and dynamically adjust their own position and velocity based on these good positions. The velocity adjustment is based upon the historical behaviors of the particles themselves as well as their companions. In this way, the particles tend to fly towards better and better searching areas over the searching process [31, 34]. The searching procedure based on this concept can be described by 4.5.14.

$$v_i^{k+1} = w * v_i^k + c_1 rand_1 \times (pbest_i - x_i^k) + c_2 rand_2 \times (gbest - x_i^k) \quad x_i^{k+1} = x_i^k + v_i^{k+1} \quad (4.5.14)$$

In 4.5.14, C_1, C_2 are constants, defined as acceleration coefficients; w is the inertia weight factor; $rand_1$ and $rand_2$ are two random functions in the range of $[0,1]$; x_i represents the i th particle and $pbest_i$ the best previous position of x_i , $gbest$ is the particle among the entire population; v_i is the rate of the position change(velocity) for particle x_i ; $gbest$ is the best particle among the entire population; v_i is the position change(velocity) for particle x_i . Velocity changes in (5.5a). comprise three parts, i.e. the momentum part, the cognitive part, and the social part. This combination provides a velocity getting closer to $pbest$ and $gbest$. Every particles current position is then evolved according to 4.5.14, which produces a better position in the solution space. Figure 4.5.10 is a conceptual illustration of this searching process according to 4.5.14.

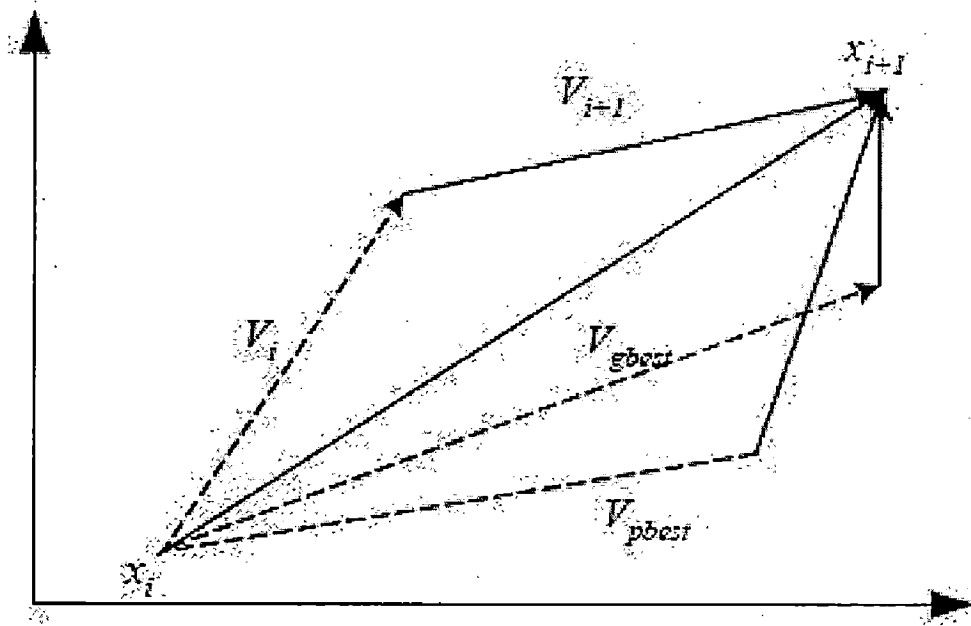


Figure 4.5.10: Vector space representation of PSO

According to 4.5.14, several factors impact the performance of the PSO algorithm, i.e. the inertia weight factor w and the two acceleration coefficients, c_1 and c_2 . Since the introduction of the PSO method in 1995, a considerable amount of work have been done in improving the original version of PSO by varying these three factors. Using different or time varying inertia weight factor can balance the local and global search during the optimization process. The acceleration constants serve dual purposes in this algorithm. First, they control the relative influence toward $gbest_i$ and $pbest_i$ respectively. Second, the two acceleration coefficients com-

bined form what is analogous to the step size of an adaptive algorithm. Acceleration coefficients closer to zero will produce fine searches of a region, while coefficients closer to one will result in lesser exploration and faster convergence. Setting the acceleration greater than one allows the particle to possibly overstep g_{best} and p_{best} , resulting in a broader search. Further discussion about the proper parameter setting can be found in [32]-[35] and their references.

4.5.3 PSO Algorithm

In fully exploring the search region, one wants each particle (each bird in the preceding example) to have a certain degree of randomness in its search. Also, the particle should be able to approach the optimum. Particles are then coded with speed and position only. Speed determines the search direction. Position determines its fitness value by evaluating the objective function. PSO is easy to implement. There are very few parameters that need to be specified. Studies also indicate that PSO is robust and quickly solves optimization problems. PSO is first initialized with a swarm of particles. These initial particles are generally randomly assigned. The position of each particle represents a feasible solution. Then, PSO searches for the optimal solution by updating its generations. In every generation (iteration), each particle is compared with two "best" values, p_{best} and g_{best} . p_{best} is the best solution (with the greatest fitness value of the corresponding objective function) a particular particle has achieved so far. g_{best} is the global best that any particle has achieved. The particle then updates its position and velocity with the following pseudo code:

1. Initialize x_{id} and V_{id} , for all i, d ;
2. Let $p_{best_i} = x_i, g_{best_i} = \min_{x_j} f(x_j)$ for all i (Initialize p_{best_i} and g_{best_i});
3. For $i=1$ to n Do
4. For $d = 1$ to m Do
5. $x_{id} = x_{id} + V_{id}$
6. $V_{id} = w * V_{id} + c_1 xrand_1() * (p_{best_{id}} - x_{id}) + c_2 xrand_2() * (g_{best_d} - x_{id})$;
7. End Do
8. If $f(x_i) < f(p_{best_i})$, then $p_{best_i} = x_i$;
9. If $f(x_i) < f(g_{best_i})$, then $g_{best_i} = x_i$;
10. End Do

where

- $i = 1$ to n , the index number of particles that comprise the "swarm",
- $d = 1$ to m , the index number of the dimension of a particle,
- $f = f(x)$, the objective function,
- X_i = the position of particle i
- V_t = the velocity of particle,
- w = the inertia weight,
- c_1, c_2 = learning factors,
- $rand_1(), rand_2()$ = random scalars that fall between 0 and 1,
- $pbest_{id}$ = the the particle best for the d th dimension of the i th particle,
- $gbest_d$ = the d th dimension of the global best.

The number of particles in the group typically ranges from 20 to 40. A group size of 10 is generally large enough to achieve good results. For some particularly complicated optimization problems, a larger group size, say 200, can be applied. Increasing the group size will accelerate the convergence speed and the probability of finding a global optimum; however, a larger group size normally takes a longer time to compute. Furthermore, a larger group may not be suitable for real-time optimization. In this research, the group size is set to 10.

Particle dimension indicates the number of elements that compose a particle. It is determined by the problem to be optimized. For this research, the particle dimension is set to the number of parameters that will be tuned for the FLC. Such parameters include high and low limits of fuzzy membership functions and the weights for individual fuzzy rules. Particle velocity indicates how fast a particle will move. This variable is an m -dimensional vector, where m is the particle dimension. To search effectively, particles' velocities are confined to predefined maximum values V_{max} . The maximum velocity is then set for each of its dimension (variable to be optimized). A high V_{max} enables particles to fully explore the search region. However, with too high a V_{max} , particles will be moving too fast and will easily pass the optimum solution. A low V_{max} will allow particles to easily catch the local optimum. However, a low V_{max} may result in incomplete exploration of the search region and may confine the searching to a local optimum rather than a global optimum. Suppose that $[A, B]$ is the feasible range defining the search region of the j th dimension of particle i ; the maximum velocity will then generally be defined as $-(A-B)$.

The inertia weight w is applied to weight the impact of the history of velocities on the current velocity, thus affecting the equilibrium between global (wide-ranging) and local (nearby) exploration abilities of the particles (Shi and Eberhart, 1998). A large value of w favors global searching (exploring new areas), while a small w tends to facilitate local searching that finds better solutions within the current search area. A proper inertia weight helps to maintain the balance between global searching and local searching abilities. Thus, the PSO will require a smaller number of iterations, or fast convergence. A study by Shi and Eberhart (1998)[50] indicates that when V_{max} is relatively large, an inertia weight $w = 0.8$ is a good choice. When V_{max} is relatively small, an inertia weight of approximately 1 is a good choice.

Learning constant c_1 is the factor that indicates how much a particle is affected by its historical personal (particle) best position (solution). Learning constant c_2 indicates how much the particle is affected by the global (group) best position within the whole group. Under most conditions, c_1 and c_2 range from 0 to 4. Generally, $c_1=c_2=1.4$, but this need not to be the case. The best particle position, p_{best} , represents the individual best solution a particular particle has ever found. For a group of particles, the value of p_{best} is equal to the number of particles. The best global position, g_{best} , is the best solution that has ever been found by the whole group (swarm). Accordingly, there is only one g_{best} in every iteration. Normally, an exit condition is activated when PSO either finds a solution that meets the minimum error requirement, or when the maximum number of iterations is reached. In this research, PSO will not exit until it reaches its maximum number of iterations. This is because the objective function minimizes the speed error in the PMSM. Intuitively, there will not be any particle that can achieve a error of 0. As a result, the global optimum is used as the optimal solution for the objective function after a whole set of iterations.

$$x_{id} = x_{id} + V_{id}V_{id} = X * V_{id} + c_1 * rand_1() * (p_{best_{id}} - x_{id}) + c_2 * rand_2() * (g_{best_d} - x_{id}) \quad (4.5.15)$$

Eqn 3.1.11 updates a particle's position after every iteration. Equation 3.1.12 updates a particle's speed after every iteration. Equation 3.1.11 is easily understood. Equation 3.1.12 has three components. Component 1 expresses the inertia of particles. Component 2 can be explained by Thorndike's law of effect (1927)[49], which indicates that a "correct" random action is more likely to be repeated in the future. Here, the cognitive action stresses the action of moving toward the "correct" solution. This component represents the learning of individual particles. Such learning stimulates the particle to move toward a better position. Component 3 can be explained by Bandura's vicarious reinforcement and imitative learning (1963)[48], which indicates that the probability of a certain action will be stressed once the result of that action leads to a better solution. In other words, the cognition of a particular member will be emulated

by other members if this member's action produces a better solution.

4.5.4 Objective Function

After the PSO creates a population of solutions for the FLC parameters, the fitness value of each solution is calculated using the objective function of the torque based on speed and its change. The Objective Function is given as follows

$$\begin{aligned} speed1 &= ge * e^{ge*10} + 10 * overshoot \\ speed2 &= gce * e^{gce*10} + 10 * overshoot \\ speed3 &= gu * e^{gu*10} + 10 * overshoot \end{aligned} \quad (4.5.16)$$

Hence, Fitness Function

$$F1 = \frac{1}{1 + speed} \quad F2 = \frac{1}{1 + speed} \quad F3 = \frac{1}{1 + speed} \quad (4.5.17)$$

The optimization is performed every 20 seconds and the FLC parameters are updated in the same timely manner. A default FLC parameter set is selected to be used as the initial parameters for the very first simulation interval. PSO then attempts to perform optimization. At the beginning of every other simulation interval, the PSO is initialized with several randomly selected FLC parameter sets as its particles. The fitness value of each FLC parameter set is then calculated according to the objective function. PSO then updates its global best and particle best for the next iteration. After the PSO reaches its maximum iteration, the optimization is finished and the parameter set with the highest fitness value is selected as the FLC parameter set to be applied in the next simulation interval. The PSO is stopped at the end of the simulation.

Fig. 4.5.11 shows the optimization of fuzzy parameters in the PMSM Model.

4.6 Type-2 Fuzzy Logic Control Approach

Permanent magnet synchronous motor drives are highly efficient, high speed machines and they have simple mechanical construction and possess high reliability. However the performance is affected by the uncertainties in the drive system and also load disturbances due to the growing demands of consumer power consumption. They should operate even under conditions of high uncertainties and load disturbances. Traditional control strategies have already been developed to handle such uncertainties. But they lack high precision and under very high load disturbances they cannot handle uncertainties.

