ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Comparative analysis of Genetic algorithm tuned Fractional and Integer order PI controllers with Adaptive neurofuzzy controller for speed control of indirect vector controlled Induction motor

A thesis submitted to the School of Electrical and Computer Engineering of Addis Ababa Institute of Technology, School of Graduate Studies, Addis Ababa University in partial fulfillment of the requirement for the Degree of **Masters of Science in Control Engineering.**

By

Girma Kassa

Advisor: Dereje Shiferaw (Ph.D.)

January, 2019

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Comparative analysis of Genetic algorithm tuned Fractional and Integer order PI controllers with Adaptive neurofuzzy controller for speed control of indirect vector controlled Induction motor

A thesis submitted to the School of Electrical and Computer Engineering of Addis Ababa Institute of Technology, School of Graduate Studies, Addis Ababa University in partial fulfillment of the requirement for the Degree of **Masters of Science in Control Engineering.**

By

Girma Kassa

Approval by Board of Examiners

_____	_____
Chairman, Dept. Graduate Committee	Signature
Dr. Dereje Shiferaw	_____
Advisor's Name	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or other universities, all sources of materials used for this thesis work have been fully acknowledged.

Name: Girma Kassa

Signature: _____

Place: Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa, Ethiopia

Date of Submission: January, 2019

This thesis has been submitted for examination with my approval as a university advisor

Dr. Dereje Shiferaw _____

Advisor's Name Signature

Acknowledgement

I wish to express my sincere appreciation to my advisor, Dr. Dereje Shiferaw for his insight and support throughout this thesis. In addition to my advisor, I would like to thank all friends of Electrical and Computer Engineering Department for their assistance and encouragement throughout this work.

Abstract

This thesis presents design and comparative analysis of fractional order PI controller, integer order PI controller and an adaptive neurofuzzy controller trained by input output data from fractional order PI controller for indirect vector controlled induction motor. The parameters of the two PI controllers were genetically optimized using square of error as a fitness function. The proposed neurofuzzy controller trained with input and output data of fractional order PI controller incorporates fuzzy logic algorithm with a multilayer artificial neural network structure using hybrid learning algorithm. This improves the performance of induction motor drive. The fractional order model of induction motor has been also investigated using simulation results and it was inferred that optimized model of induction motor is an integer order model. The performance of adaptive neurofuzzy inference system controller, was compared with fractional and integer order PI controllers using MATLAB simulation results with different operating conditions. It was observed from the simulation results that by using ANFIS, FOPI, and IOPI controllers, for the reference speed of 50 rad/sec, the percentage peak overshoots were 0.496%,13.068% and 15.698% respectively. Thus, ANFIS shows dramatic decrease in overshoot. Also the speed reaches its desired set value at 0.15 second in ANFIS controlled IM drive. These show the effectiveness of the designed neurofuzzy controller and the designed neurofuzzy controller tries to speed up the performance of IM drive. On the other hand, FOPI controller showed better performance than IOPI controller for IM drive, this is because of FOPI controller has one additional parameter for tuning which is integration order.

Keywords: Fractional order controller, Induction Motor, Indirect vector control, Adaptive Neurofuzzy Inference System, Artificial Neural Network, Fuzzy logic, Hybrid learning algorithm.

Table of Contents

List of figures

List of tables

List of symbols and abbreviations

GA	Genetic Algorithm
PI	Proportional Integral
PID	Proportional Integral Derivative
ANN	Artificial Neural Network
FLC	Fuzzy Logic Controller
ANFIS	Adaptive Neurofuzzy Inference System
TSK	Tagaki Sugno Kang
MATLAB	Matrix Laboratory
IM	Induction Motor
IVCIM	Indirect Vector Controlled Induction Motor
FOPID	Fractional Order Proportional Integral Derivative
FOPI	Fractional Order Proportional Integral
IOPI	Integer Order Proportional Integral
λ	Integral Order
μ	Derivative Order
FOMCON	Fractional Order Modeling and Control
IFOC	Indirect Field Oriented Control
PWM	Pulse Width Modulation
Kp	Proportional gain
Ki	integral gain
IGBT	Insulated Gate Bipolar Transistor
aD_t^{α}	Fractional order derivative/integral

Γ	Euler's gamma function
GL	Grunwald-Letnikov definition
RL	Riemann-Liouville definition
LTI	Linear Time Invariant
R	Real number
N	Natural number
$\mathcal{J}$	Laplace transform operator
!	Factorial operator
LTI	Linear Lime Invariant
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
BIBO	Bounded Input Bounded Output
MMF	Magneto Motive Force
EMF	Electro Motive Force
FOC	Field Oriented Control
DQ	Direct and Quadrature axis
AC	Alternating Current
DC	Direct Current
IFOC	Indirect Field Oriented Control
SRRF	Synchronously Rotating Reference Frame
$\lambda_{ds}, \lambda_{qs}, \lambda_{0s}$	d,q and zero axis stator flux linkage
λ'_{dr} , $\lambda'_{qr}, \lambda'_{0r}$	d,q and zero axis rotor flux linkage referred to stator
R_s	Stator resistance
R'_r	Rotor resistance referred to stator
V_{ds},V_{qs}, V_{0s}	d,q and zero axis stator voltage
V'_{dr}, V'_{qr},V'_{0r}	d,q and zero axis rotor voltage referred to stator
i_{ds} , i_{qs},i_{0s}	d,q and zero axis stator current
$i'_{dr}, i'_{qr}, i'_{or}$	d,q and zero axis rotor current referred to stator

L_m	Mutual inductance
L_{ls}	Stator self-inductance
L'_{lr}	Rotor self-inductance referred to stator
ω_e	Synchronous speed
ω_r	Rotor speed
T_e	Electromagnetic torque
T_L	Load torque
J	Moment of inertia
F	Friction coefficient
P	Number of poles
SVPWM	Space Vector Pulse Width Modulation
ω_{sl}	Slip frequency

CHAPTER ONE

1. Introduction

1.1 Background

Vector controlled induction motor drive is a very accepted method for high performance system response [1,2]. This method employs the conventional/integer order Proportional Integral, Proportional Integral Derivative controller or their adaptive versions, for variable speed drive applications. The difficulties of obtaining the exact parameters of the induction motor leads to cumbersome design approach. Also the conventional fixed gain PI and PID controllers are very sensitive to disturbances, parameter variations and system nonlinearity. Artificial Neural Network and Fuzzy Logic Control demands special attention for speed control of high performance induction motor drives.

Fuzzy Logic Controller yields superior and faster control [3,4], without the need of accurate mathematical model of the system and works well for complex, nonlinear, multidimensional system with parameter variations or with less precise signals. The main design problem lies in the determination of consistent and complete rule set and shape of the membership functions. A lot of trial and error has to be carried out to obtain the desired response which is time consuming. On the other hand, ANN alone is insufficient if the training data are not enough to take care of all the operating modes.

ANFIS is used as an intelligent tool to design FLC. It helps to generate and optimize membership functions as well as the rule base from the simple data provided. ANFIS combine the learning power of neural network with knowledge representation of fuzzy logic. This thesis presents speed control scheme of vector controlled IM based on Neuro fuzzy controller. The proposed NFC is adapted by hybrid learning algorithm in order to minimize the square of the error between desired and actual output. Layers of ANN structure is utilized to train the parameters of the FLC, which eliminates unwanted trial and error as was in the case for a conventional fuzzy logic control [5].

In recent years fractional order calculus has gained a lot of attention, especially in the field of system theory and control systems design due to more accurate modeling and control enhancement possibilities [6,7]. It is claim that fractional order differential equations can describe real world systems more adequately. In control practice, it is useful to consider the fractional order controller design for an integer order plant [8].

Genetic Algorithm can optimize parameters of both fractional and integer PI controllers.GA initially generates a random population, which is implemented with small population size in order to allow the controller to be optimized and converge at a faster rate [9].

1.2 Literature Review

Some of materials that I have reviewed to do this thesis are the following.

1. In [10], optimization methods in fractional order control of electric drives has also been discussed. This paper presents a comparative study on the optimization methods in fractional PID controller design for induction motor. Choosing the best optimization algorithm for tuning fractional order controllers is crucial in order obtain best system response. In this study genetic algorithm, local search, nonlinear sequential quadratic programming, and particle swarm optimization algorithms are used to tune a fractional PI controller for induction motors.
2. In [11], adaptive neurofuzzy speed controller for vector controlled induction motor drive has been discussed. But this paper used data from conventional (integer order) PI controller.
3. In [12] indirect vector control of induction motor using fuzzy logic controller presents and shows fuzzy logic controller can replace the PI controller.
4. In [13] fractional order modeling and control(FOMCON) toolbox for MATLAB is explained by Aleksei Tepljakov. This paper presents all the major modules comprising the toolbox and discuss the corresponding mathematical concepts. Fractional order system analysis, identification and fractional controller design, tuning and optimization in the context of the toolbox are presented and discussed.
5. In [14] Fractional order modeling and control of dynamic systems has been discussed. Modeling real dynamic systems using fractional order calculus and control has been presented.
6. In [15] applications of fractional calculus have been discussed. Different definitions of fractional derivatives and fractional integrals has been presented. By means of them explicit formula and graphs of some special functions are derived. Also they review some applications of fractional calculus.

1.3 Problem statement

The most widely used controller for controlling speed of vector controlled induction motor is the conventional proportional integral controller. But, the performance of PI controller diminishes under the uncertainties of motor which include unknown load on motor and its resistance variations. It is also sensitive to system nonlinearity.

Fractional order differential equations can describe real world systems more adequately. This makes fractional order modeled systems to accurate modeling and control enhancement possibilities than conventional (integer) order modeled systems. As a result, from fractional order model accurate data is obtained compared to conventional ordered system. The training data for ANFIS becomes accurate; therefore, the performance of ANFIS increases.

Many literatures have done on ANN controller, FLC including ANFIS for controlling the speed of induction motor. But, nobody uses fractional order PI controller to generate training data for ANFIS controller. This thesis proposes it is possible to enhance the control action of ANFIS by using input and output data of fractional order PI controller as a training data.

1.4 Objective of the thesis

1.4.1 General Objective

The main objective of this thesis is designing and giving comparative analysis of genetic algorithm tuned fractional and integer order PI controllers with adaptive neurofuzzy controller for speed control of indirect vector controlled Induction motor.

1.4.2 Specific Objective

- ✓ To investigate fractional order model of induction motor.
- ✓ To use Genetic algorithm tuning method to determine fractional and integer order PI parameters.
- ✓ To use ANFIS controller for indirect vector controlled induction motor
- ✓ To compare and evaluate the performance of genetically optimized fractional order PI and integer order PI controllers with ANFIS controller trained by fractional PI controller data.

1.5 Scope of the thesis

The scope of this thesis is comparing the performance of IOPI, FOPI, and ANFIS controller trained with data from fractional order PI controller. The thesis is implemented using only MATLAB/SIMULINK simulation with no practical implementation.

1.6 Methodology

The methodology will be followed to solve the problem are as follows. The study begins with gathering and studying literatures that related to this thesis.

- ✓ Modeling of IM using reference frame
- ✓ Investigate fractional order mathematical model of induction motor.
- ✓ Design fractional and integer PI controllers for indirect vector control of induction motor.
- ✓ Replace the function of fractional and integer order PI controllers by ANFIS controller.
- ✓ Performance evaluation and analysis.