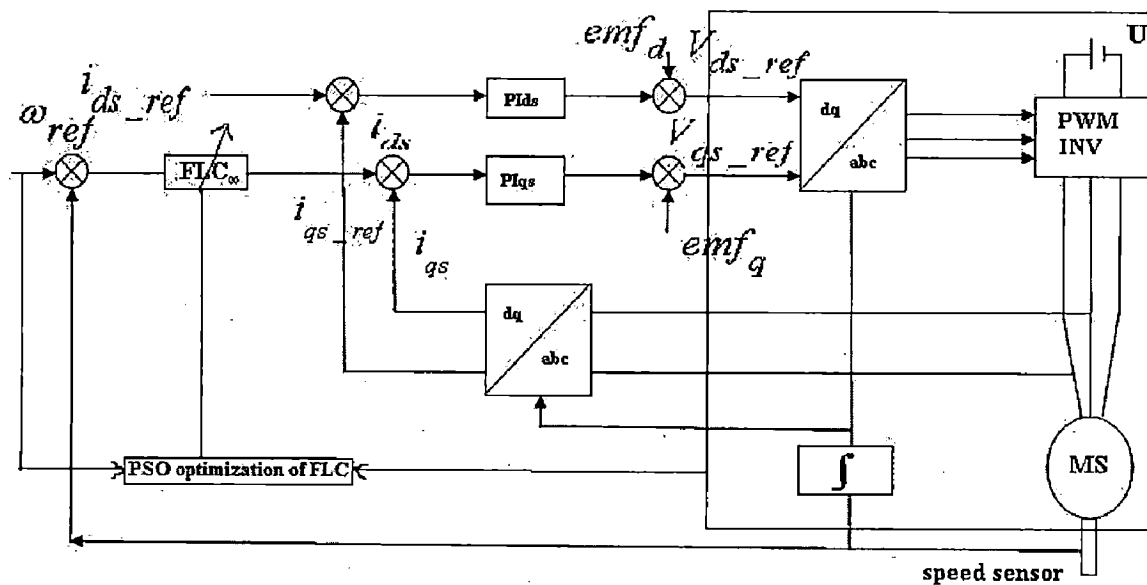


Figure 4.5.11: PSO optimization of FLC in Permanent Magnet Synchronous Motor Control

Fuzzy Logic controller is such a traditional controllers using expert decision making rules and with a powerful reasoning capacity. The fuzzy sets were presented by L.A.Zadeh in 1965[18] to process/manipulate data and information affected by unavoidable uncertainty/imprecision. They were designed to mathematically represent the vagueness and uncertainty of linguistic problems;there by obtaining formal tools to work with intrinsic imprecision indifferent type of problems;it is considered a generalization of the classic set theory. Intelligent Systems based on fuzzy logic are fundamental tools for non linear complex system modeling.The fuzzy sets and fuzzy logic are the base for fuzzy systems,where their objective has been to model how the brain manipulates inexact information such as an uncertainty in system. The uncertainty here is the measurement and process uncertainty of PMSM. Uncertainty in the antecedent part of fuzzy occurs due to the sensor measurements which typically contains some amount of noise due to atmospheric conditions which is given as input.The uncertainties at the consequents of the fuzzy occur due to the external load disturbances.

The traditional Type-1 FLC's cannot fully handle the uncertainties associated with linguistic and numerical reasoning problems under changing dynamic disturbances.Type-1 FLC handles the uncertainties associated by using precise and crisp membership functions. Once the Type-1 MF has been chosen, all the uncertainty disappears, because Type-1 membership functions are totally precise[15] and become certain.The designed Type-1 Fuzzy sets may not be precise with the associated uncertainties.It may cause the degradation in the PMSM's control.

Type-2 fuzzy sets are used for modeling uncertainty and imprecision in a betterway.These

Type-2 fuzzy sets were originally presented by Zadeh in 1975 and are essentially "fuzzy fuzzy" sets where the fuzzy degree of membership is a Type-1 fuzzy set[18, 30]. The new concepts were introduced by Mendel and Liang[16, 27] allowing the characterization of a Type-2 fuzzy set with a superior membership function and an inferior membership function; these two functions can be represented each one by a Type-I fuzzy set membership function. The interval between these two functions represents the foot print of uncertainty(FOU), which is used to characterize a Type-2 fuzzy set of the inputs and output of PMSM. Type-2 FLC have grades of membership function that are themselves fuzzy. At each value of a primary variable (Speed, Torque, Current) the membership is a function. The secondary membership function whose domain is the primary membership interval. Hence, the membership function is a three dimensional and Type-2 uses the third dimension that provides new design degrees of freedom for handling the various disturbances and uncertainties.

4.7 Interval Type-2 Fuzzy Set Theory

4.7.1 Type-2 Fuzzy Sets

A type-2 fuzzy set [26, 25] expresses the non-deterministic truth degree with imprecision and uncertainty for an element that belongs to a set. A type-2 fuzzy set denoted by \tilde{A} , is characterized by a type-2 membership function $\mu_{\tilde{A}}(x, u)$, where $x \in X, u \in J_x^u \subseteq [0, 1]$ and $0 \leq \mu_{\tilde{A}}(x, u) \leq 1$ defined in equation(12)[15].

$$\tilde{A} = (x, \mu_{\tilde{A}}(x)) \mid x \in X \quad \tilde{A} = (x, u, \mu_{\tilde{A}}(x, u)) \mid x \in X, \forall u \in J_x^u \subseteq [0, 1] \quad (4.7.18)$$

\tilde{A} can also be expressed as follows[21]:

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x, u) / (x, u) J_x^u \subseteq [0, 1] \quad (4.7.19)$$

where $\int \int$ denotes union over all admissible x and u [21]. J_x is called primary membership of x , where $J_x \subseteq [0, 1] \forall x \in X$ [21]. The uncertainty in the primary memberships of a type-2 fuzzy set \tilde{A} , consists of a bounded region that is called the footprint of uncertainty(FOU). It is the union of all primary memberships [21]. Recently, it has been shown that regardless of the choice of the primary membership function (triangle, Gaussian, trapezoid), the resulting FOU is about the same[22]. According to [22], the FOU of a type-2 membership function also handles the rich variety of choices that can be made for a type-1 membership function, i.e., by using type-2 fuzzy sets instead of type-1 fuzzy sets, the issue of which type-1 membership function to choose diminishes in importance. For type-2 fuzzy sets there are new operators named the

meet (denoted by \sqcap) and join (denoted by \sqcup) to account for the intersection and union[27]. According to [28], a type-2 fuzzy set can be thought of as a large collection of embedded type-1 sets each having a weight to associated with it.

At each value of x say $x = x'$, the 2-D plane whose axes are u and $\mu_{\tilde{A}}(x', u)$ is called a vertical slice of $\mu_{\tilde{A}}(x, u)$. It is $\mu_{\tilde{A}}(x = x', u)$ for $x' \in X$ and $\forall u \in J_{x'} \subseteq [0, 1]$. A secondary membership function is a vertical slice of $\mu_{\tilde{A}}(x, u)$. It is $\mu_{\tilde{A}}(x = x', u)$ for $x' \in X$ and $\forall u \in J_{x'} \subseteq [0, 1]$ [18], i.e.,

$$\mu_{\tilde{A}}(x = x', u) \equiv \mu_{\tilde{A}}(x') = \int_{u \in J_{x'}} f_{x'}(u) / (u) J_{x'} \subseteq [0, 1] \quad (4.7.20)$$

in which $0 \leq f_{x'}(u) \leq 1$. Because $\forall x' \in X$, the prime notation on $\mu_{\tilde{A}}(x')$ is dropped and $\mu_{\tilde{A}}(x)$ is referred to as a secondary membership function [21]; it is a type-1 fuzzy set which is also referred to as a secondary set [21]. Many choices are possible for the secondary membership functions. According to [20], the name that we use to describe the entire type-2 membership function is associated with the name of the secondary membership functions. For example, when $f_x(u) = 1, \forall u \in J_x \subseteq [0, 1]$ then the secondary membership functions are interval sets, and, if this is true for $\forall x \in X$, we have the case of an interval type-2 membership function which characterizes the interval type-2 fuzzy sets[20]. Interval secondary membership functions reflect a uniform uncertainty at the primary memberships of x [20]. Since all the memberships in an interval type-1 set are unity, in the sequel, an interval type-1 set is represented just by its domain interval, which can be represented by its left and right end-points as $[l, r]$ [25]. The two end-points are associated with two type-1 membership functions that are referred to as upper and lower membership functions which are bounds for the footprint of uncertainty $\text{FOU}(\tilde{A})$ [25]. The upper membership function is associated with the upper bound of $\text{FOU}(\tilde{A})$ and is denoted by $\mu_{\tilde{A}}(x), \forall x \in X$ [20]. The lower membership function is associated with the lower bound of $\text{FOU}(\tilde{A})$ and is denoted by $\mu_{\tilde{A}}(x), \forall x \in X$ [20].

The type-2 fuzzy controller which controls the synchronous motor has many advantages when compared to a conventional type-1 fuzzy controller. Following are some of the advantages: As the type-2 fuzzy sets membership functions are themselves fuzzy and include a FOU, then they can model and handle the linguistic and numerical uncertainties associated with the inputs and outputs of the synchronous motor FLC in changing and dynamic unstructured environments and hence they can handle the difficulty associated with determining the exact membership functions for the fuzzy sets [25]. Therefore, FLCs that are based on type-2 fuzzy sets will have the potential to produce a better performance than the type-1 FLCs. Using type-2 fuzzy sets to represent the FLC inputs and outputs will result in the reduction of the FLC rule base when compared to using type-1 fuzzy sets as the uncertainty represented in the footprint of uncertainty in type-2 fuzzy sets lets us cover the same range as type-1 fuzzy sets with smaller number of labels and the rule reduction will be greater when the number of the FLC inputs

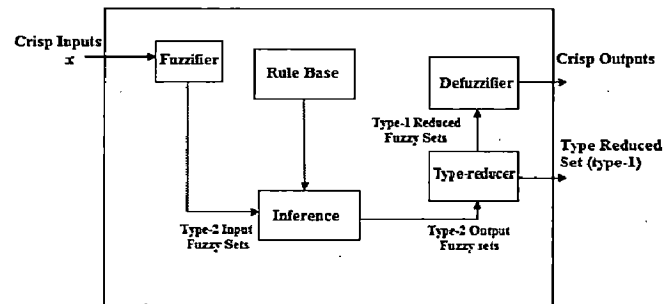


Figure 4.7.12: Type-2 Fuzzy Logic Controller

increases[20]. Each input and output will be represented by a large number of type-1 fuzzy sets which are embedded in the type-2 fuzzy sets[25, 21]. The use of such a large number of type-1 fuzzy sets to describe the input and output variables allows for a detailed description of the analytical control surface as the addition of the extra levels of classification gives a much smoother control surface and response.

4.7.2 Type-2 Fuzzy Logic controller (FLC)

A type-2 FLC contains five components which are fuzzifier, rule base, fuzzy inference engine, type-reducer and defuzzifier. The synchronous motor control uses an Interval type-2 fuzzy set as they are simple to use[21] and they distribute uncertainty evenly among all membership functions[20]. Since the general type-2 FLC is computationally intensive and the computation simplifies a lot when using interval type-2 FLC[21] (using interval type-2 fuzzy sets) which enables us to design a controller for synchronous motor in real time. The type-2 FLC works as follows, the crisp inputs from the input sensors are first fuzzified into, in general, input type-2 fuzzy sets which then activates the inference engine and the rule base to produce output type-2 fuzzy sets. These output type-2 fuzzy sets are then processed by the type-reducer which combines the output sets and then performs a centroid calculation, which leads to type-1 fuzzy sets called the type-reduced sets[20]. The defuzzifier can then defuzzify the type-reduced type-1 fuzzy outputs to produce crisp outputs to be fed to the motor. The uncertainty is handled by the antecedents and consequents of interval type-2 fuzzy sets which include FOU to accommodate the linguistic and numerical uncertainties associated with changing unstructured environments. The interval type-2 fuzzy sets also include a large number of embedded type-1 fuzzy sets and thus according to [11] the type-2 FLC can be thought of as a collection of many different embedded type-1 FLCs [11] to deal with the different uncertainties. An interval type-2 FLC can be depicted in fig. 4.7.12

The synchronous motor is controlled using a seven term fuzzy logic controller as shown in the figure 4.7.15. Where NB-negative big, NM-Negative Medium, NS-Negative Small, ZR-Zero, PB-Positive Big, PS-Positive Small. The Interval Type-2 Fuzzy logic system (IT2 FLS) is constructed by a gaussian primary Membership function (MF) with uncertain mean and fixed standard deviation. It can be described as

$$\mu_{\bar{A}}(x) = \exp\left[-\frac{1(x-m)^2}{\sigma^2}\right] \quad (4.7.21)$$

The Type-2 Fuzzy set is in a region called a footprint of uncertainty and is bounded by an upper MF and a lower MF, which are denoted as $\bar{\mu}_{\bar{A}}(x)$ and $\underline{\mu}_{\bar{A}}(x)$

Both the inputs and outputs used the same membership functions in the controller. And we use MIN-MAX method for inferencing and for type reduction we use Center of Sets Method and Defuzzification Method employed is Centroid Defuzzification [17].

4.7.3 Seven Term Fuzzy Logic Controller

The conventional Type-1 Fuzzy Controller using the seven term rule base is shown in the figure 4.7.15.

CONTROL CHANGE		ERROR CHANGE						
		NB	NM	NS	ZR	PS	PM	PB
ERROR	NB	NB	NB	NB	NB	NM	NS	ZR
	NM	NB	NB	NB	NM	NS	ZR	PS
	NS	NB	NB	NM	NS	ZR	PS	PM
	ZR	NB	NM	NS	ZR	PS	PM	PB
	PS	NM	NS	ZR	PS	PM	PB	PB
	PM	NS	ZR	PS	PM	PB	PB	PB
	PB	ZR	PS	PM	PB	PB	PB	PB

Figure 4.7.13: Seven Term Type-1 Fuzzy Logic Rule Base

where NB-Negative Big, NM-Negative medium, NS-Negative Small, ZR-Zero, PB- Positive Big, PM-Positive Medium, PS-Positive Small. It uses triangular membership functions (MFs). Both the Inputs and Outputs use same MFs in the controller. The type-2 Foot print of Uncertainty can be obtained by taking the base end point values of the type-1 MFs and adding an arbitrary +/-10 perc to give uncertainty intervals associated with them. The type-2 FLC uses gaussian MFs for both the input error, change in error and the outputs.