After simulation using fractional and integer order PI controllers the required data for ANFIS training are gained. For ANFIS training hybrid method of training is used. Finally, brief analysis depending on the MATLAB/SIMULINK simulation results will be given in terms of parameter and load variation. The overall schematic diagram of speed control of indirect vector controlled induction motor using ANFIS is shown below.

1.7 Organization of the thesis

The thesis has organized into five chapters. Chapter one presents introduction, statement of the problem, objectives of the study, methodology and literature review leading towards the completion of the thesis. The second chapter discusses about integer and fractional order modeling of induction motor, fractional calculus, reference frame transformation, reference frame model of induction motor. Chapter three presents about control of induction motor; concept of vector control, FLC, ANN, ANFIS, and GA. Chapter four presents about simulation results obtained using MATLAB/SIMULINK and discussions of these results. Finally, Chapter five is talking about conclusion and recommendations for future works.

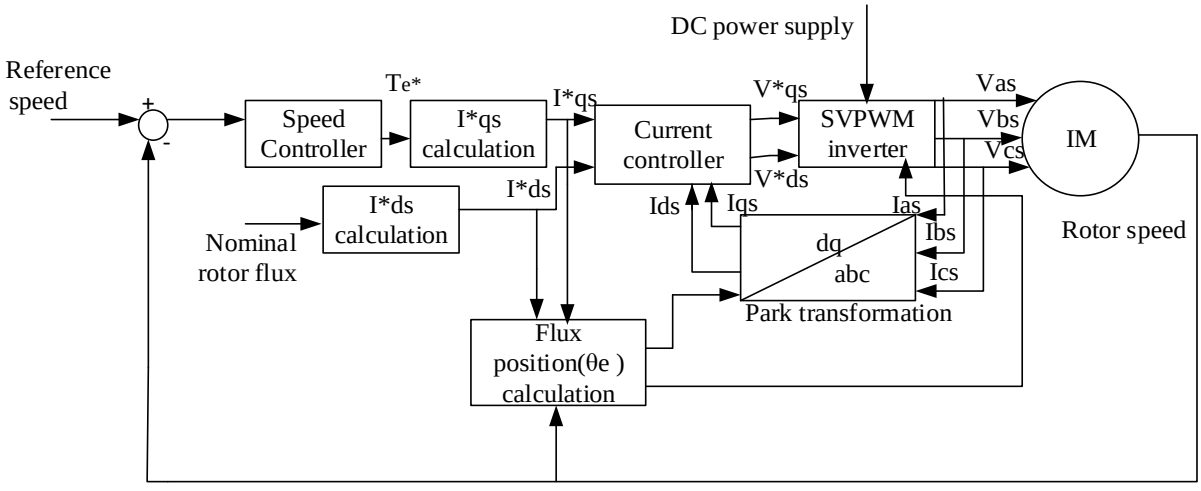


Figure 1.1 Complete diagram of indirect vector control

CHAPTER TWO

2. Modeling of three phase Induction Motor

In this chapter, fractional order calculus and systems, MATLAB toolboxes for analyzing fractional order systems, reference frame transformations i.e. Clarke and Park transformations will be introduced. Finally, induction motor model based on an induction machine equivalent circuit in a synchronously rotating dq reference frame will be discussed; also fractional order model of induction motor from integer order model based on MATLAB simulation results will be investigated.

2.1 Introduction to Fractional order modeling

2.1.1 Historical Background of fractional calculus

Fractional calculus is a topic being more than 300 years old. The idea of fractional calculus has been known since the regular calculus, with the first reference probably being associated with Leibniz and L'Hospital in 1695 where half order derivative was mentioned [19].

Fractional integrals and derivatives appear in the theory of control of dynamical systems, when the controlled system or/and the controller is described by a fractional differential equation. The mathematical modeling and simulation of systems and processes, based on the description of their properties in terms of fractional derivatives, naturally leads to differential equations of fractional order and to the necessity to solve such equations [20].

Fractional order calculus was not particularly popular until recent years when benefits stemming from using its concepts became evident in various scientific fields, including system modeling and automatic control. It is also apparent that this rise of interest is related to accessibility of more efficient and powerful computational tools provided by the evolution of technology and introduction of computer algebra systems (CAS), such as MATLAB and Mathematica [21].

Recent findings support the notion that fractional order calculus should be employed where more accurate modeling and robust control are concerned. Specifically, fractional order calculus found its way into complex mathematical and physical problems [21].

2.1.2 Definition and Laplace transform of fractional order calculus

Fractional systems, or non-integer order systems, can be considered as a generalization of integer order systems. Fractional calculus is a generalization of integration and differentiation to non-integer order fundamental operator aD_t^{α} where a and t are the limits of the operation and $\alpha \in R$ [20]. The continuous integro-differential operator is defined as [19].

$$aD_t^{\alpha} = \begin{cases} \frac{d^{\alpha}}{dt^{\alpha}}, & \alpha > 0 \\ 1, & \alpha = 0 \\ \int_a^t (d\tau)^{-\alpha}, & \alpha < 0 \end{cases} \quad \ldots\ldots (2.1)$$

The Laplace transform is an essential tool in dynamic system and control engineering. There exist multiple definitions of the fractional operator. In the section below main definitions of fractional order derivatives are presented [16, 17, 19, 20, 21, 22].

Definition 2.1. Riemann-Liouville definition of fractional order derivative

$$aD_t^{\alpha} f(t) = \frac{1}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_a^t (t-\tau)^{m-\alpha-1} f(\tau)d\tau \ldots\ldots (2.2)$$

Where $m - 1 < \alpha \leq m \in N$ and $\alpha \in R$ is a fractional order of differentiation of function f(t). The term Γ(.) represents gamma function.

The Laplace transform of RL fractional order differentiation for causal system is given as follows

$$\mathscr{L}[0D_t^{\alpha} f(t)] = s^{\alpha} F(s) - \sum_{k=0}^{n-1} s^k 0D_t^{\alpha-k-1} f(0) \ldots\ldots (2.3)$$

$f(0)$ are the initial conditions, and $n - 1 < \alpha \leq n \in N$.

Definition 2.2. Caputo's definition of fractional order derivative

$$aD_t^{\alpha} f(t) = \frac{1}{\Gamma(m-\alpha)} \int_a^t f^{(m)}(\tau)(t-\tau)^{m-\alpha-1} d\tau \ldots\ldots (2.4)$$

Where $m - 1 < \alpha \leq m \in N$ and $\alpha \in R$ is a fractional order of the differentiation of function f(t).

The Laplace transform of Caputo fractional operator for causal system is given as

$$\mathscr{L}[0D_t^{\alpha} f(t)] = s^{\alpha} F(s) - \sum_{k=0}^{n-1} s^{\alpha-k-1} f^{(k)}(0) \ldots\ldots (2.5)$$

Where $n - 1 < \alpha \leq n \in N$.

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt \ldots\ldots (2.6)$$

$$\Gamma(x) = (x-1)! \ldots\ldots (2.7)$$

Definition 2.3. Grünwald-Letnikov definition fractional order derivative

$$aD_t^{\alpha} f(t) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{[\frac{t-a}{h}]} (-1)^j \binom{\alpha}{j} f(t-jh) \ldots\ldots (2.8)$$

Where [·] means the integer part, h is the step size

Laplace transform of the Grünwald-Letnikov fractional operator for causal system is given by

$$\mathcal{J}[oD_t^{\alpha} f(t)] = s^{\alpha} F(s) \ldots\ldots (2.9)$$

$$\binom{\alpha}{j} = \frac{\alpha!}{j!(\alpha-j)!} = \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \ldots\ldots (2.10)$$

Rieman-Liouville and Caputo definitions are very close. The difference is the order of initial conditions. In Caputo definition these conditions are of integer order which makes them easier to interpret. This is not the case of RL definition where the initial conditions are of fractional order.

2.1.3 Properties of fractional order calculus

Fractional order differentiation has the following properties [16, 19, 21]

1. If f(t) is an analytic function, then the fractional order differentiation oD_t^{α}f(t) is also analytic with respect to t.If α = n where n is integer, then the operator oD_t^{α}f(t) can be understood as the usual operator $\frac{d^n}{dt^n}$.
2. Operator of order α = 0 is the identity operator: $0D_t^0$f(t) = f(t).
3. Fractional order differentiation is linear; if a, b are constants, then

   $0D_t^{\alpha}$ [af (t) +bg(t)]=a$0D_t^{\alpha} f(t)$ +b$0D_t^{\alpha} g(t)$……..…….……..…..(2.11)
4. For the fractional order operators with α > 0, β > 0, and under reasonable constraints on the function f(t) it holds the additive law of exponents.

   oD_t^{α} [$oD_t^{\beta} f(t)$] = oD_t^{β} [$oD_t^{\alpha} f(t)$] = $oD_t^{\alpha+\beta} f(t)$…..…… (2.12)
5. The fractional order derivative commutes with integer order derivative.

   $\frac{d^n}{dt^n}(aD_t^{\alpha} f(t))$ = $aD_t^{\alpha}(\frac{d^n f(t)}{dt^n})$= $aD_t^{\alpha+n} f(t)$……...……..(2.13)

   Under the condition t=a ,we have $f^{(k)}(a)$=0, (k=0,1,2,3……..…n-1).

For numerical calculation of fractional order derivatives, we can use the relation derived from the GL definition. This approach is based on the fact that for a wide class of functions the three definitions GL, RL, and Caputo's are equivalent if f(a)=0. Under the homogeneous initial conditions, the Riemann Liouville and the Caputo derivative are equivalent.

2.1.4 System modeling using fractional calculus

SISO and MIMO systems can be modeled by factional order differential equations. A general LTI fractional order system can be described by a fractional differential equation of the form [19]

$a_n D_t^{\alpha_n} y(t) + a_{n-1} D_t^{\alpha_{n-1}} y(t) + \ldots\ldots\ldots\ldots\ldots + a_0 D_t^{\alpha_0} y(t) =$

$b_m D_t^{\beta_m} u(t) + b_{m-1} D_t^{\beta_{m-1}} u(t) + \ldots\ldots\ldots\ldots\ldots + b_0 D_t^{\beta_0} u(t)$ …...……..……..... (2.14)

The following figure shows classification of LTI systems

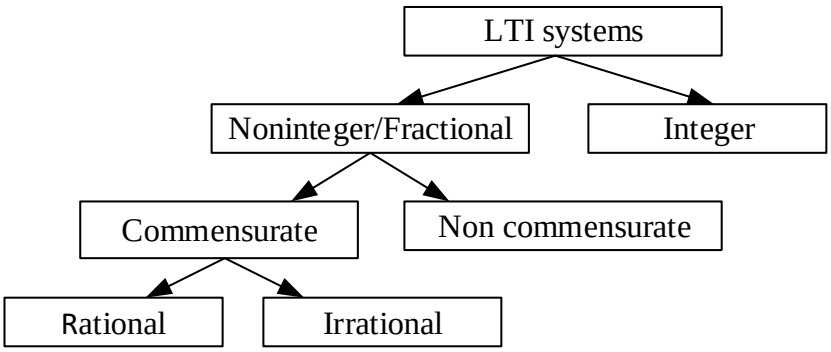


Figure 2.1 Classification of LTI systems

Where the derivative term denotes the Grunwald-Letnikov, the Riemann-Liouville or the Caputo's fractional derivative. The corresponding transfer function of incommensurate real orders has the following form [19].