The type-2 FLC is shown in figure 4.7.16

4.7.4 FLC Inference and Defuzzification

The FLCs uses the MIN rule connection method, the MIN inference action method and the MAX aggregation Method. For defuzzification, the Type-1 FLCs used is a centre of gravity

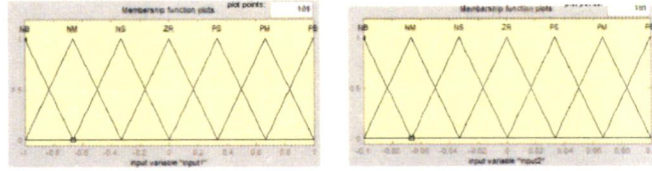


Figure 4.7.14: Inputs of type 1 FLC

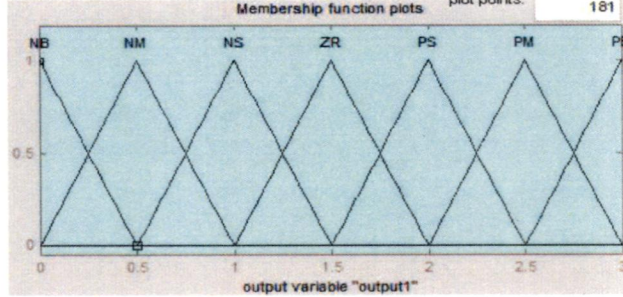


Figure 4.7.15: Output of Type-1 FLC

method to generate crisp values of output and the Interval Type-2 FLC used is a Centroid Of Gravity Method[17], and a crisp value was obtained by averaging the upper and lower interval values.

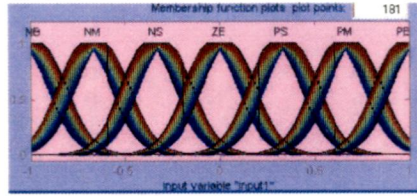
4.8 Description of the system

To investigate the effectiveness of the Type-2 FLC, a second-order system transfer function with the following prescribed characteristics: 0.3 s rise time, no steady-state error and unity damping ratio to avoid overshoot is chosen as the reference model for the periodic step command. Therefore, in accordance with [29], the following transfer function is chosen in this study:

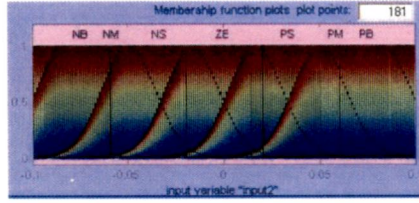
$$G(s) = \frac{168.11}{s^2 + 25.93s + 168.11} \quad (4.8.22)$$

The PMSM used in this drive system is a three-phase four-pole 750 W, 3.47 A, 1000 rpm type. By using the field-oriented technique, the PMSM drive can be reasonably represented by the following equations [29]:

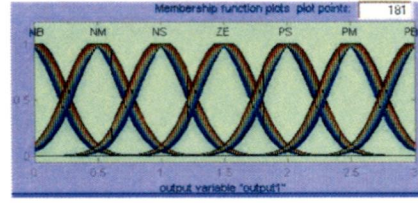
$$T_e = K_t * i_{qs} H(s) = \frac{1}{Js + B} = \frac{b}{s + a} \quad (4.8.23)$$



(a) input1



(b) input2



(c) output

Figure 4.7.16: Type-2 FLC gaussian membership functions

where the detailed parameters of the system scale are
 $K_t=0.6732 \text{ nm/a}$, $a=4.4$, $b=15.2$, $\bar{j} = 0.066 \text{ Nmsrad/v}$
 $\bar{B} = 0.289 \text{ Nm/V}$

The schematic diagram of the speed control system under study is shown in figure 4.8.17 using Type-2 FLC and figure 4.8.18 shows the Type-1 FLC. The power circuit consists of a continuous voltage supply which can be provided by a six rectifier thyristors and a three phase GTO thyristors inverter whose output is connected to the stator of the synchronous machine [14]. The field current i_f of the synchronous machine, which determines the field flux level is controlled by voltage v_f . The self control operation of the inverter-fed synchronous machine results in a rotor field oriented control of the torque and flux in the machine as mentioned in section 2. The flux in the machine is controlled independently by the field winding and the torque is affected by the i_f . The parameters of the synchronous machine are given in the Figure 4.8.17 shows the schematic diagram of the speed control of synchronous motor using the Type-2 Fuzzy Logic System.

4.9 Synchronous Motor control using Type-2 FLC

The Rule base of the Type-2 FLC is shown in figure 4.9.20 which shows the firing of different rules. The surface viewer is shown in figure 5.1.4. It shows variation of the control output for varying inputs applied based on the firing of the given fuzzy rules. The speed and the three phase current of the synchronous motor using a Type-2 FLC is shown in figure 6.1.12. There are no overshoots in the speed and the rise time is less and the speed settles into a steady

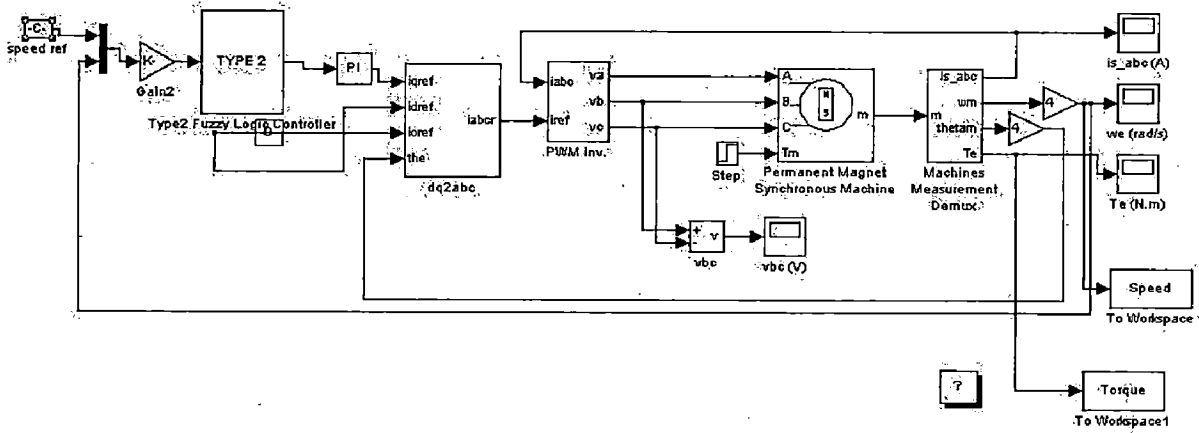


Figure 4.8.17: Schematic diagram of speed control of PMSM using Type-2 Fuzzy Logic

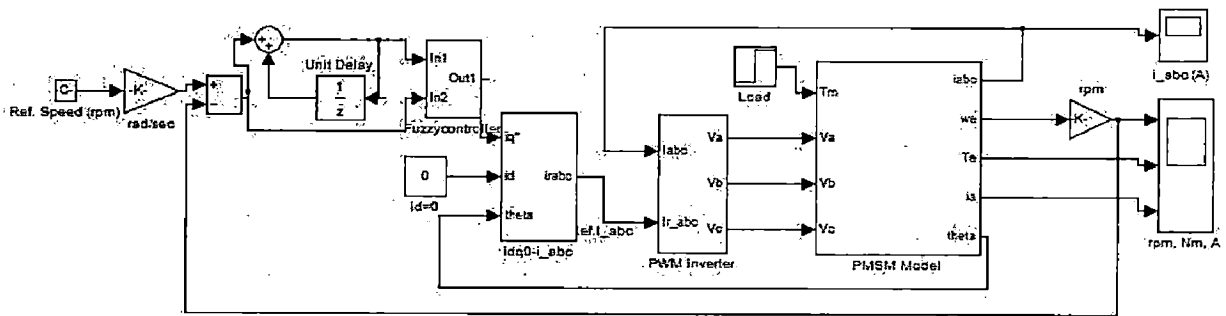


Figure 4.8.18: Schematic diagram of speed control of PMSM using Type-1 Fuzzy Logic

state position in a very less time period. The Three phase stator currents shows that at the instant of starting, maximum current is drawn from the supply system and the permanent magnet synchronous motor easily maintains it steady state and hence using the Type-2 FLC the stability is fastly achieved to the PMSM motor.