$$G(S) = \frac{b_m S^{\beta m} + \cdots + b_1 S^{\beta_1} + b_0 S^{\beta_0}}{a_n S^{\alpha_n} + \cdots + a_1 S^{\alpha_1} + a_0 S^{\alpha_0}} = \frac{Q(S^{\beta_k})}{P(S^{\alpha_k})} \quad \ldots\ldots(2.15)$$

In frequency domain 2.15 has the form

$$G(j\omega) = \frac{b_m (j\omega)^{\beta m} + \cdots + b_1 (j\omega)^{\beta_1} + b_0 (j\omega)^{\beta_0}}{a_n (j\omega)^{\alpha_n} + \cdots + a_1 (j\omega)^{\alpha_1} + a_0 (j\omega)^{\alpha_0}} = \frac{Q((j\omega)^{\beta_k})}{P((j\omega)^{\alpha_k})} \quad \ldots\ldots(2.16)$$

Where a_k (k=0,1,2,……..n), b_k (k=0,1,…….m) are constants., and α_k (k=0,1,…….n), β_k (k=0,1……m) are arbitrary real or rational numbers and without loss of generality they can be arranged as $\alpha_n > \alpha_{n-1} > \ldots\ldots > \alpha_0$, and $\beta_m > \beta_{m-1} > \ldots\ldots > \beta_0$ .The incommensurate order system (2.15) can also be expressed in commensurate form by the multivalued transfer function (19).

$$H(S) = \frac{b_m S^{m/v} + \cdots + b_1 S^{1/v} + b_0}{a_n S^{n/v} + \cdots + a_1 S^{1/v} + a_0}, v > 1 \quad \ldots\ldots(2.17)$$

Note that every fractional order system can be expressed in the form (2.17) and the domain of H(S) definition is a Riemann surface with v Riemann sheets. The system is said to be of commensurate order if all the orders of derivation are integer multiples of a base order γ such that α_k, β_k = kγ, γ∈$R^+$and is of non-commensurate order if no common factor exists. The LTI fractional order systems can also be represented by the following state space model (19).

$$\left.\begin{aligned} {}_0D_t^q\, x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}\right\} \quad \ldots\ldots(2.18)$$

Where x∈R^n, u∈$R^r$and y∈R^p are the state, input and output vectors of the system and A∈$R^{n\times n}$, B∈ $R^{n\times r}$, C∈ $R^{p\times n}$, D∈ $R^{p\times r}$, and q= $[q_1, q_2, \ldots, q_n]^T$ are the fractional orders. If

$q_1=q_2=...=q_n\equiv\alpha$, system (2.18) is called a commensurate order system, otherwise it is an incommensurate order system.

The state-space model allows representation of MIMO fractional order systems.

$G(S) = C(S^{\alpha}I - A)^{-1}B + D$……..…..…..……(2.19)

Observability and controllability concept is similar to integer order systems [19].

i. The system (2.18) is controllable on $\lceil t_0 , t_{final} \rceil$ if the controllability matrix has rank n

$C_a = [B \;\; AB \;\; A^2B \;\; \ldots\ldots A^{n-1}B]$……..……...…..……(2.20)

ii. The system (2.18) is observable on $\lceil t_0 , t_{final} \rceil$ if the observability matrix has rank n

$O_a = [C, CA, CA^2, \ldots, CA^{n-1}]^T$ ……....……..…...…… (2.21)

Non-commensurate fractional order non-linear system has the form [20]

$oD_t^{qi}\, x_i(t) = f_i(x_1(t), x_2(t), \ldots\ldots\ldots, x_n(t), t)$……..……..……(2.22)

$x_i(0) = c_i \,,\; i = 1,2,$……., n

Where c_i are initial conditions, or in its vector representation.

$D^q X = f(X)$ ……..……..……..…...…… (2.23)

Where q= $[q1, q2, \ldots, qn]^T$ for 0<q_i<2, (i=1, 2,…,n) and X∈R^n.

The equilibrium points of the system (2.23) are calculated by solving the following equation

$f(X)$=0…….……..……..……...…… (2.24)

2.1.5 Stability of fractional LTI Systems

Stability as an extremely important property of dynamical systems can be investigated in various domains. The usual concept of BIBO or external stability in time domain can be defined via the following general stability conditions [19]. A causal LTI system with impulse response h(t) will be BIBO stable if the necessary and sufficient condition is satisfied.

$\int_0^{\infty} \|h(\tau)\|\, d\tau < \infty$……..……(2.25)

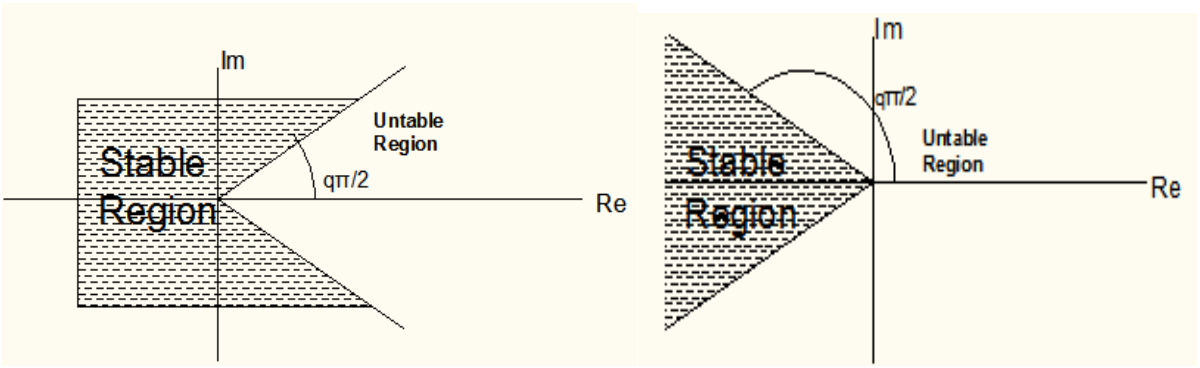
Where the output of the system is defined by convolution

y(t)=h(t)*u(t)=$\int_0^{\infty} h(\tau)u(t-\tau)d\tau$……..……..(2.26)

Another very important domain is frequency domain. In the case of frequency method for evaluating the stability we transform the S plane into the complex plane $G_o(j\omega)$ and the transformation is realized according to the transfer function of the open loop system $G_o(j\omega)$. During the transformation, all roots of the characteristic polynomial are mapped from S plane into the critical point (-1, j0) in the plane $G_o(j\omega)$.The mapping of the S plane into $G_o(j\omega)$ plane is conformal, that is, the direction and location of points in the S plane are preserved in the $G_o(j\omega)$ plane. However, we cannot directly use algebraic tools, for example, Routh-

Hurwitz criteria for the fractional order system because we do not have a characteristic polynomial [19].

In the fractional case, the stability is different from that in the integer one. An interesting point is that a stable fractional system may have roots in right half of the complex w plane as shown in figure 2.2. Since the principal sheet of the Riemann surface is defined $-\pi < \arg(s) < \pi$, by using the mapping $w=s^q$, the corresponding w domain is defined by $-q\pi < \arg(w) < q\pi$, and the w plane region corresponding to the right half plane of this sheet is defined by $-q\pi/2 < \arg(w) < q\pi/2$ [20].



(a) $0<q<1$ (b) $1<q<2$

Figure 2.2 Stability regions of fractional order systems

Matignon's stability theorem states that the fractional transfer function G(s) = Z(s)/P(s) is stable if and only if the following condition is satisfied in w-plane

$|\arg(w)| > q\,\pi/2\ ,\ \forall w \in C,\ P(w) = 0$……..……..……(2.27a)

Where $0 < q < 2$ and $w=s^q$. When w= 0 is a single root of P(s), the system cannot be stable. For q = 1, this is the classical theorem of pole location in the complex plane, no pole is in the closed right plane of the first Riemann sheet.

The fractional order linear time invariant system state space model (2.18) is stable if

$|\arg(\mathrm{eig}(A))| > q\,\pi/2$…….……..…...……(2.27b)

Where $0 < q < 1$ and eig(A) represents the eigenvalues of matrix A

2.1.6 MATLAB toolboxes for analyzing fractional order systems

In recent years, as fractional calculus becomes more and more broadly used across different academic disciplines, there are increasing demands for the numerical tools for the computation of fractional integration/differentiation, or the simulation of fractional order systems. Time to time, being asked about which tool is suitable for a specific application.

1. @fotf

@fotf (fractional order transfer function) is a control toolbox for fractional order systems developed by Xue. Most of the functions inside are extended from the matlab built in functions. In [23], the code and usage of the @fotf toolbox are described in very detail. It uses the overload programming technique to enable the related methods of the matlab built in functions to deal with fractional order models. The transfer function objects generated from it can be interactive with those generated from the matlab transfer function class. Yet, the overloading of associated functions such as impulse (), step (), etc., lost the plotting functionality. As a work around, users can simply define a time vector as the second input to these functions. fotf toolbox supports time delay in the transfer function. It does not directly support transfer function matrix, hence, MIMO systems cannot be simulated directly. However, since it provides Simulink block encapsulation of the involved function fotf(), multiple input/output relationship can be established by manually adding loop interactions in Simulink block diagrams. Drawback of @fotf is that the sampling time has relatively big impact on the accuracy [23].

2. Ninteger

Ninteger, non-integer control toolbox for matlab, is a toolbox intended to help with developing fractional order controllers and assessing their performance**.** It uses integer order transfer functions to approximate the fractional order integrator/differentiator. It also provides Simulink block encapsulation of the involved functions, such as 'nid' and 'nipid' blocks. Moreover, it offers a user friendly GUI for fractional order PID controller design. There is a problem with ninteger toolbox in Matlab version 2013a or later. Without additional editing, it has conflicts with some built-in functions due to the overload editing of the Matlab built-in function "isinteger()"[22].

3. CRONE

The CRONE Toolbox, developed since the nineties by the CRONE team, is a Matlab and Simulink toolbox dedicated to applications of non integer derivatives in engineering and science [24]. It evolved from the original script version to the current object-oriented version. A good feature of the CRONE toolbox is that some of the methods are implemented for MIMO fractional transfer functions. Several other toolboxes are inspired by CRONE, e.g ninteger and FOMCON. A drawback of the CRONE toolbox is that time delay cannot be incorporated into the generated fractional order transfer function. CRONE is a toolbox much more powerful than merely simulating fractional order systems. In spite of this basic functionality, it is also capable of fractional order system identification and robust control analysis and design.

4. FOMCON

The FOMCON (Fractional Order Modeling and Control) toolbox is developed by Aleksei Tepljakov [16]. Its kernel utilizes the algorithms in fotf, ninteger and Crone.It encapsulates some of the major functionalities of those three toolboxes, and builds a GUI shell on top, aiming at extending classical control schemes for FO controller designs. The relation of FOMCON with the three toolboxes is shown in figure 2.3. Some notable changes to the original fotf are

- ✓ newfotf() uses the string parser to enable users to input transfer function as a string
- ✓ tf2ss() is overloaded and foss() is added, which makes the conversion between fractional order transfer function object and fractional order state space object. The CRONE toolbox is also able to do the task, yet the script is encrypted in Matlab P code format.

In this thesis FOMCON Matlab toolbox is used because it includes the major functionalities of fotf, ninteger and Crone toolboxes.

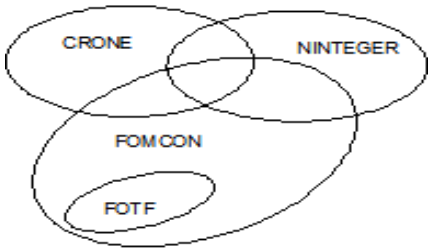


Figure 2.3 FOMCON's relation to other fractional order matlab toolboxes [16].

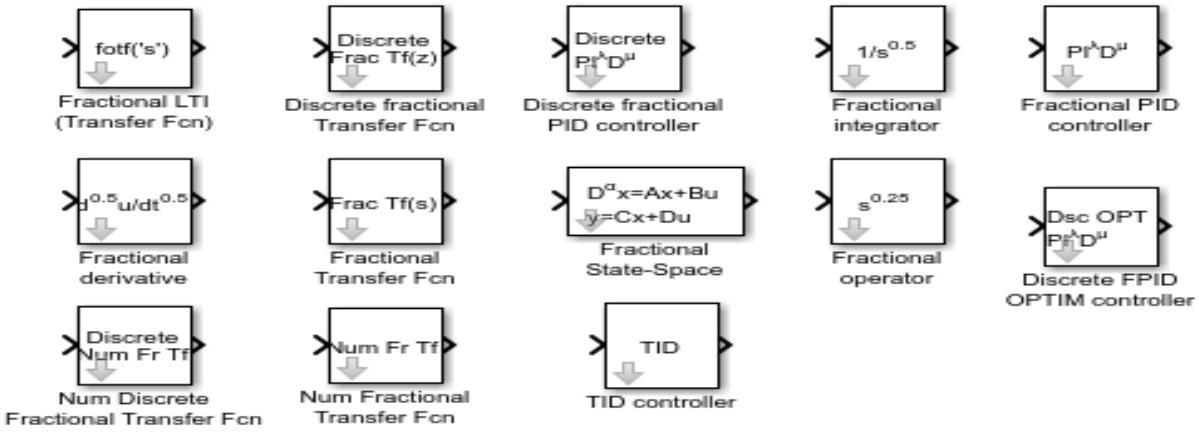


Figure 2.4 The Simulink block set provided in FOMCON toolbox

2.2 Reference frame transformation

2.2.1 Introduction

The inductances of all AC machines vary according to the rotor position. Because of that, the voltage equations of an AC machine are expressed as time varying differential equations as long as the rotor of the machine rotates. The transformation of physical variables of an AC machine using reference frame theory could make the analysis be easy by transforming the time varying differential equations to the time invariant differential equations. The electrical variables such as voltage, current, and flux in a, b, and c phases of a three phase system can be transformed to the variables in d, q, and n or 0 (direct, quadrature, and neutral) orthogonal axes, where the magnetic couplings between axes are zero. Usually, the d axis, which means the direct axis, is the axis where the main flux directs. And the q axis, which means the quadrature axis, lies 90°ahead of the d axis spatially with regard to the positive rotational direction of a rotating MMF. Also, the n or 0 axis, which means the neutral axis and sometimes called as zero sequence axis, is orthogonal to the dq axes in three dimensional space, and the n axis is perpendicular to the plane where the rotating MMF lies; hence the current or voltage at the n axis does not contribute to the rotating MMF and to the torque either, but only to losses. This kind of transformation from the a, b, and c phases to the orthogonal axes can be done by complex vector algebra or matrix algebra. For theoretical analysis and physical understanding, the transformation based on the complex vector would be easier compared to that on the matrix algebra. But, for the computer simulation and programming of the real time control software of the electric machine, the transformation with the matrix algebra is more convenient [29]. In this thesis transformation based on matrix algebra is used.

In the viewpoint of the magnitude of variables at dqn axes compared to that at three phases, there are two methods of transformation. One is called the phase magnitude invariance method, where the magnitude of variables at each phase of the three phase system is the same to that of dq axes components in the balanced steady state. But the power and torque expressed in dqn axes should be multiplied by 3/2 to get the same torque and power expressed in terms of three phase variables. The other one is called the power invariance method, where the magnitude of power or torque expressed in a three phase system is the same as those in dqn axes. But in the power invariance method, the magnitude of variables at each dq axis is $\sqrt{3/2}$ times that of variables at a three phase system [29].

The reference frame theory can be used to simplify the analysis of electric machines and also to facilitate the simulation and digital implementation of control schemes. [28].

In this thesis, for the convenience of comparison of simulation results with calculated values, and also for easy conversion to three phase variables to dq variables, the phase magnitude invariance method is used.

Based on speed of reference frame there are four main reference frames of motion, which could be used to model induction machines. These are arbitrary reference frame, stationary reference frame, rotor reference frame and synchronous reference frame. The two commonly employed coordinate transformations with induction machine are the stationary and the synchronous reference frame [31].

i. **Stationary reference frame**: In this case the dq axis is not rotate. So, the reference frame speed is zero (ω =0). It is best suited for studying stator variables only, for example variable speed stator fed IM drives, because stator d axis variables are exactly identical to stator phase A variable.

ii. **Rotor reference frame**: The reference frame speed is equal to the rotor speed ($\omega = \omega_r$). Since in this reference frame the d axis of the reference frame is moving at the same relative speed as the rotor phase A winding and coincident with its axis, it is best suited when analysis is confined to rotor variables as rotor d axis variable is identical to phase-rotor variables.

iii. **Synchronous reference frame**: When the reference frame is rotating at synchronous speed, both the stator and rotor are rotating at different speeds relative to it. The reference frame speed is equal to synchronous speed ($\omega = \omega_e$). Synchronously rotating reference frame is suitable when analog computer is employed because both stator and rotor dq quantities becomes steady DC quantities.

iv. **Arbitrary reference frame**: In this reference frame speed is unspecified ω and the dq axis can rotate at an arbitrary speed, there is no relative speed between the four coils d_s, q_s, d_r, q_r.

2.2.2 Clarke and Park transformations

The Clarke and Park transformation is a transformation of coordinates that designed to transform multiphase stationary coordinate system in to two axes stationary and rotating coordinate system respectively. The stationary two phase variables of Clarke's transformation are denoted as α and β. As shown in figure 2.5 the α axis coincides the phase a axis and the β axis lags the α axis by 90°. The transformation is bidirectional, third variable known as the zero sequence component is added [27].

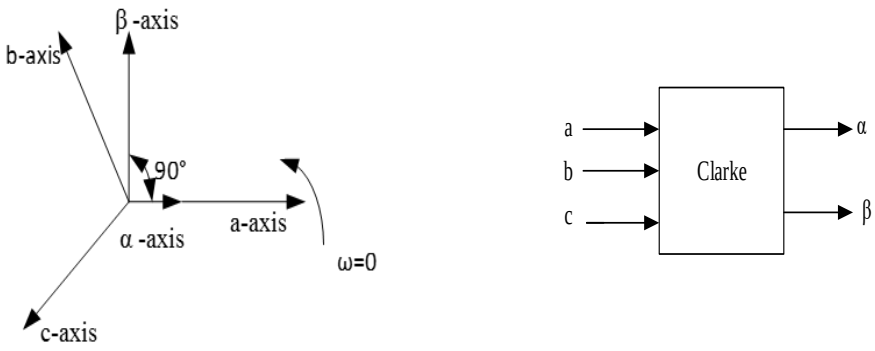


Figure 2.5 Relationship between αβ and abc quantities

$[f_{\alpha\beta0}]$=[$T_{\alpha\beta0}$] $[f_{abc}]$……..……..……..…...…(2.28)

$[f_{abc}]$= $[T_{\alpha\beta0}]^{-1}$[$f_{\alpha\beta0}$]……..……...……..……(2.29)

f_{abc} and $f_{\alpha\beta0}$ represents stator or rotor variables such as voltage,current,flux of Ac machines.Where the transformtion matrix $[T_{\alpha\beta0}]$ is given by

$$[T_{\alpha\beta0}]=\frac{2}{3}\begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \quad \text{……..…....……(2.30)}$$

The inverse transformtion is

$$[T_{\alpha\beta0}]^{-1}=\begin{bmatrix} 1 & 0 & 1 \\ -1/2 & \sqrt{3}/2 & 1 \\ -1/2 & -\sqrt{3}/2 & 1 \end{bmatrix} \quad \text{……...…...……(2.31)}$$

$[f_{\alpha\beta0}]^T$= $[f_\alpha\ f_\beta\ f_0]$……...……..……...……..……(2.32)

$[f_{abc}]^T$= $[f_a\ f_b\ f_c]$……...…...……..…....……(2.33)

In this thesis Clarke transformation is used in SVPWM pulse generator, since the input of SVPWM pulse generator from external source is in the form of αβ. The induction motor considered in this thesis is balanced three phase. Therefore; the zero sequence component is null. Rotating transformation from abc to dq coordinates, also called 3/2 rotating transformation, projects balanced three phase quantities (voltages or currents) to rotating two axis coordinates at a given angular velocity [26]

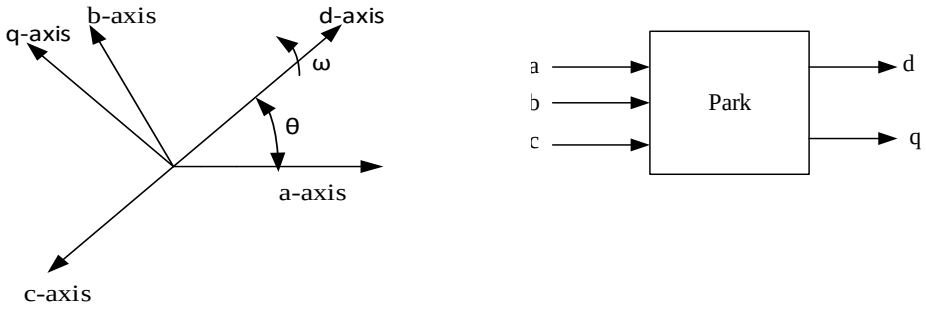


Figure 2.6 Relationship between the dq and abc quantities

The transformation matrix to convert a variable in a three phase axis into the variable in a rotating axis with arbitrary speed can be derived as follows [25, 26, 27, 28, 29]. The form is similar to equation 2.28 but the transformation matrix is different as

$[f_{dq0}]=[\,T_{dq0}\,]\,[f_{abc}]$……..……..……..……(2.34)

$[f_{abc}]\ =[T_{dq0}]^{-1}\,[f_{dq0}]$……..……...……(2.35)

$$[T_{dq0}]=C_{dq0}\begin{bmatrix}\cos(\theta) & \cos(\theta-\frac{2\pi}{3}) & \cos(\theta+\frac{2\pi}{3})\\ \sin(\theta) & \sin(\theta-\frac{2\pi}{3}) & \sin(\theta+\frac{2\pi}{3})\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix}$$……...……(2.36)

The inverse transformtion is

$$[T_{dq0}]^{-1}=C_{abc}\begin{bmatrix}\cos(\theta) & \sin(\theta) & 1\\ \cos(\theta-\frac{2\pi}{3}) & \sin(\theta-\frac{2\pi}{3}) & 1\\ \cos(\theta+\frac{2\pi}{3}) & \sin(\theta+\frac{2\pi}{3}) & 1\end{bmatrix}$$……..……..……(2.37)

Where $C_{dq0}\times C_{abc}={}^{2}/_{3}$

Note that the transform matrices and multiplying factors are the same for both voltages and currents.

In the original Park's transformation, the factor C_{dq0} for abc to dq0 transformation is 2/3, while the factor C_{abc} for dq0 to abc transformation is 1. In some publications, however, C_{dq0} and C_{abc} are both defined to be $\sqrt{{}^{2}/_{3}}$ [26].