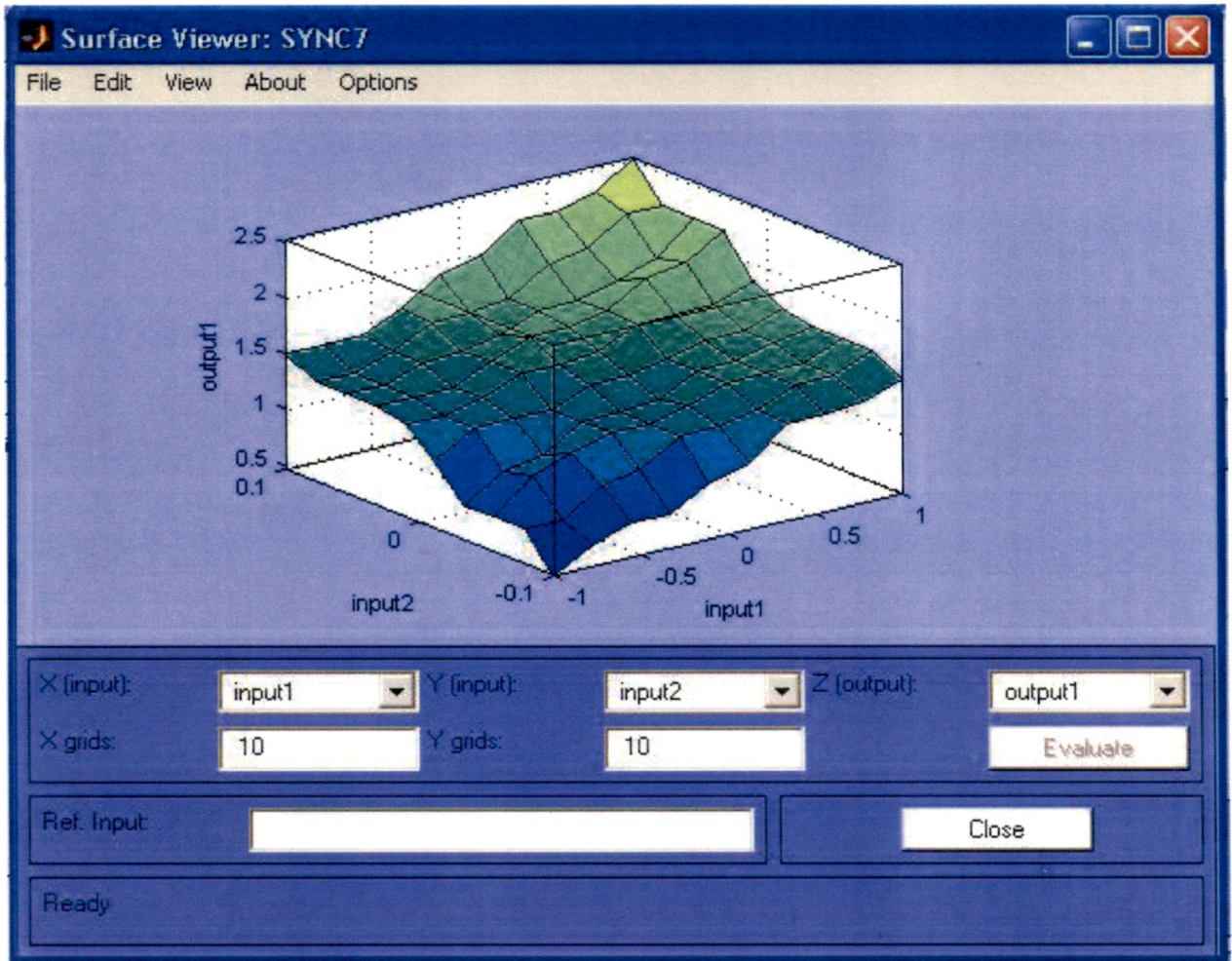


Figure 4.9.19: Surface Viewer for Type-2 FLC

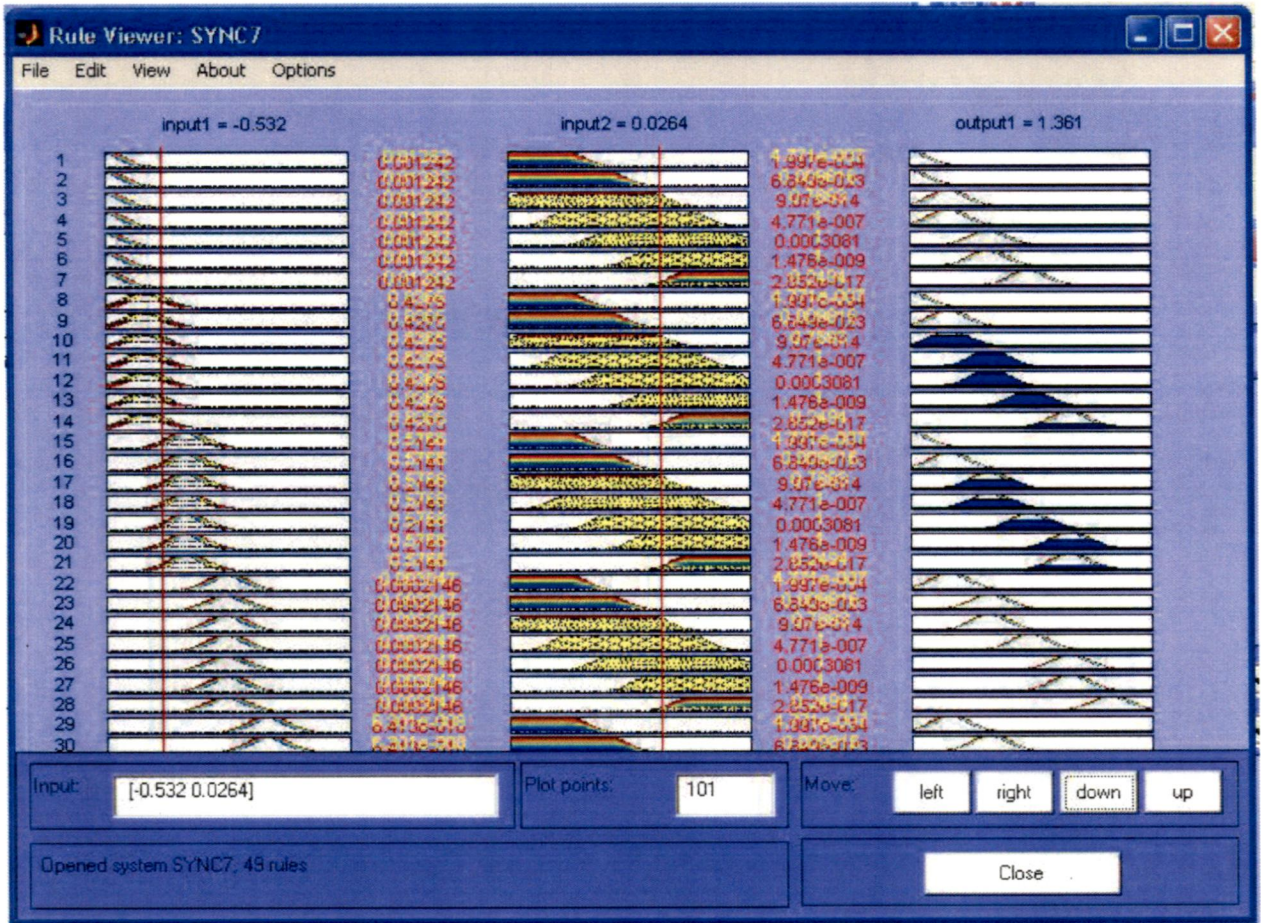


Figure 4.9.20: Firing of different rules in a Type-2 FLC

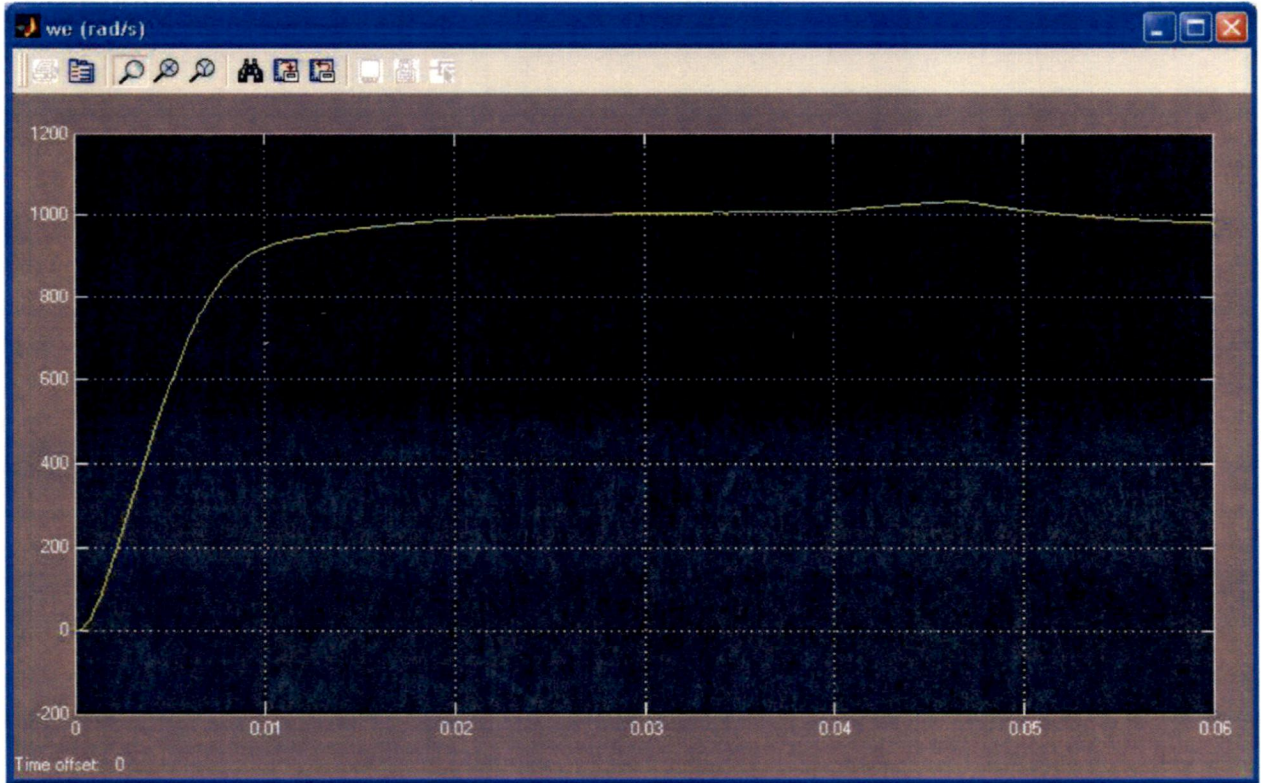


Figure 4.9.21: Speed Control of PMSM using Type-2 FLC

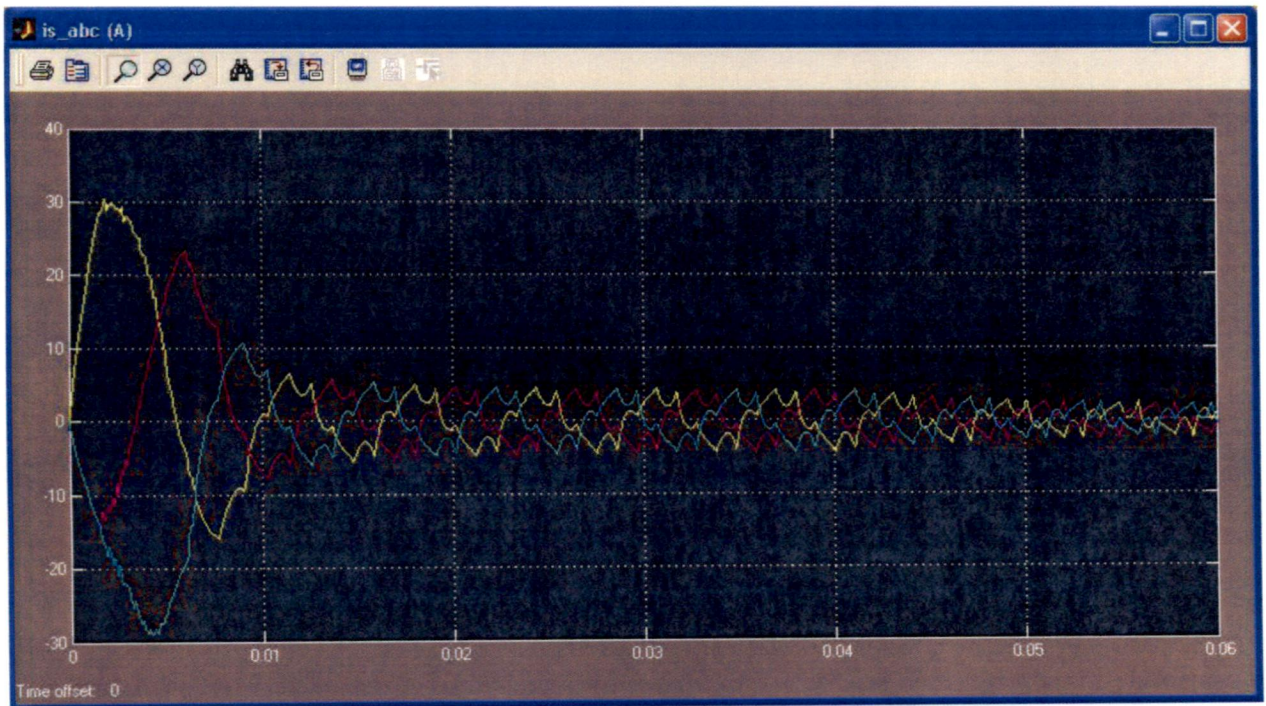


Figure 4.9.22: Three Phase stator Current of PMSM using Type-2 FLC

Chapter 5

Adaptive Networks: Architectures and Learning Algorithms

The fuzzy model under the framework of adaptive networks is called Adaptive Neuro-Fuzzy Inference System (ANFIS) [43]. The adaptive neural networks have the advantages of being able to learn, and adapt to the system. Fuzzy logic is a rule based method that is framed with the expertise of human knowledge. Thus, ANFIS combines the advantages of adaptive neural networks and fuzzy logic.

5.1 Adaptive Neuro Fuzzy Inference System

Adaptive neural-fuzzy inference system ANFIS was first introduced by J. Jang in 1993 [53]. The model considered here is based on Takagi-Sugeno inference model [51, 52]. ANFIS uses a hybrid learning algorithm to identify consequent parameters of Sugeno-type fuzzy inference systems. It applies a combination of the leastsquares method and backpropagation gradient descent method for training fuzzy inference system membership function parameters to emulate a given training data set. The neuro-fuzzy system owes much from the feedforward neural network with supervised learning capability. The fuzzy inference system under consideration has two inputs speed and load torque and disturbance (load parameters) and produce the value of speed separately.

The ANFIS uses five membership functions as shown in figure 5.1.3. These memberships characteristics are trained by ANFIS to provide the previously calculated gain values for each cases of values of (speed) given the specific load characteristics (speed,torque,current) as inputs.The figure 5.1.4represents the output surface for the ANFIS controller.

The output of the fuzzy system for approximation of P and K is trained by the ANFIS in five layers of networks as follows:The anfis model is shown in the figure 5.2.7.

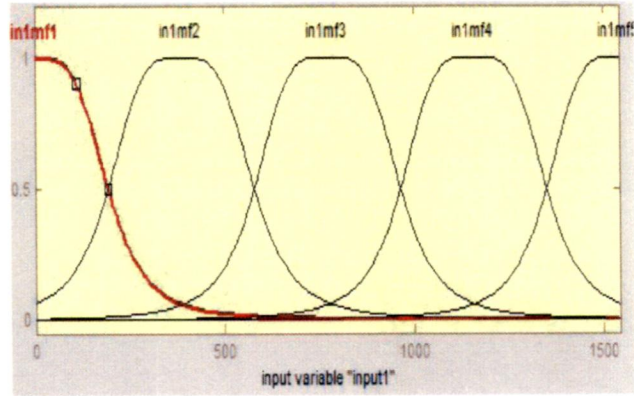


Figure 5.1.1: Membership Function for input1

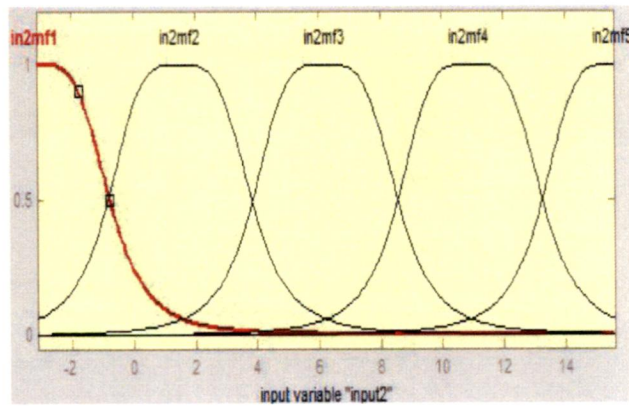


Figure 5.1.2: input2

5.1.1 The first layer

This layer is a basic Fuzzification layer where the crisp inputs are allocated relative fuzzy, values. Bell shaped fuzzy memberships are utilized. The output of the layer one for i^{th} membership function is calculated as:

$$O_i^1 = \mu A_i(x) = e^{-(\frac{x-c_i}{a_i})} \quad (5.1.1)$$

where n_p (or n_q) is the load model parameter. The parameters a_i , b_i and c are premise parameters derived from the characteristics of the bell-shaped membership functions as shown in the Fig. 5.1.6. These premise parameters are updated in the adaptation process using gradient decent method in the backward pass.

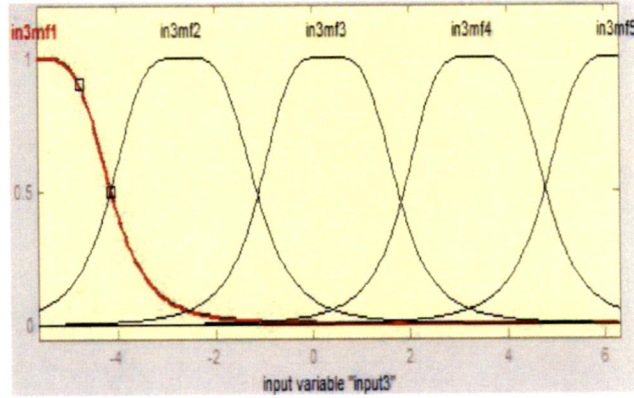


Figure 5.1.3: input3

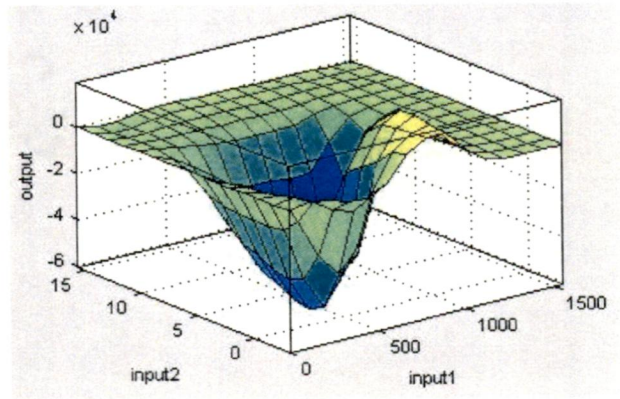


Figure 5.1.4: Surface viewer of the ANFIS controller

5.1.2 The second layer

The outputs of the nodes labeled p in layer two are a result of multiplication of inputs from the layer one nodes.

$$O_i^2 = w_i = \mu_{A_i}(X)\mu_{B_i}(Y) \quad i = 1, 2 \quad (5.1.2)$$

In this case the value of the nodes output represents the strength of the rule.