In this thesis, C_{dq0} is defined as ${}^{2}/_{3}$ and C_{abc} as 1.

$[f_{dq0}]^{T}=[f_d\ f_q\ f_0]$……..……(2.38)

The relation between park and clark transformation is given as

$$\begin{bmatrix}[f_d]\\ [f_q]\end{bmatrix}=\begin{bmatrix}\cos(\theta)\ \sin(\theta)\\ -\sin(\theta)\ \cos(\theta)\end{bmatrix}\begin{bmatrix}[f_\alpha]\\ [f_\beta]\end{bmatrix}$$……..……..……..(2.39)

$$\begin{bmatrix}[f_\alpha]\\[f_\beta]\end{bmatrix}=\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}[f_d]\\[f_q]\end{bmatrix}\quad\text{.....(2.40)}$$

where θ is, the angle between the synchronously rotating reference frame d axis and the stationary frame a axis. The angular velocity ω and the angular displacement θ of arbitrary reference frame are related as

$$\omega=\frac{d\theta}{dt}\quad\text{.....(2.41)}$$

Thus,

$$\theta=\int \omega\, dt\quad\text{.....(2.42)}$$

The above concepts can be checked by using MATLAB/SIMULINK as follows. As shown in figure 2.7 the inputs of abc to dq transformation block are transformation angle (theta) which produces 0-2π radian and three single phase currents $[i_{as}, i_{bs}, i_{cs}]$ with magnitude 220A, 60Hz frequency and shifted by 120° each other. The abc to dq transformation block yields two phase currents $[i_{ds}, i_{qs}]$ with frequency 60 Hz, magnitude 220A and shifted each other by 90° as shown in figure 2.8.The simulation result summarizes the theory discussed above. Clarke transformation is the same as park transformation except transformation angle (theta) is zero.

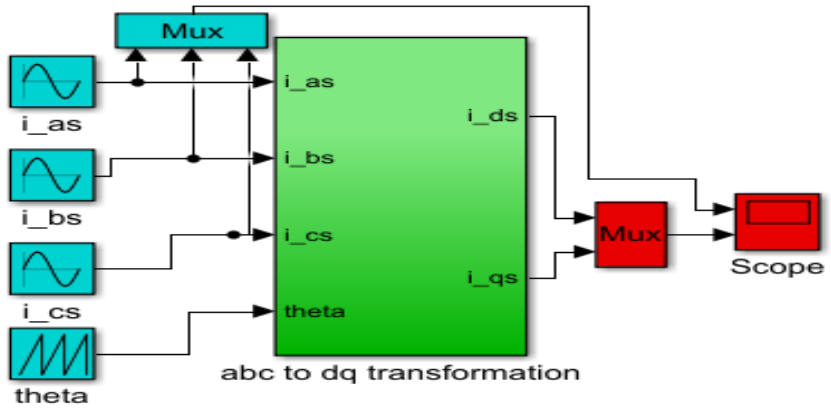


Figure 2.7 Park transformation simulink model

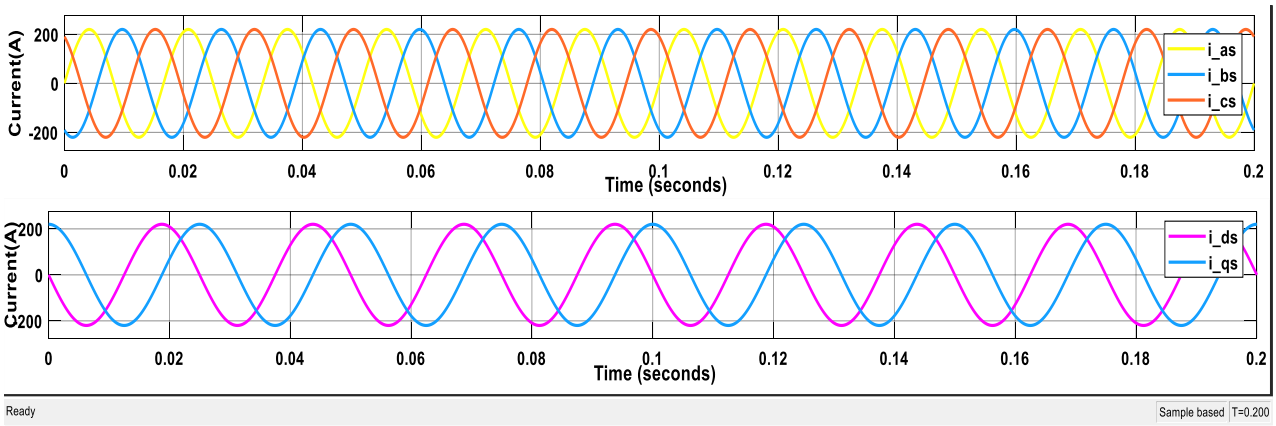


Figure 2.8 Simulation results of park transformation

2.3 DQ dynamic integer order model of three phase induction motor in SRRF

The modeling of the three phase induction machine generally retained, relies on several hypotheses. The first hypothesis consists in assuming that the magneto motive forces created by different stator and rotor phases are spread in a sinusoidal way along the air gap, when those windings are crossed by a constant current. An appropriate dispersion of the windings in space allows reaching this aim. The machine air gap is also supposed to be constantly thick; the notching effects, generating space harmonic are ignored [30].

For this modeling, the following hypotheses about the physical behavior of the materials are considered [30].

- ✓ The skin effect is not taken into account.
- ✓ The temperature in the motor stays constant whatever the operating point is, which leads to constant parameters in mathematical models (stationarity).
- ✓ The magnetic fields are not saturated, are not submitted to the hysteresis phenomenon and are not the center of Foucault's currents (for all practical purposes, the magnetic circuit is leafed through to limit the effects). This allows defining linear inductions.

In fact, the parameters of these models vary in saturation, skin effect and temperature [30].

In the development of an induction motor model, a three phase induction motor with symmetrical windings is assumed. Using the above transformation methods and assumptions the voltage, flux linkage and torque equations in the synchronously rotating reference frame are given as follows [25, 26, 27, 28, 29,30].

Stator voltage equations

$V_{qs} = R_s i_{qs} + \omega_e \lambda_{ds} + P\lambda_{qs}$……(2.43)

$V_{ds} = R_s i_{ds} - \omega_e \lambda_{qs} + P\lambda_{ds}$…… (2.44)

$V_{0s} = 0 = R_s i_{0s} + P\lambda_{0s}$……(2.45)

Rotor voltage equations

$V'_{qr} = 0 = R'_r i'_{qr} + (\omega_e - \omega_r)\lambda'_{dr} + P\lambda'_{qr}$……(2.46)

$V'_{dr} = 0 = R'_r i'_{dr} - (\omega_e - \omega_r)\lambda'_{qr} + P\lambda'_{dr}$…… (2.47)

$V'_{0r} = 0 = R'_r i'_{0r} + P\lambda'_{0r}$……(2.48)

Where P=$\frac{d}{dt}$

Stator flux linkage equations

$\lambda_{qs} = L_{ls}\, i_{qs} + L_m\left(i_{qs} + i'_{qr}\right)$……(2.49)

$\lambda_{ds} = L_{ls}\, i_{ds} + L_m(i_{ds} + i'_{dr})$……(2.50)

$\lambda_{0s} = 0 = L_{ls}\, i_{0s}$ ……(2.51)

Rotor flux linkage equations

$\lambda'_{qr} = L_{lr}\, i'_{qr} + L_m \left(i_{qs} + i'_{qr}\right)$……..…..……(2.52)

$\lambda'_{dr} = L_{lr} i'_{dr} + L_m \left(i_{ds} + i'_{dr}\right)$……..……..……(2.53)

$\lambda'_{0r} = 0 = L_{lr}\, i'_{0r}$ ……..……..……(2.54)

Mechanical equation

$T_e = J\frac{d\omega_r}{dt} + F\omega_r + T_L$……..…….....……..(2.55)

Electromagnetic torque equations

$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(i_{qs}\lambda_{ds} - i_{ds}\lambda_{qs}\right)$……..……..……(2.56a)

$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right) L_m\left(i_{qs}i'_{dr} - i_{ds}i'_{qr}\right)$……..……....(2.56b)

$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\left(\lambda'_{qr}i'_{dr} - \lambda'_{dr}i'_{qr}\right)$……..……..……(2.56c)

The zero sequence component becomes zero because symmetrical induction motor is considered in this thesis.

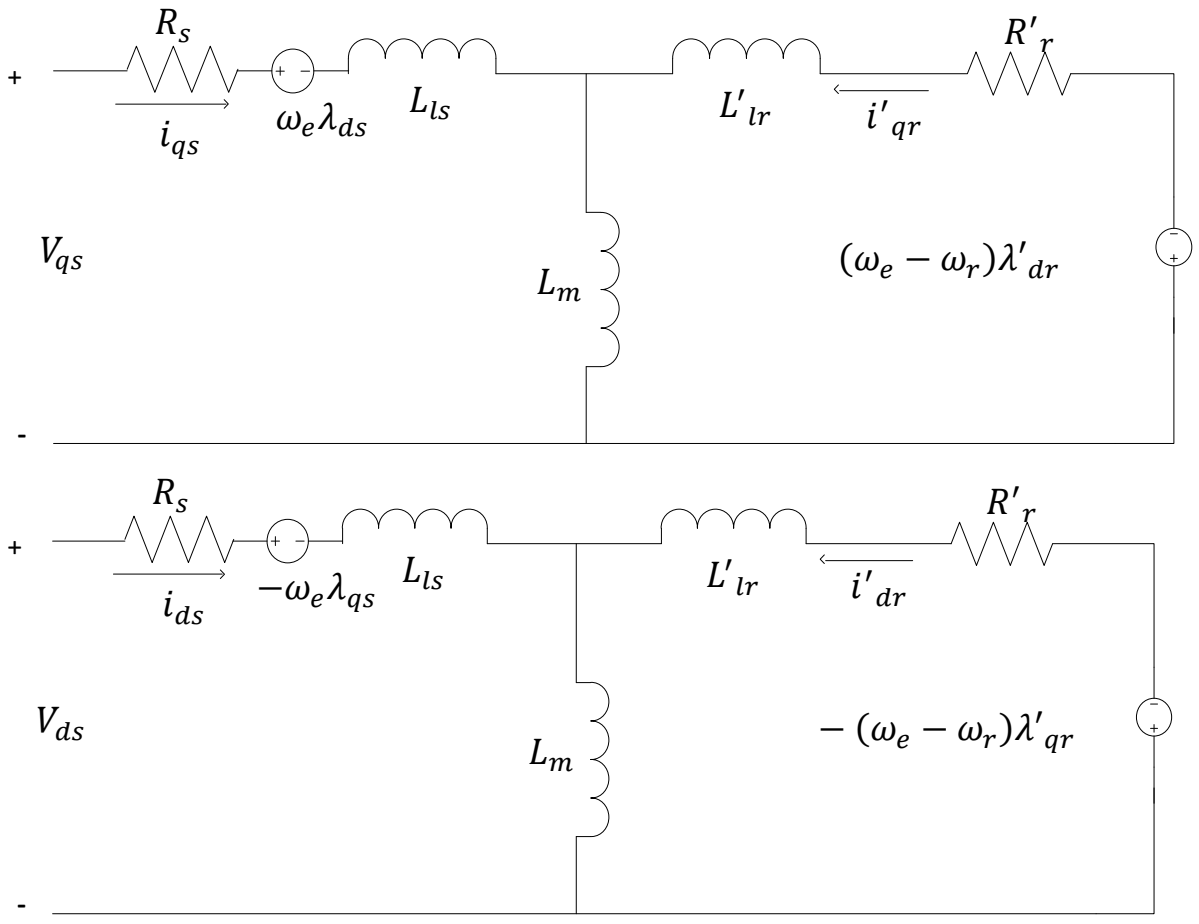


Figure 2.9 SRRF equivalent circuit of 3-phase symmetrical induction motor

2.4 DQ dynamic fractional order model of three phase induction motor in SRRF

After having the well-known integer order model, it is possible to write dimensionless fractional order model of induction motor and determine the unknown order of differentiation.