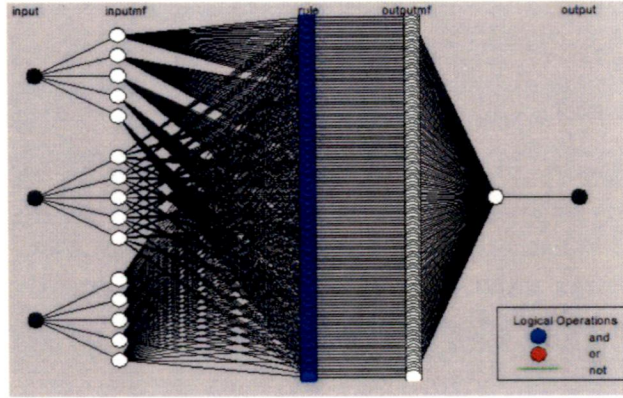


Figure 5.1.5: ANFIS architecture for three input single output.

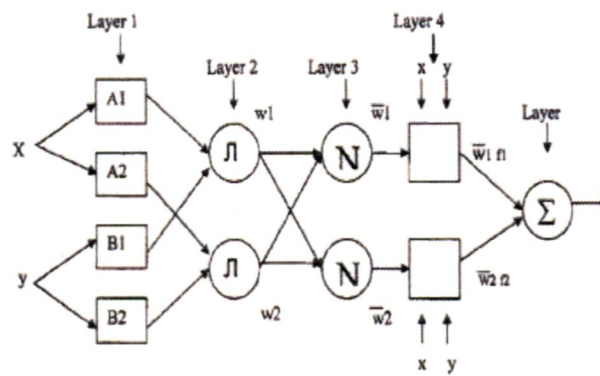


Figure 5.1.6: ANFIS architecture

5.1.3 The third layer

The nodes in this layer are represented by circular nodes labeled N. The outputs of these nodes are basically the ratios of the i th output of the previous layer to the sum of all output of the previous layer.

$$O_i^3 = \bar{w}_i = \frac{w_i}{w_1 + w_2} \quad i = 1, 2 \quad (5.1.3)$$

5.1.4 The fourth layer

This layer produces the defuzzified Takagi and Sugeno-type output for each previous i th output

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i (p_i + q_i + r_i) \quad i = 1, 2 \quad (5.1.4)$$

where p_i, q_i, r_i are consequent parameters

5.1.5 The fifth layer

The single node in this layer computes the overall outputs as the summation of all incoming signals, i.e.,

$$O_i^5 = \text{overalloutput} = \sum_i \bar{w}_i f_i = \frac{\sum_i \bar{w}_i f_i}{\sum_i w_i} \quad i = 1, 2 \quad (5.1.5)$$

Therefore, the ANFIS output is clearly a linear function of the adjustable defuzzifier parameters.

5.1.6 Hybrid Learning Algorithm

The premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output f in fig. 5.1.6 can be rewritten as

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 = \bar{w}_1 f_1 + \bar{w}_2 f_2 = (\bar{w}_1 x) p_1 + (\bar{w}_1 y) q_1 + (\bar{w}_1) r_1 + (\bar{w}_2 x) p_2 + (\bar{w}_2 y) q_2 + (\bar{w}_2) r_2 \quad (5.1.6)$$

which is linear in the consequent parameters $(p_1, q_1, r_1, p_2, q_2, r_2)$. As a result, we have $S =$ set of total parameters $S_1 =$ set of premise parameters $S_2 =$ set of consequent parameters

More specifically, in the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. 5.1 summarizes the activities in each pass[47].

The Consequent parameters thus identified are optimal under condition that the premise parameters are fixed. Accordingly the hybrid approach is much faster than the strict gradient descent and it is worthwhile to look for the possibility of decomposing the parameter set in the manner of (10). For type-1 M, this can be achieved if the membership function on the consequent part of each rule is replaced by a piecewise linear approximation with two consequent parameters (see Fig. 5.1.6). In this case, again, the consequent parameters constitute set S_2 and the hybrid learning rule can be employed directly. However, it should be noted that the computation complexity of the least squares estimate is higher than that of the gradient descent.

	Forward Pass	Backward Pass
Premise Parameters	Fixed	Gradient Descent
Consequent Parameters	Least Squares Estimate	Fixed
Signals	Node Outputs	Error rates

5.2 Modeling of the synchronous motor using the ANFIS

ANFIS is a fuzzy system and used in classification, modeling and control problems. It is based on Takagi and Sugeno model fuzzy if-then rules representation [54], which is different from commonly used fuzzy logic controllers [54, 55]. The consequent part of the rule is a function of input variables. The system considered in this paper has three inputs speed(ω), torque(t_e) and current(i_a) and the output is current(i_{qs}). The inference mechanism of ANFIS is mathematically expressed by the set of the rules. These rules are generated through the experience of operating the system, which may be feedback from the plant operator, design engineer, or the expert. The k^{th} rule is generally expressed in the form (If premise THEN consequence). Fig. ?? shows the synchronous motor control using ANFIS.

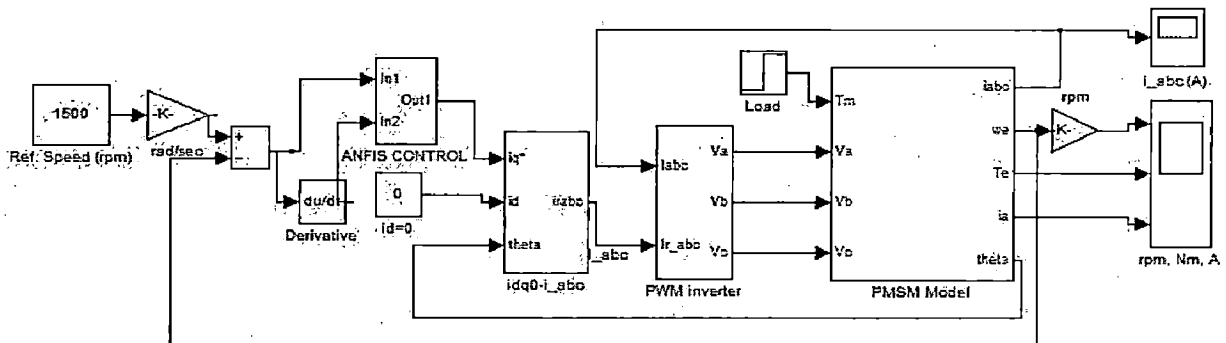


Figure 5.2.7: Synchronous Motor Control using Anfis.

Chapter 6

Results and Discussion

6.1 Model Reference Neural Network Controller

To investigate the effectiveness of the proposed adaptive neural speed controller, a second-order system transfer function with the following prescribed characteristics: 0.3 s rise time, no steady-state error and unity damping ratio to avoid overshoot is chosen as the reference model for the periodic step command. Therefore, in accordance with [39], the following transfer function is chosen in this study:

$$G(s) = \frac{168.11}{s^2 + 25.93s + 168.11} \quad (6.1.1)$$

The PMSM used in this drive system is a three-phase four-pole 750 W 3.47 A 1500 rpm type. By using the field-oriented technique, the PMSM drive can be reasonably represented by the following equations [14]:

$$T_e = K_t * i_{qs} H(s) = \frac{1}{Js + B} = \frac{b}{s + a} \quad (6.1.2)$$

where the detailed parameters of the system scale are

$K_t=0.6732 \text{ Nm/A}$, $a=4.4$, $b=15.2$, $\bar{j} = 0.066 \text{ Nm/rad/v}$, $\bar{B} = 0.289 \text{ Nm/V}$ The simulation is realized using the SIMULINK software in MATLAB environment. Figure 6.1.12 shows the performances of the neural MRC controller.

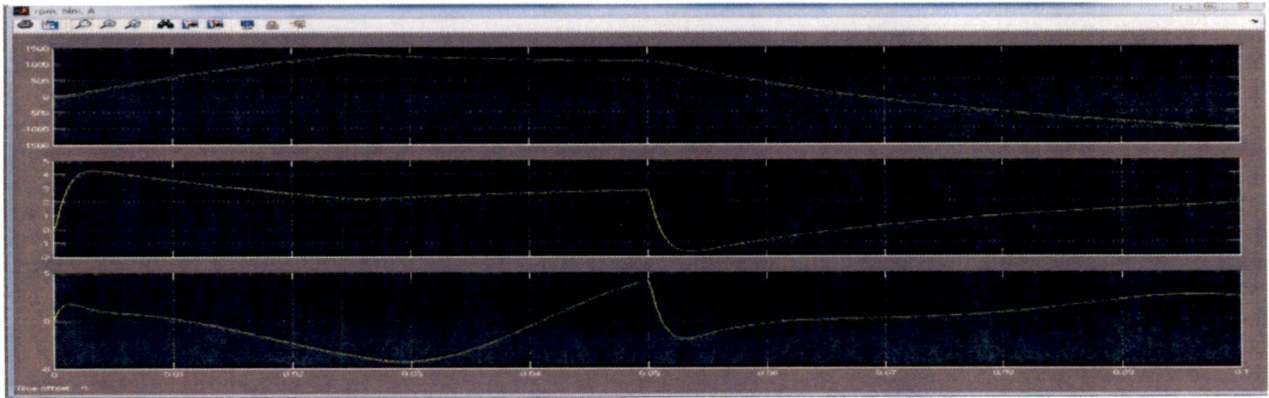


Figure 6.1.1: a.Speed b.Torque C.stator current

To illustrate the performances of control, we have simulated the starting mode of the motor without load, and the T_l)at the instance $t = 2$ s and its elimination at t application of the load ($T = 2\text{Nm}$) = 3 s; in presence of the variation of parameters considered (the moment of inertia) with speed step of +100 rad/s.

The robustness of the system can be shown in the Figure 6.1.2 and the torque figure(6.1.2) after the application of load at instants $t= 0.45,1.6,2.1$ with loads $T_l=3\text{Nm},5\text{Nm},6\text{Nm}$.

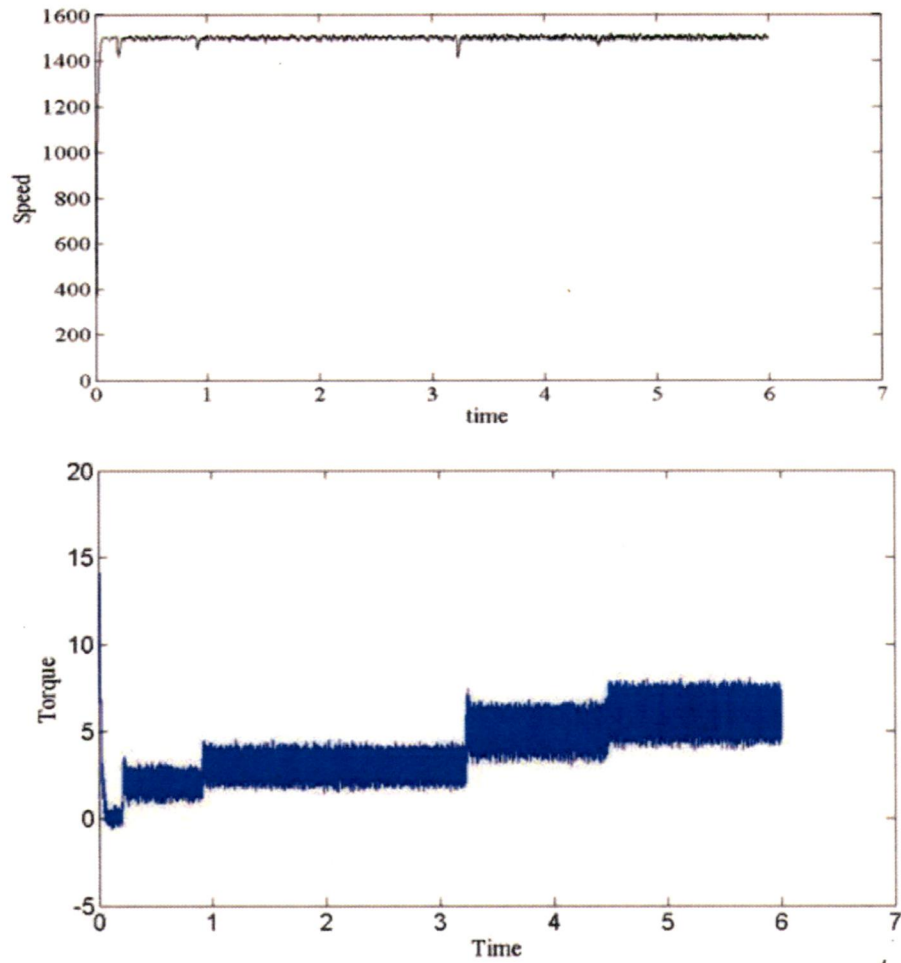


Figure 6.1.2: speed and Torque

6.1.1 Fuzzy Controller

The fuzzy scaling parameters used for control of the synchronous motor are $g_e=0.1$, $g_{ce}=0.4$ and $g_u=30$. The output of fuzzy logic controller for load torque of 4 N-m are shown in Fig. 6.1.6. The fuzzy surface Viewer is shown in fig. 6.1.3.