Stator voltage equations

$$V_{qs} = R_s i_{qs} + \omega_e \lambda_{ds} + \frac{d^\alpha}{dt^\alpha}\lambda_{qs} \quad (2.57)$$

$$V_{ds} = R_s i_{ds} - \omega_e \lambda_{qs} + \frac{d^\alpha}{dt^\alpha}\lambda_{ds} \quad (2.58)$$

$$V_{0s} = 0 = R_s i_{0s} + \frac{d^\alpha}{dt^\alpha}\lambda_{0s} \quad (2.59)$$

Rotor voltage equations

$$V'_{qr} = 0 = R'_r i'_{qr} + (\omega_e - \omega_r)\lambda'_{dr} + \frac{d^\alpha}{dt^\alpha}\lambda'_{qr} \quad (2.60)$$

$$V'_{dr} = 0 = R'_r i'_{dr} - (\omega_e - \omega_r)\lambda'_{qr} + \frac{d^\alpha}{dt^\alpha}\lambda'_{dr} \quad (2.61)$$

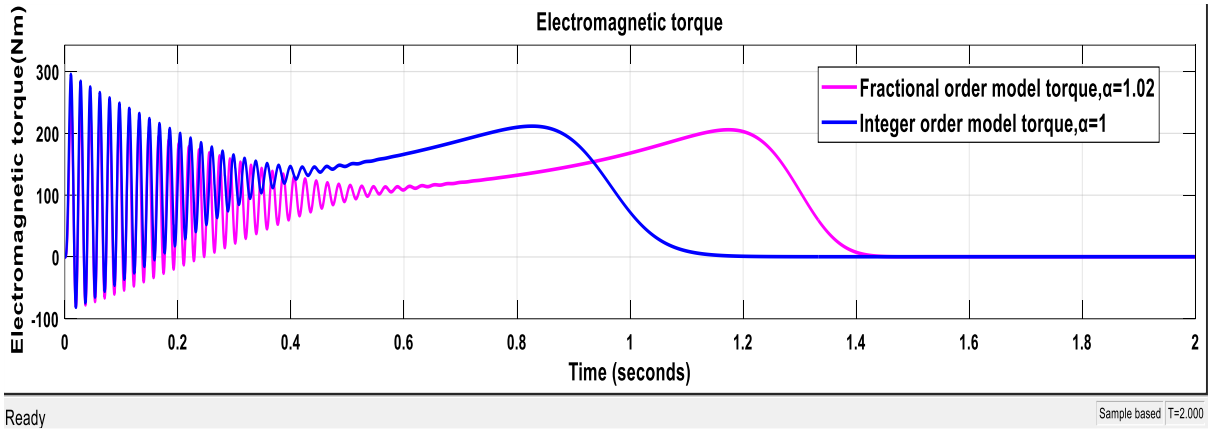
$$V'_{0r} = 0 = R'_r i'_{0r} + \frac{d^\alpha}{dt^\alpha}\lambda'_{0r} \quad (2.62)$$

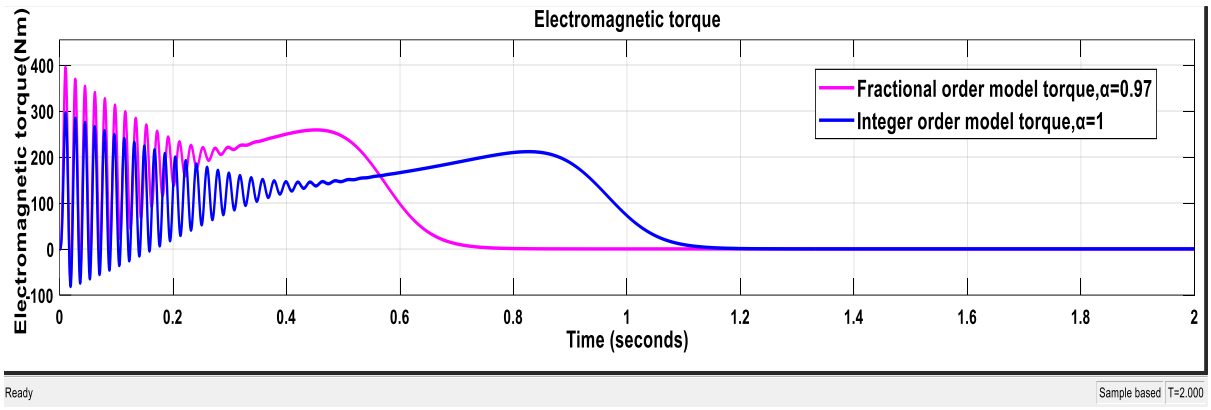
Where α is order of differentiation

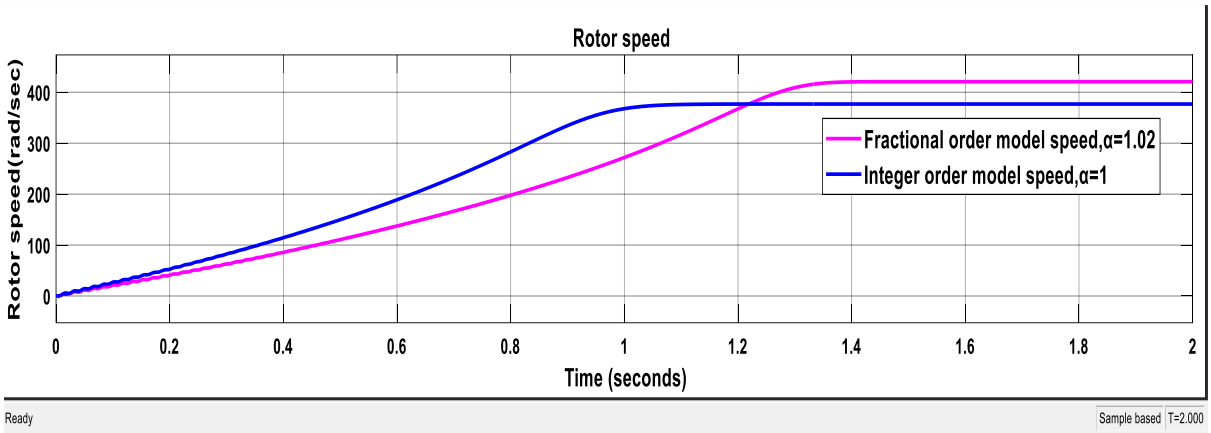
Flux linkage, electromagnetic torque, and Mechanical equations are the same as integer order model, because these equations have no any derivative term.

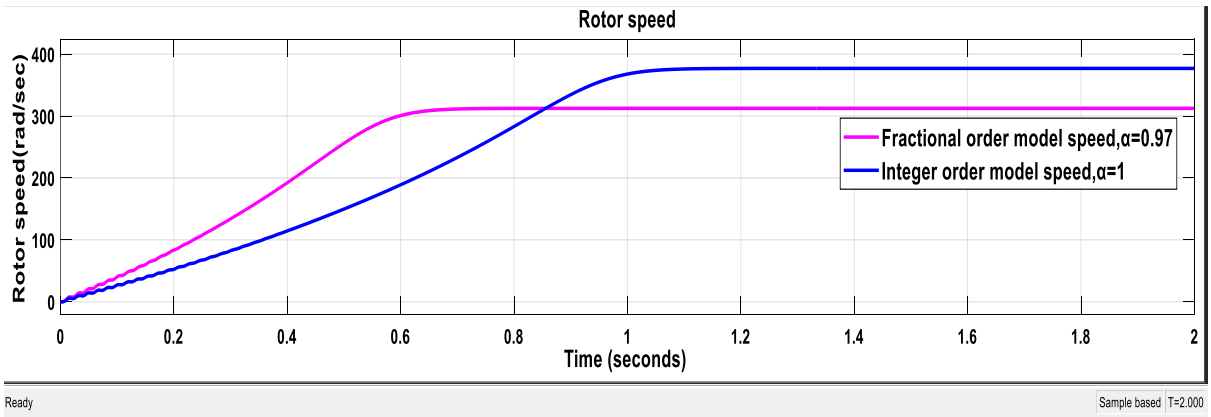
2.5 Fractional order model investigation and induction motor model validation

Once the integer and dimensionless fractional model of induction motor have been determined; model validation and determination of differentiation order (α) for fractional order model must be performed before designing controllers. The parameters of the induction motors used for simulation are listed under table C.1 of appendix C. To illustrate the fractional order model of induction motor, simulation study using different values of differentiation order α is demonstrated. At the initial time instant (t=0), the motor previously de-energized and at standstill, is connected to the respective three phase 60Hz supply as shown in figure A.1 of appendix A, which is used for simulation to determine differentiation order (α). The power source is constructed using a signal generator block from the Simulink library. The load torque is assumed to be 0Nm (no load). Figures 2.10(a-c) show the results of MATLAB simulation using the Simulink model. Table 2.1 summarizes the speed and electromagnetic torque simulation results of induction motors for different order of integration (α).For simulation three different induction motors having 2.24KW, 7.457KW and 7.5 KW power are used. The following figures are speed-time and torque-time simulation results of three different induction motors.

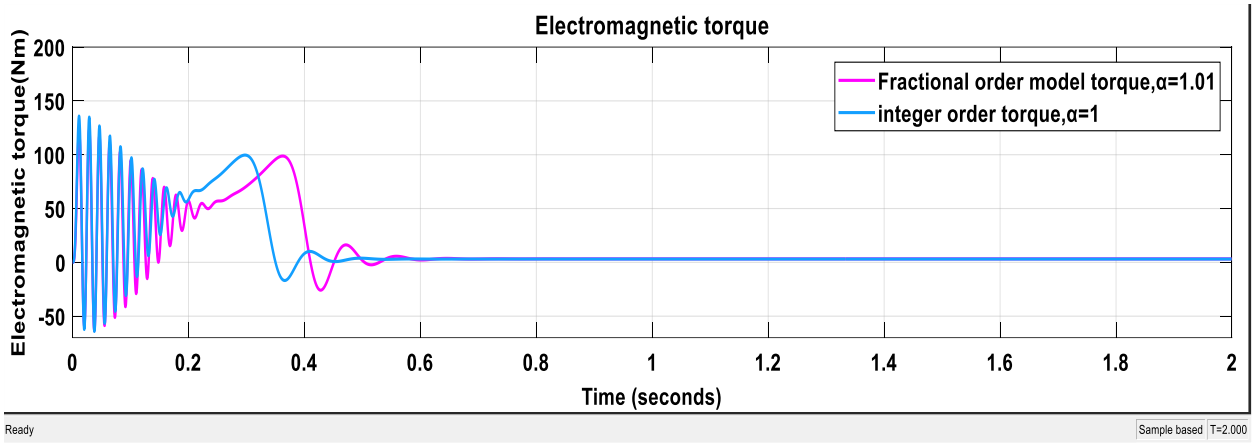

Electromagnetic torque
Electromagnetic torque(Nm)
Time (seconds)
Fractional order model torque,α=1.02
Integer order model torque,α=1
Ready
Sample based
T=2.000

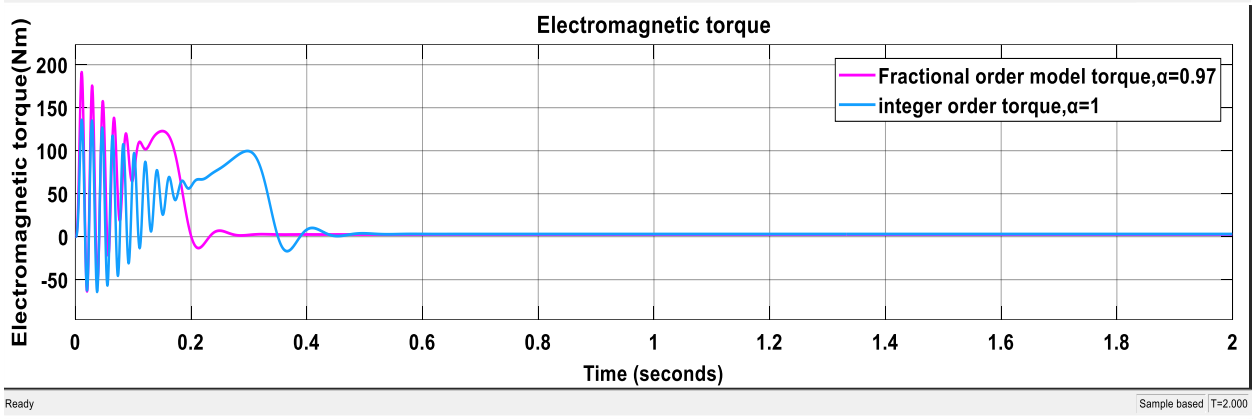

Electromagnetic torque
Electromagnetic torque(Nm)
Time (seconds)
Fractional order model torque,α=0.97
Integer order model torque,α=1
Ready
Sample based
T=2.000

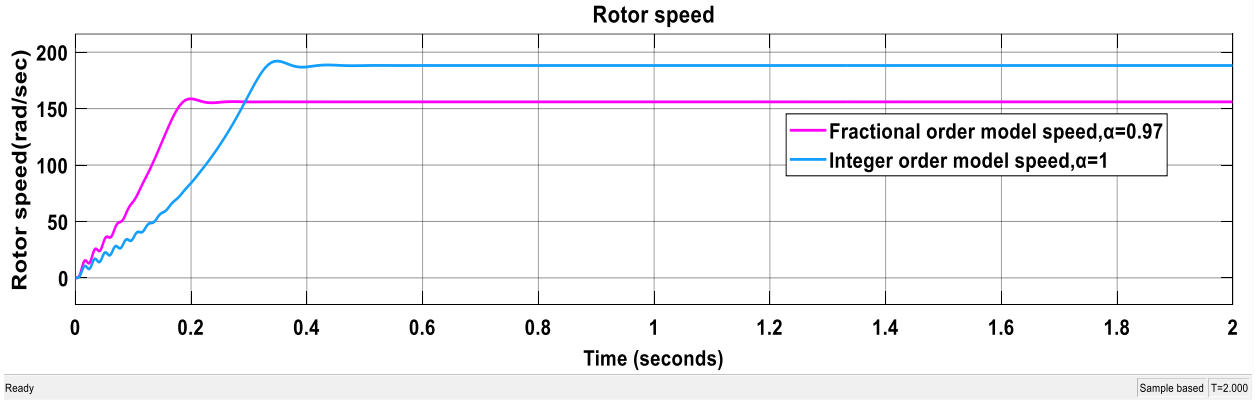

Rotor speed
Rotor speed(rad/sec)
Time (seconds)
Fractional order model speed,α=1.02
Integer order model speed,α=1
Ready
Sample based
T=2.000

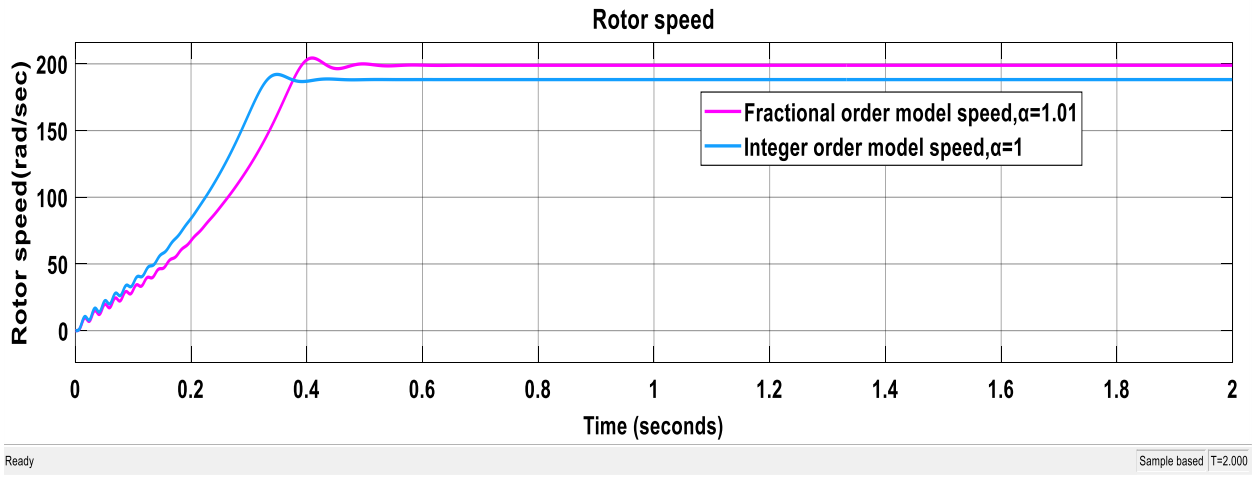

Rotor speed
Rotor speed(rad/sec)
Time (seconds)
Fractional order model speed,α=0.97
Integer order model speed,α=1
Ready
Sample based
T=2.000

(a)

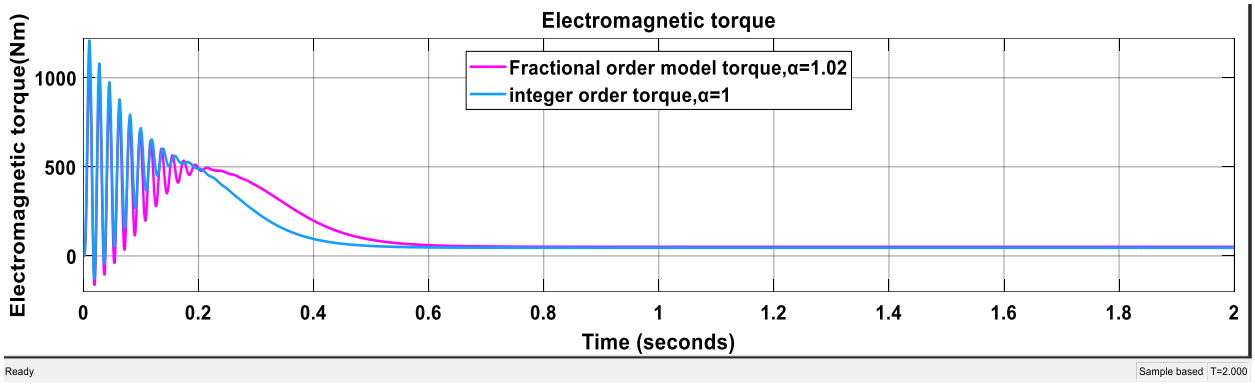

Electromagnetic torque
Electromagnetic torque(Nm)
Time (seconds)
Fractional order model torque,α=1.01
integer order torque,α=1
Ready
Sample based
T=2.000

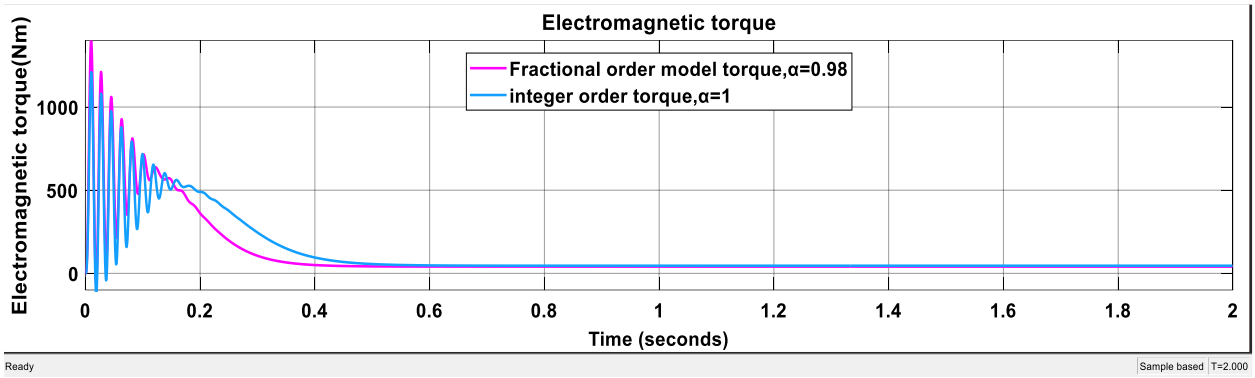

Electromagnetic torque
Electromagnetic torque(Nm)
Time (seconds)
Fractional order model torque,α=0.97
integer order torque,α=1
Ready
Sample based
T=2.000

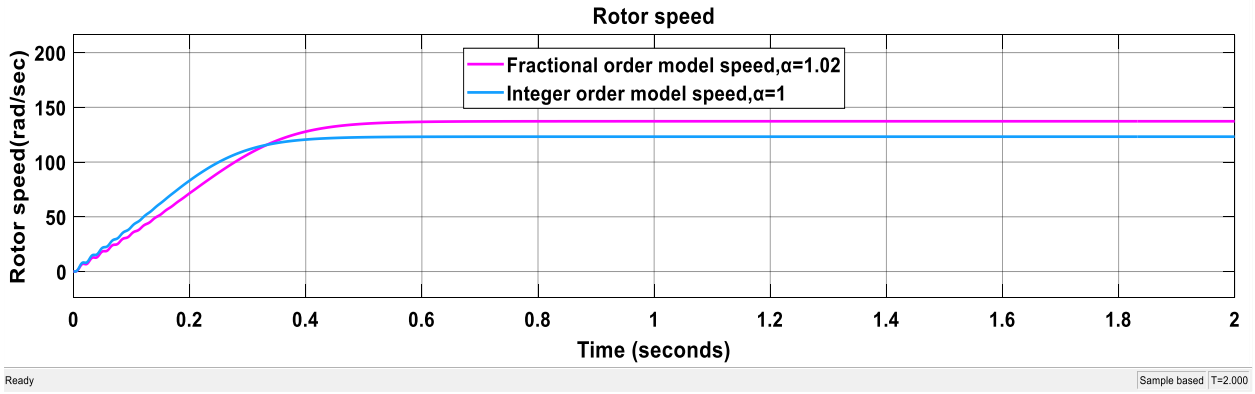

Rotor speed
Rotor speed(rad/sec)
Time (seconds)
Fractional order model speed,α=0.97
Integer order model speed,α=1
Ready
Sample based
T=2.000