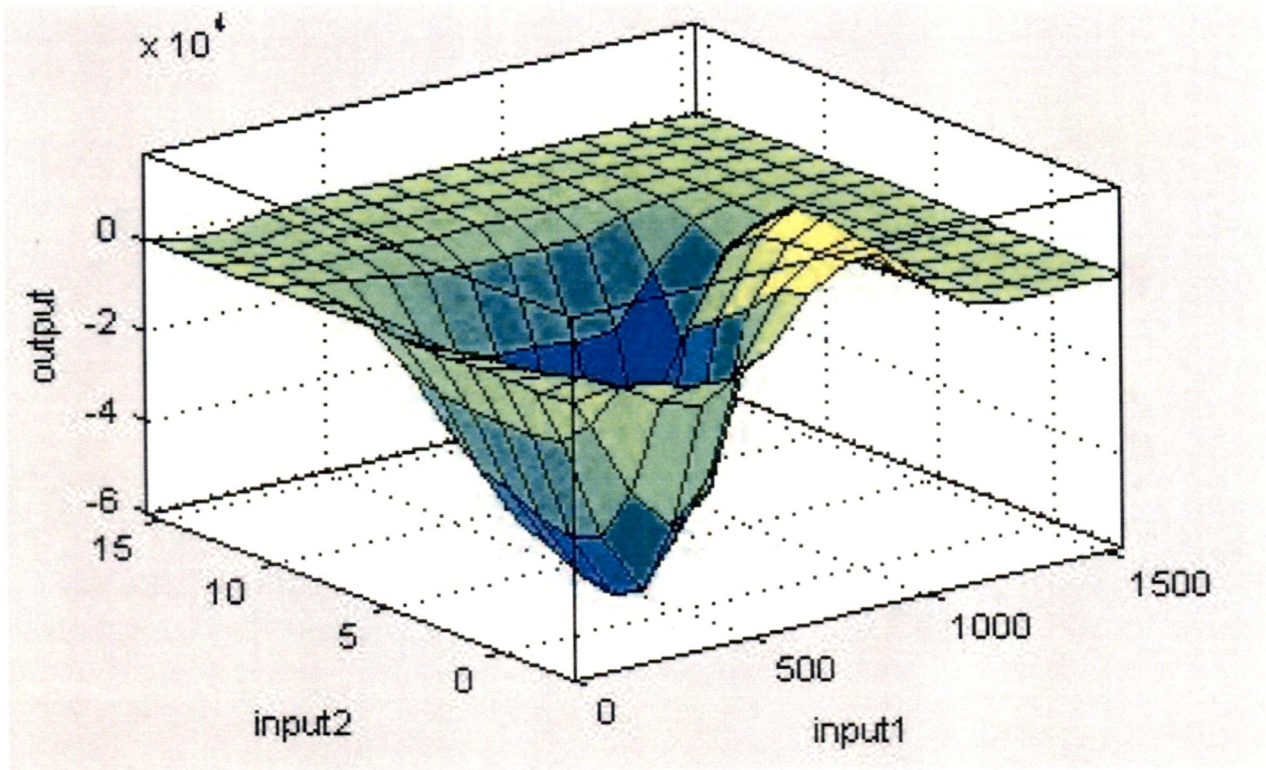


Figure 6.1.3: Surface Viewer

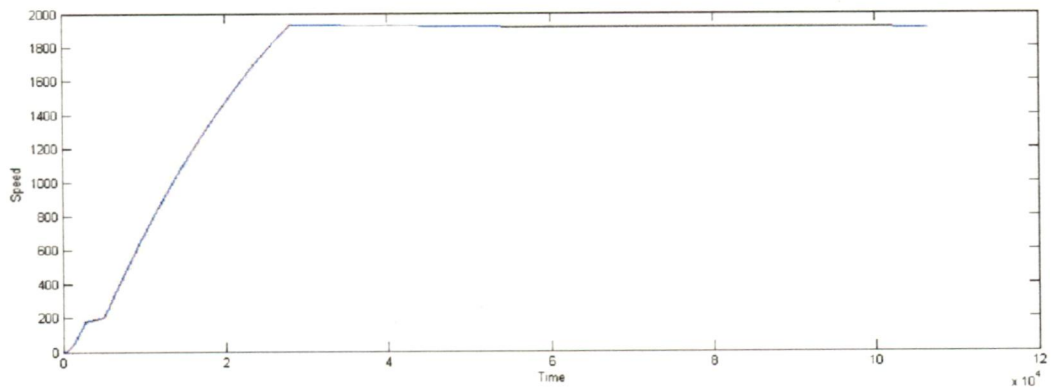


Figure 6.1.4: Speed using Fuzzy Logic Controller

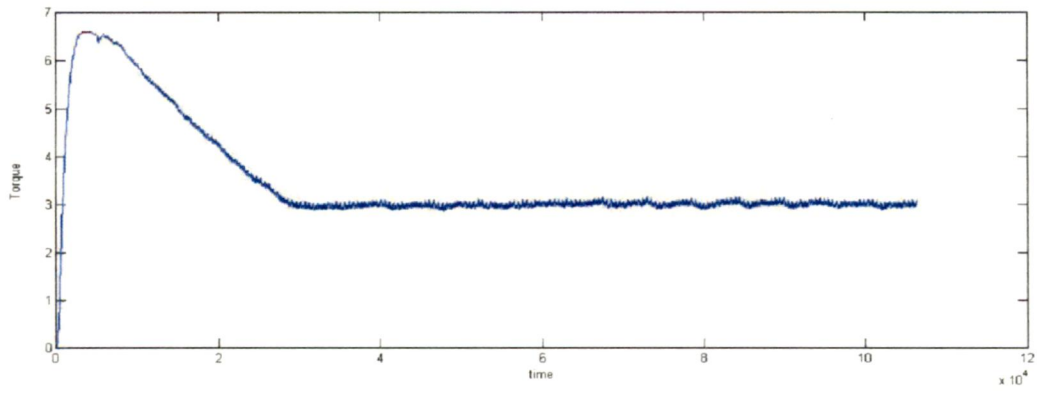


Figure 6.1.5: Torque using Fuzzy Logic Controller

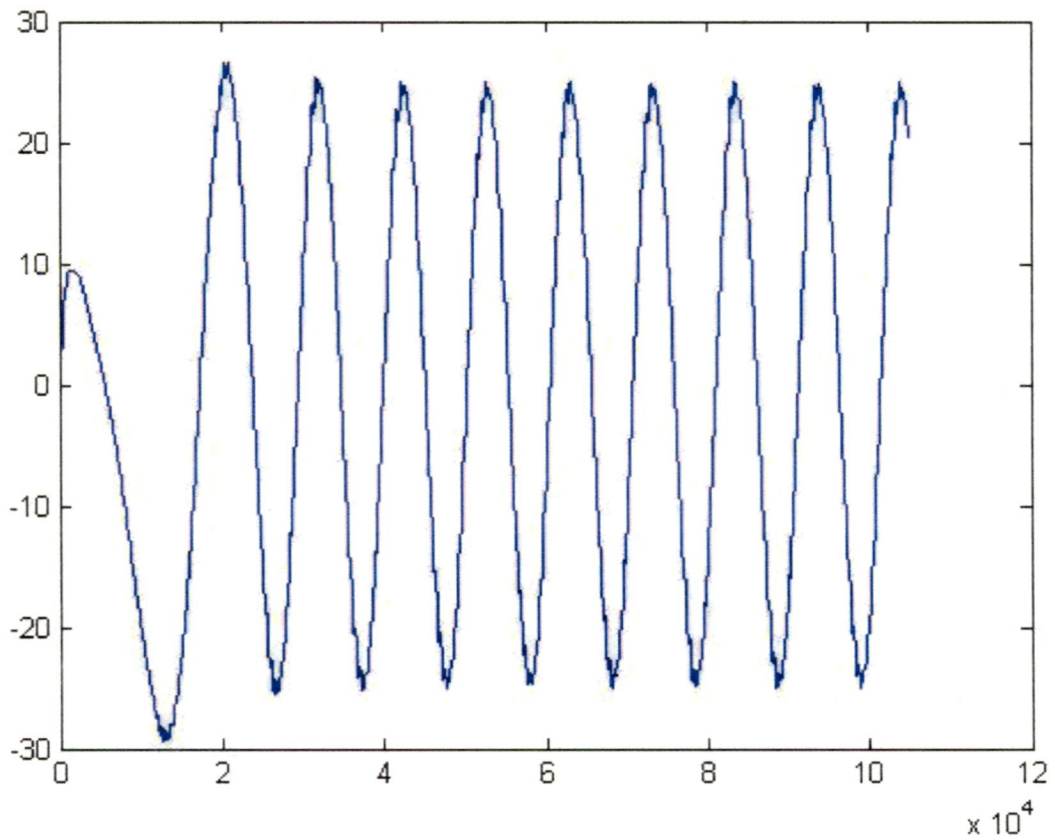


Figure 6.1.6: Stator Current using Fuzzy Logic Controller

The particle swarm based minimum and maximum values for the scaling of fuzzy logic controller are shown in table 6.1.

PSO optimization results for the first run are shown in table 6.2

	<i>ge</i>	<i>gce</i>	<i>gu</i>
<i>Minimum</i>	0.01	0.01	0.1
<i>Maximum</i>	0.4	0.4	30

Table 6.1: Minimum and maximum search limits for each parameter

	<i>ge</i>	<i>gce</i>	<i>gu</i>
<i>FirstRun</i>	0.0788	0.01	8.685

Table 6.2: Particle Swarm Optimization results

6.1.2 Anfis Controller

The performance results using the ANFIS controller resulted in better control as shown in figure 6.1.9 shows the output at different loads of $T_l = 4\text{Nm}$, 6Nm , 7Nm at time instants $t = 0.45\text{ sec}$, 1.6 sec , 2.1 sec

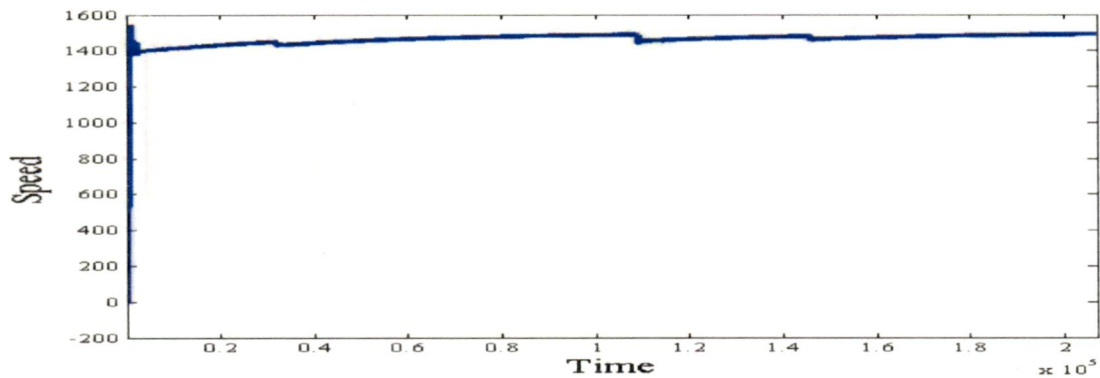


Figure 6.1.7: Speed using ANFIS

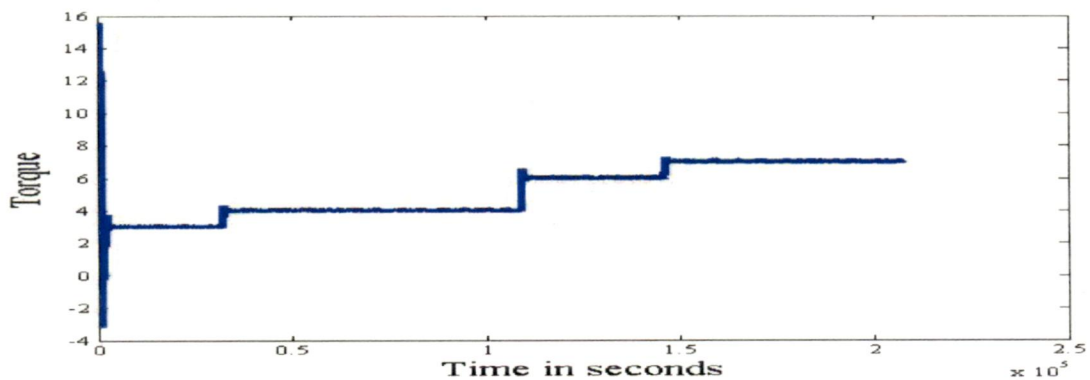


Figure 6.1.8: Torque using ANFIS

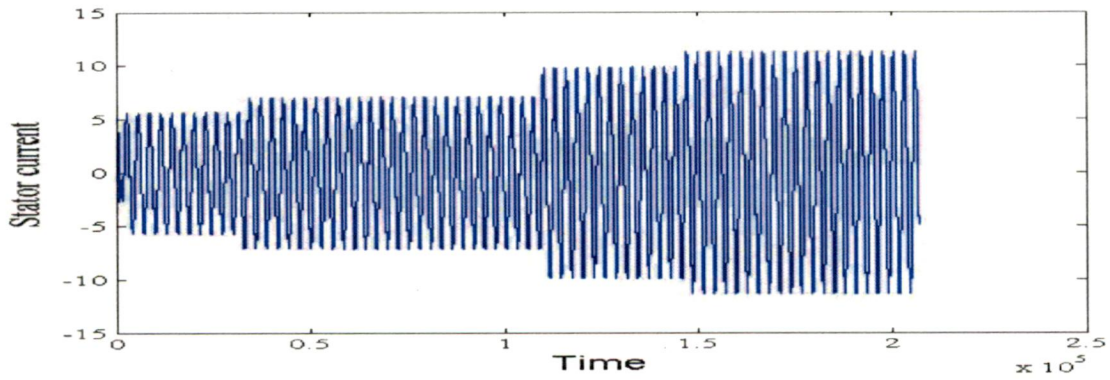


Figure 6.1.9: Stator Current using ANFIS

6.1.3 Comparison Between Neural,Fuzzy and ANFIS Controller

Comparison of the three controllers are shown in the following table 6.3

Property	<i>NC</i>	<i>FLC</i>	ANFIS
Overshoot	less than 0.5	less than 0.1	Less Than 0.2
Settling Time	0.4s	0.06s	0.2s
Starting Maximum Current	60 A	58 A	30 A
Speed range	up to critical speed	up to double speed	Up to extreme speed
Computation	High	Very High	Lower Than Others
High	High	Less than Others	

Table 6.3: Comparison of the three controllers

6.1.4 Comparison between Type-1 Fuzzy and Type-2 Fuzzy Logic

The simulation is realized using the SIMULINK software in MATLAB environment. Figure 6.1.12 shows the performances of the Type-2 FLC. It can be observed that at the instant of applying load torques at 0.1s,0.5s,0.84s,1.15s there exists sudden spikes and dips in the type-1 Fuzzy Controller where as the type-2 Fuzzy Logic Controller shows no oscillations, overshoots or dip in the motor speed. The torque output for a type-1 FLC as shown in fig 6.1.12 large oscillations at various instants of applied load torque. The type-2 FLC shows no oscillations in the output torque as shown in fig 6.1.12

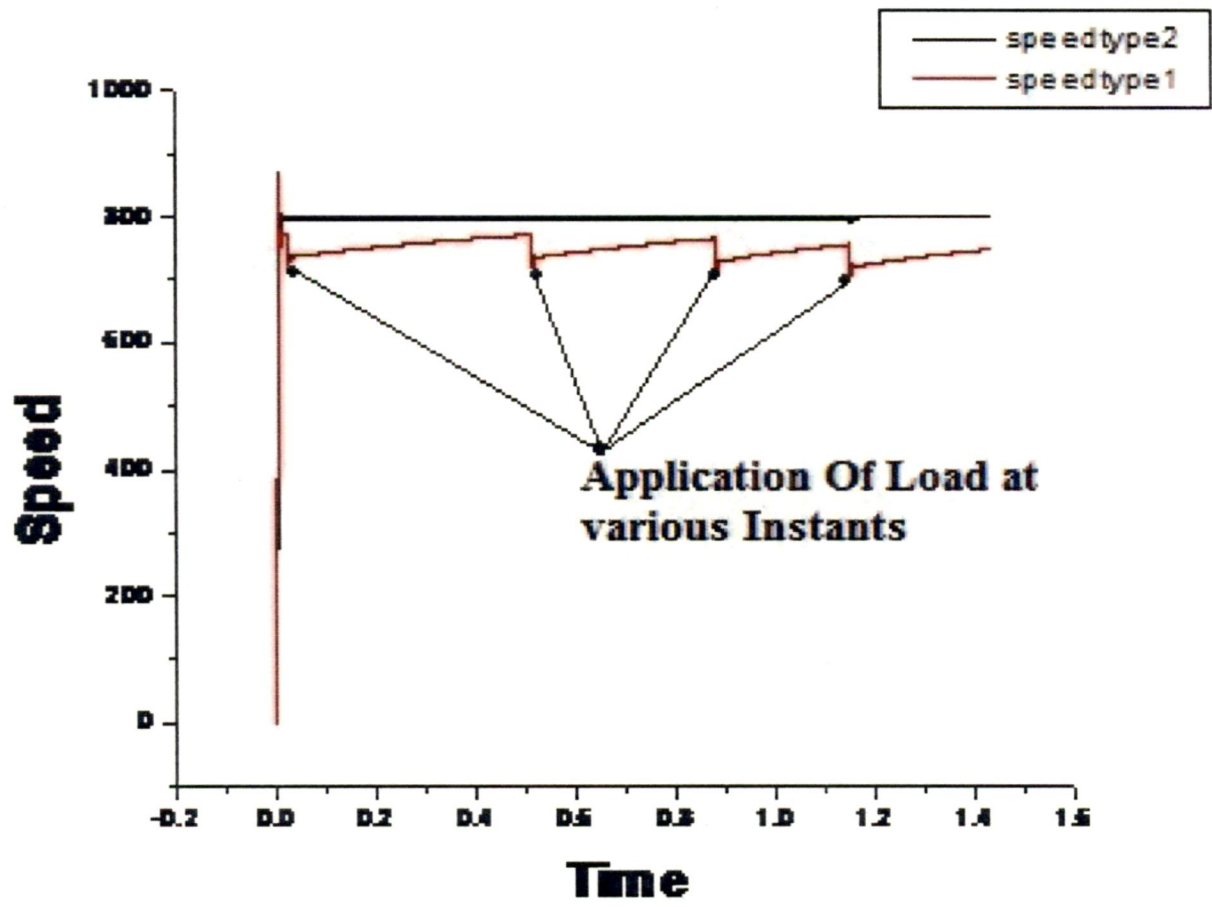


Figure 6.1.10: Speed comparison of type1 and type2 FLC

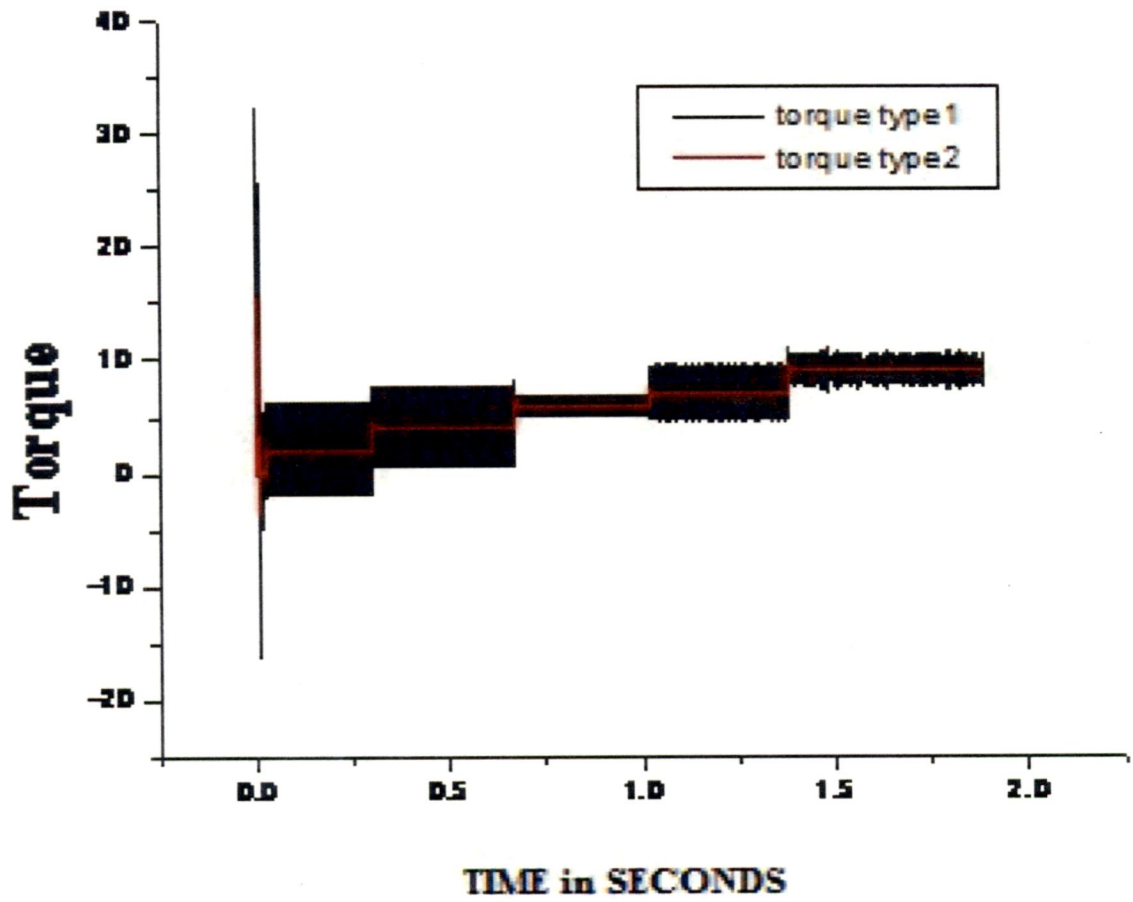


Figure 6.1.11: Torque comparison of type1 and type2 FLC

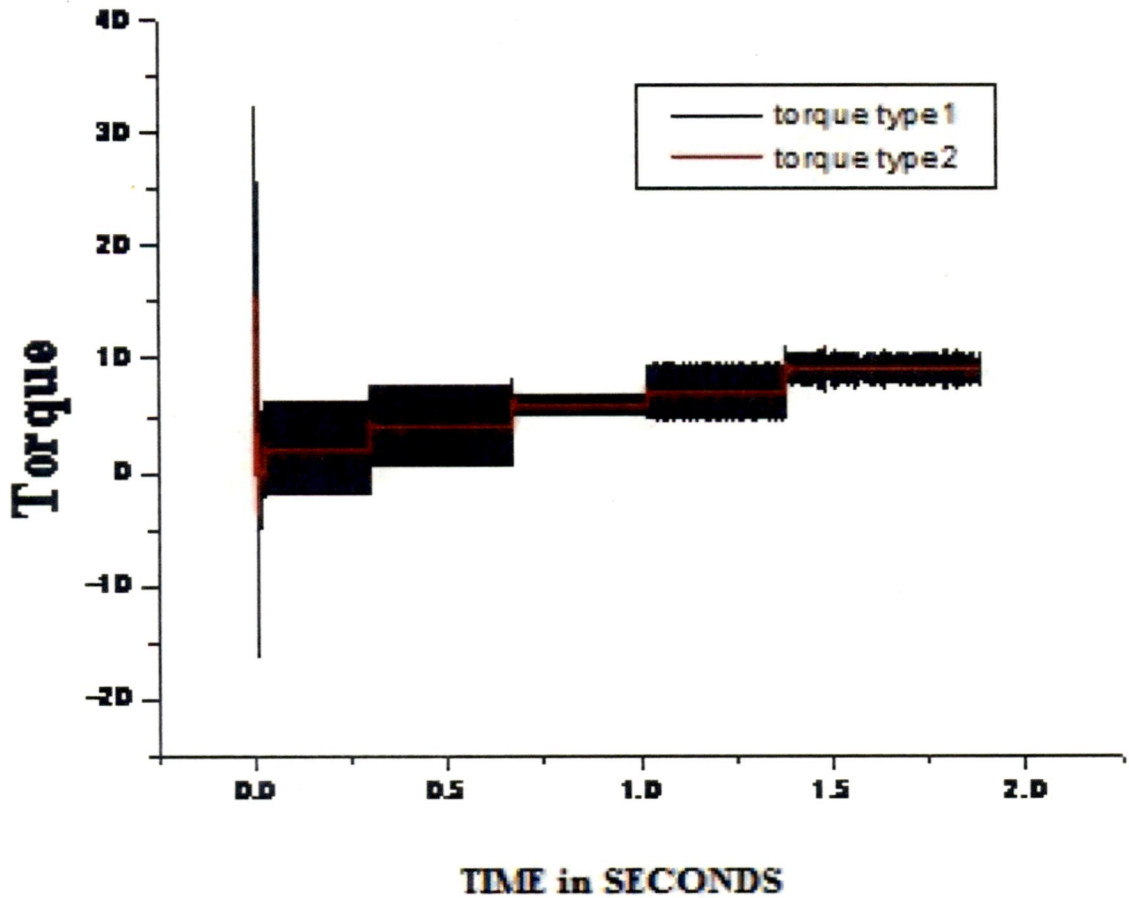


Figure 6.1.12: Direct axis and Quadrature axis currents for type2 FLC

Table 6.4 gives the comparison of results of the simulation of a type-1 FLC and type-2 FLC based vector controlled PMSM along with the conventional based controller. It can be observed from the table that the settling time is least in case of type-2 FLC than in Type-1 FLC. The rise time of Type-2 FLC is less than the conventional PI controller and Type-1 FLC.

	<i>RiseTime</i>	<i>SettlingTime</i>
<i>Type - 1</i>	0.3	0.6
<i>Type - 2</i>	0.1	0.3

Table 6.4: Comparison of type-1 and type-2 fuzzy

Chapter 7

Conclusion

7.1 Conclusion and Scope of Future work

From the review of the previous works the synchronous motor is a good choice for high performance variable speed drives in industry in view of performance and economy. However, the performance depends on the type of controllers used. Using simple controller with the assumption of $i_{ds}=0$, the control of synchronous motor becomes easier with the cost of reluctance torque and high speed operation. The equivalent circuit model of permanent magnet synchronous motor drive is investigated and the operation is analyzed. A vector control method is incorporated by controlling the quadrature axis component

An Adaptive Model Reference Neural Network Controller is proposed for controlling the synchronous motor. It is able to follow the command speed smoothly and quickly maintaining global stability. The disadvantage is that it requires a lot of computation which makes it difficult and cumbersome to design and implement. Moreover a large number of parameters are associated with these types of controllers which make it more expensive. Uncertainty and non-linearity from the motor mechanical load sometimes cause the drive system to become unstable in the absence of proper control.

A Fuzzy logic controller is developed to control the speed and torque component such that the motor can be run efficiently. In order to verify the efficacy of FLC in high performance application, a vector control scheme of the PMSM incorporating the FLC has been simulated. However the FLC showed good transient response but shows steady state ripple. Particle swarm intelligence is used in estimating the fuzzy scaling parameters of the FLC.

For further improvement of the intelligent control performance, an ANFIS based Neuro fuzzy controller is developed with tuning the membership function. The tuning procedure of the membership functions of the neuro fuzzy logic controller is discussed in this chapter. The neuro fuzzy controller is designed in such a way that the computational burden remain low which is

suitable for real-time implementation.

A Type-2 FLC is proposed which has a fuzzy nature incorporating a third membership function eliminates nonlinearities and effects under dynamic conditions and in applications requiring three dimensional control such as robotics. Simulation results verify the feasibility of the proposed controllers for real life industrial applications. Finally simulation results are presented to validate the controllers performance. This thesis develops different types of speed controllers for high performance synchronous motor.

- In the development of an adaptive controller, only the mechanical parameters were estimated. d and q axes inductances are varying with different operating condition. So future work can be done with the estimation of electrical parameters too.
- In neuro fuzzy controller design, the tuning of membership functions was done online. But there was still ripple in speed. These ripples caused by the ripple in command q -axis current. a filter can be designed and used to optimize the speed ripple. But this might make the controller slow.
- In the development of Type-2 Fuzzy logic controller for synchronous motor the controller though shows ripple free speed and ripple free torque content even on application of high loads, the controller is very slow in giving the results.
- Work has been started in developing a type-2 fuzzy logic controller based synchronous motor to apply on six Degrees Of freedom robot system. Since the fuzzy nature of type-2 fuzzy eliminates any uncertainty

References

- [1] Consoli A., Scarcella G., and Testa.A., "Industry application of zero-speed sensorless control techniques for PM synchronous motors," *IEEE Trans. Industry Application*, vol.37,pp.513-521,Mar./Apr.2001.
- [2] Uddin M.N., Radwan T.S., George G.H., and Rahman M.A., "Performance of Current Controllers for VSI-Fed IPMSM Drive," *IEEE Trans. Industry Application*, Vol.36,No.6,Nov./Dec.2000.
- [3] Slemon G.R., "Electrical Machines and Drives", Reading,MA: Addison-Wesley,1992,pp.503-511.
- [4] Cao X.,Fan L., "Self-Tuning PI Controller for Permanent Magnet Synchronous Motor based on Iterative Learning Control", *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on* , vol.1, no., pp.756-760, 20-22 Dec. 2008
Self-Tuning PI Controller for Permanent Magnet Synchronous Motor Based on Iterative Learning Control
- [5] Zadeh L.A., "Outline of a New Approach to the Analysis of Complex System and Decision Processes", *IEEE Trans. on Syst,Man and Cybern.*, vol.SMC-3,1973,pp.28-44.
- [6] "Fuzzy Logic Toolbox User Guide",The Math Works.,1997.
- [7] Cirstea M.N., Dinu A., Khor J.G., McCormick M., "Neural and Fuzzy Logic Control of Drives and Power Systems,"Newnes, Oxford, 2002.
- [8] Bose B.K., "Expert System, Fuzzy logic, and neural network Applications in power Electronics and motion control", *Proceedings of the IEEE*, Vol. 82, NO. 8 August 1994, 1303-1321
- [9] Bhler H., "Rglage par logique floue," Presse Polytechniques et Universitaires Romandes, Lausanne, 1994.

- [10] JSpooner J.T., Maggiore M., Ordonez R., Passino K.M., "Stable adaptative control and estimation for nonlinear system, Neural and fuzzy approximator techniques", Wiley-Interscience, 2002.
- [11] Baghli L., "*Contribution la commande de la machine asynchrone, utilisation de la logique floue, des rseaux de neurone et des algorithmes gntiques.*" Thse de doctorat, S.T.I.M.A-NANCY, 1999.
- [12] Azizur Rahman M., Radwan T.S., Osheiba A.M., and Lashine A.E., "Analysis of current controllers for voltage source inverter," *IEEE Trans. Industry Electronics*, vol.44,no.4, Aug.1997.
- [13] Aissaoui A.G., Abid M., Abid H., And Tahour A., "A Neural Controller for Synchronous Machine," *International Journal of Electrical Systems Science and Engineering* 1:4,2008, pp.222-229.
- [14] Bose B.K., "*Modern Power Electronics and AC Drives.*" Englewood Cliffs, NJ: Prentice-Hall, 2002.
- [15] Hani A.Hagras, "Hierarchical Type-2 Fuzzy logic control architecture for Autonomous Mobile Robots," *Proc.IEEE*, vol.12, pp.524-539, no.4, August 2004.
- [16] Mendel J.M., "Fuzzy Logic Systems for Engineering: a Tutorial," *Proceedings of IEEE*, Vol.83, no.3, March 1995.
- [17] Karnik, N.N., and Mendel, J.M.: "Centroid of a type-2 fuzzy set," *Inform. Sci.*, 2001, 132, pp. 195-220
- [18] Zadeh L.A., "The Concept of A Linguistic Variable and Its Application to Approximate Reasoning - 1 ," *Information Sciences* 8, PP. 199 - 249. 1975
- [19] Wang L.X and Mendel L.M., "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, September 1992, pp.807-814
- [20] Mendel, J.M., and John, R.I.B., and Liu, F.: "Type-2 Fuzzy Systems Made Simple," *IEEE Trans. Fuzzy Syst.*, 2002, 10, (2), pp.117-127.
- [21] Mendel, J.M., and John, R.I.B., and Liu, F.: "Interval Type-2 Fuzzy Logic Systems Made Simple," *IEEE Trans. Fuzzy Syst.*, 2006, 14, (6), pp.808-821.
- [22] Mendel, J.M.: "Uncertain rule-based fuzzy logic systems; Introduction and new directions" (Prentice-Hall, Englewood Cliffs, NJ, 2001)

- [23] Uddin, M.N., and Rahman, M.A.: "Fuzzy logic based speed control of an IPM synchronous motor drive," in Proc. 1999 *IEEE Canadian Conf. Electr. Comput. Eng.*, May 1999, pp. 1259-1264.
- [24] Karnik Nilesh, N., Mendel Jerry, M., "Type-2 Fuzzy Logic Systems," *IEEE Transactions on Fuzzy Systems*, VOL.7, No.6, December 1999, pp.643-658.
- [25] Karnik, N.N and Mendel, J.M., "Type-2 fuzzy logic systems: Type-reduction," Proc. IEEE Conference on Systems, Man and Cybernetics, pp. 2046-2051, Oct. 1998c.
- [26] Karnik, N.N and Mendel, J.M., "Introduction to type-2 fuzzy logic systems," *Proc. 1998 IEEE FUZZ Conf.*, pp. 915-920, May 1998.
- [27] Liang, Q., and Mendel, J.M., "Interval Type-2 Fuzzy Logic Systems: Theory and Design," *IEEE Trans. on Fuzzy Systems*, vol. 8, pp. 535-550, 2000.
- [28] Liang, Q., and Mendel, J.M., "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 30, pp. 329-339, Aug. 2000.
- [29] Kebbati, Y., Girerd, C., Chapuis, Y.A., and Braun, F., "Advances in FPGA/ASIC Digital Integration Solui for Vector Control of Motor Drives," *Proceedings of International Power Electronic Conference (IPEC)*, Vol. 3, pp 1177-1182, 2000.
- [30] Zadeh, L.A. "Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering*," Vol.1, 891-900 (1989)
- [31] Kennedy, K., and Eberhart, L.A., Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, Perth, Australia, 1995.
- [32] Shi Y., Particle Swarm Optimization, Feature article, Electronic Data Systems, Inc., *IEEE Neural Networks Society*, Feb. 2004.
- [33] Kennedy, J., Eberhart, R.C.M and Shi, Y., "Swarm Intelligence", San Francisco: Morgan Kaufmann Publishers, 2001.
- [34] Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., and Nakanishi, Y., "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. Power Systems*, vol. 15, no. 4, pp. 1232-1239, Nov. 2001.
- [35] Xiaohui Hu, Yuhui Shi, and R. Eberhart, "Recent advances in particle swarm," *IEEE Congress on Evolutionary Computation (CEC2004)*, vol. 1, pp. 9097, 19-23 June 2004.

- [36] Widrow,B., and Lehr,M.A., "30 years of adaptive neural networks: perceptron, madaline, and backpropagation", *Proceedings of the IEEE* Vol. 78, NO. 9, September 1990, 1415-1442.
- [37] Low,T.S., Lee,T.H., and H. K. Lim, "A methodology for neural network training for control of drives with nonlinearities", *IEEE Transaction on Industry Electronics* , Vol. 39, NO. 2 April 1993, 243-249
- [38] Bose,B.K.: "*Power electronics and AC drives*". Prentice Hall edition.
- [39] Lightbody,G., and Irwin,G.W., "Direct neural model reference adaptive control", *IEE Proc.-Control Theory Appl.*, Vol. 142, No. 1, January 1995
- [40] Moley kutty George,Kartik Prasad Basu,E.,Alan Tan Wee Chiat,E., "Model reference controlled separately excited DC motor" 6 April 2009 *Springer-Verlag London Limited 2009*
- [41] Abdel Ghani Aissaoui,Hamza Abid, Mohamed Abid , " Fuzzy sliding mode control for a self-controlled synchronous motor drives", *Electronic journal of Technical Acoustics*,2005,16.
- [42] Hagan,M.T., and Menhaj,M., "Training feed-forward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1999, pp. 989-993, 1994.
- [43] Jang, J.-R., "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Transactions*, pp. 665-685, (1993).
- [44] Simon Haykin, "*Neural networks: A comprehensive foundation*," Prentice hall, New Jercey,1994.
- [45] Chow,M.,Sharpe,R.N., and Hung,J.C., "On the application and design of artificial neural networks for motor fault detection - Part I", *IEEE Transaction on Industry Electronics*, Vol. 40, NO. 2, April 1993, 181-188.
- [46] Baghli,L., " Contribution la commande de la machine asynchrone,utilisation de la logique floue, des rseaux de neurone et des algorithmes gntriques. Thse de doctorat," S.T.I.M.A-NANCY, 1999.
- [47] Shing, J. and Jang, J.-R., "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm," vol. AAAI-91, pp. 762-767, (1991).
- [48] Bandura, A., Ross,D., and Ross,S.A., "Vicarious Reinforcement and Imitative Learning," *Journal of Abnorm Psychology*, 1963
- [49] Thorndike, E. L.; "The Law of Effect," Teachers College, Columbia University, 1927.

- [50] Eberhart, R. C. and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization," Lecture Notes in Computer Science; Vol. 1447, *Proceedings of 7th International Evolutionary Programming VII*, pp. 611-616, 1998
- [51] Takagi T., Sugeno M., "Derivation of fuzzy control rules from human operators control actions," *Proceeding of IFAC Symposium Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseilles, France, July 1983, pp. 5560.
- [52] T. Takagi T., Sugeno M., "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Systems Man Cybern.* 15 (1985) pp.116132
- [53] Jang J., "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.* 23 (3) (1993) 665685.
- [54] Gulley N, Jang JSR, "Fuzzy logic toolbox a reference guide," The MATHWORKS Inc.; 1995.
- [55] Ayyub M., "ANFIS based soft-starting and speed control of AC voltage controller fed induction motor," *Power India Conf IEEE*; 2006 pp.223-231

Appendix A

Appendice

A.1 Model reference Parameters

The parameters of model reference controller are

- Size of the Hidden layer=13
- Maximum reference value=0.7
- Minimum reference value=-0.7
- Maximum interval value=2 sec
- minimum interval value=0.1 sec
- Controller training samples=6000
- Controller Training Epochs=10
- Controller Training segments=30

The parameters of plant identification model are

- Size of the Hidden layer=10
- Sampling Interval=0.05 sec
- No of training samples=10000
- Maximum Plant input=15
- Minimum Plant input=-15
- Maximum interval value=2 sec

- minimum interval value=0.1 sec
- Controller Training Epochs=300
- training function = trainlm

A.2 Synchronous Motor Parameters

Three phase Synchronous Motor parameters are Rated output power 3HP, Rated phase voltage 60V, Rated phase current 14 A, Rated field voltage $v_f = 1.5V$, Rated field current $i_f = 30A$, Stator resistance $R = 0.05\Omega$, Direct stator inductance $L_s = 0.325\Omega$, Field resistance $R_f = 0.05\Omega$, Direct stator Inductance $L_{ds} = 8.4mH$ Quadrature stator inductance $L_{qs} = 3.5mH$, Field leakage inductance $L_f = 8.1 mH$, Mutual inductance between inductor and armature $M_{fd} = 7.56mH$, The damping coefficient $B = 0.005 N.m/s$, The moment of inertia $J = 0.05kg.m^2$, Pair number of poles $p = 2$.