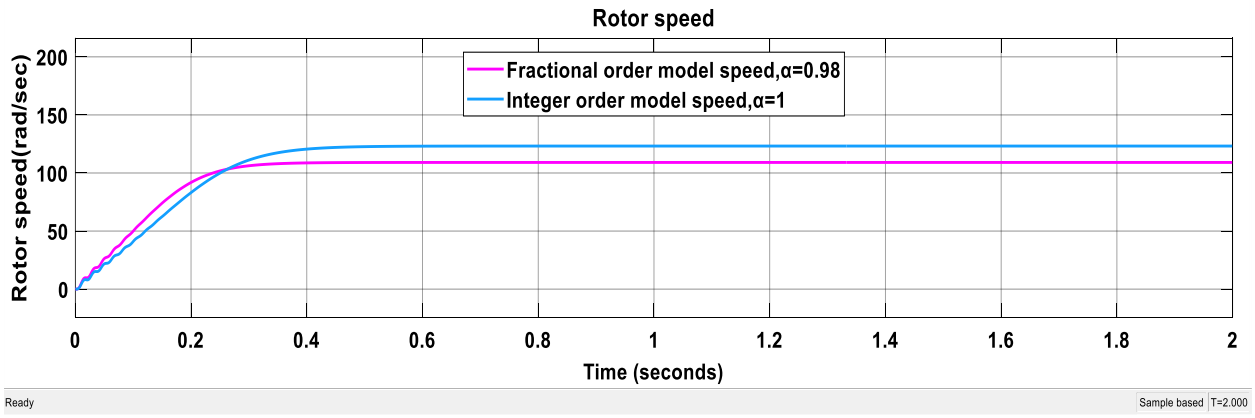

Rotor speed
Rotor speed(rad/sec)
Time (seconds)
Fractional order model speed,α=1.01
Integer order model speed,α=1
Ready
Sample based
T=2.000

(b)









(c)

Figure 2.10 Electormagnetic torque-time and Speed-time graph of different motors for different α values (a) motor 1,(b) motor 2 (c) motor 3

The induction motor works on the principle that electrical power supplied to the stator will produce a rotating magnetic flux through its iron and air gap. This rotating flux, in turn, induces currents in the rotor conductors. There currents interact with the original flux to generate a torque to make the rotor rotate. The stator has insulated windings embedded in the inner slots of the iron periphery. The placement and connection of these windings determines the number of electrical poles of the motor. As a sinusoidal excitation is applied to the windings of stator, a sinusoidal flux is established in the air gap rotating at a speed given by

$$N_S = \frac{120 f_S}{P} \quad \text{(2.63)}$$

Where N_S represents synchronous speed in rpm, f_S is the frequency of the applied excitation in Hz, and P is the number of poles. It is also known as the synchronous speed.

Different criteria are used to select the best value of α. The speed response of fractional orders at no load is not equal to the value calculated by equation 2.63. As seen from table 2.1 as the order of integration (α) increases peak torque decreases, rated speed at no load increases, settling time for torque and speed increases. The calculated synchronous speed which is calculated by equation 2.63 is equal to the simulation result at α=1, which is an integer order. Therefore, the optimized model of induction motor is an integer order model.

Table 2.1 Performance of induction motors for different order of integration (α)

Motor	α value	Peak torque (Nm)	Settling time for torque (sec)	Final value of torque at no load (Nm)	Settling time for speed (sec)	Final value of speed at no load (rad/sec)	Calculated synchronous (rad/sec)
1	1.02	247.8	1.43	0.4211	1.385	420.78	
	1	295.6	1.13	0.3769	1.11	376.94	
	0.97	395.1	0.718	0.3117	0.641	312.41	376.99
2	1.01	123.9	0.65	3.239	0.55	198.9	
	1	135.9	0.58	3.064	0.452	188.4	
	0.97	191.4	0.321	2.539	0.281	156	188.5
3	1.02	603.0	0.78	51.95	0.712	139.6	
	1	724.5	0.53	46.47	0.481	124.9	
	0.98	887.4	0.394	41.02	0.33	110.3	125.66

CHAPTER THREE

3. Control of three phase induction motor

In this chapter motor control strategies, Genetic algorithm optimization technique, and concept of intelligent control techniques will be discussed. Finally, integer and fractional order PI controller, and ANFIS controller using fractional order PI controller data will be designed. GA optimization technique will be used to get optimized parameters of both fractional and integer order PI speed controllers. Fractional order PI controller will be designed using FOMCON toolbox.

3.1 Introduction to three phase induction motor control strategies

Induction motors are the most widely used electrical motors due to their reliability, low cost and robustness. However, induction motors do not inherently have the capability of variable speed operation. Due to this reason, earlier dc motors were applied in most of the electrical drives. But, the recent developments in speed control methods of the induction motor have lead to their large scale use in almost all electrical drives.

Induction motor control problems have attracted the attention of researchers for many years. Most of the earlier researches are based on classical control theory and electric machine theory, using precise mathematical models of the induction motor. As shown in Figure 3.1, an induction motor control system consists of the controller, sensors, inverter, and the induction motor. It can be seen that a study of induction motor control involves three main electrical engineering areas: control, power electronics, and electrical machines [26].

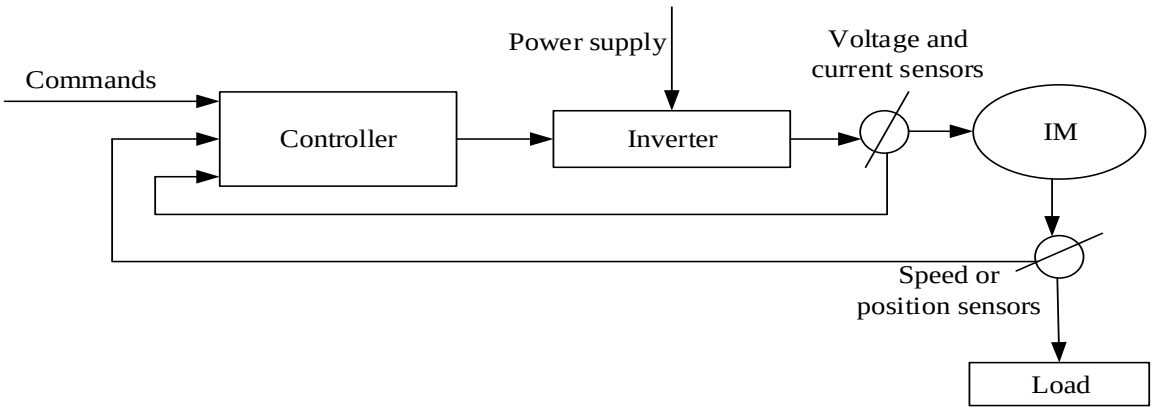


Figure 3.1 Induction motor control system [26].

There are many strategies of induction motor control. The most widely used strategies are **volts per hertz control**, **constant slip control**, **field oriented control**. The first, volts per hertz control, is designed to accommodate variable speed commands by using the inverter to apply

a voltage of correct magnitude and frequency so as to approximately achieve the commanded speed without the use of speed feedback. The second strategy is constant slip control. In this control, the drive system is designed so as to accept torque command input, and therefore speed control requires an additional feedback loop. Although this strategy requires the use of speed sensor, it is highly robust with respect to changes in machine parameters and results in high efficiency of both of the machine and inverter. One of the disadvantages of this strategy is that in closed loop speed control situations the response can be somewhat sluggish. The third strategy which also considered in this thesis is field oriented control. In this method nearly instantaneous torque control can be obtained, allowing the drive to act as a torque transducer. The disadvantage of this strategy is that in its direct form the sensor requirements are significant, and in its indirect form it is sensitive to parameter variations unless online parameter estimation or other steps are taken [25].

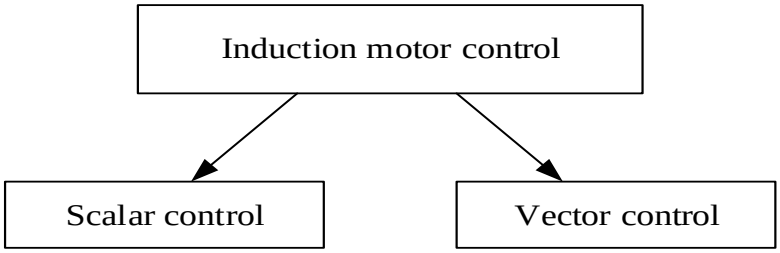


Figure 3.2 Induction motor control strategy

3.2 Vector control of induction motor

The essence of field oriented control is the decoupled control of the rotor flux λ_r and electromagnetic torque T_e of the motor to achieve high dynamic performance. Using the rotor flux orientation, the stator current of the motor can be decomposed into a flux producing component, which produces the rotor flux λ_r ,and a torque producing component, which produces the torque T_e. These two components are then controlled independently [28]. Vector control also known as field oriented control (FOC).

FOC consists of controlling the stator current represented by a vector control is based on projections which transform a three phase time and speed dependent system into a two coordinate dq time invariant system. These projections lead to a structure like that of a separately excited dc machine control. This requires information regarding the magnitude and position of rotor flux vector.

The synchronously rotating reference frame can be aligned with the stator flux or rotor flux or magnetizing flux (field flux) space vectors respectively. Accordingly, vector control is also

known as stator flux oriented control or rotor flux oriented control or magnetizing flux oriented control. Generally, in induction motors, the rotor flux oriented control is preferred. This is due to the fact that by aligning the SRRF with the rotor flux, the vector control structure becomes simpler and dynamic response of the drive is observed to be better than any other alignment of the SRRF [33].

If the field angle is calculated by using terminal voltages and currents or flux sensing windings and rotor speed, then it is known as direct FOC. The field angle can also be obtained by using rotor position measurement and slip position by partial estimation with only machine parameters but not any other variables such as voltages or currents, this class of control scheme is known as indirect FOC. The rotor field angle is obtained by submission of rotor speed and slip frequency [34]. Equation (2.56c) can be expressed as

$T_e = -(\frac{3}{2})(\frac{P}{2}) \left|\lambda'_{qdr}\right| \left|i'_{qdr}\right| \sin\theta$ ..... (3.1)

From equation (3.1) for a given magnitude of flux linkage, torque is maximized when the flux linkage and current vectors are perpendicular. Thus, it is desirable to keep the rotor flux linkage vector perpendicular to the rotor current vector. In steady state the rotor flux linkage current vector always perpendicular for all singly feed induction motors [25]. To see this, consider rotor voltage equation and solve for rotor currents becomes

$$i'_{qr} = -\frac{1}{R'_r}(\omega_e - \omega_r)\lambda'_{dr} \quad \text{.....(3.2)}$$

$$i'_{dr} = \frac{1}{R'_r}(\omega_e - \omega_r)\lambda'_{qr} \quad \text{.....(3.3)}$$

The dot product of the rotor flux linkge and rotor current vectors can be expressed as

$$\lambda'_{qdr} \cdot i'_{qdr} = \lambda'_{qr} i'_{qr} + \lambda'_{dr} i'_{dr} \quad \text{.....(3.4)}$$

Substitution of (3.2) and (3.3) into (3.4) reveals that this dot product is zero whereupon. It may be concluded that the rotor flux and rotor current vectors, as expressed in the synchronous reference frame, are perpendicular. Furthermore, if they are perpendicular in the synchronous reference frame, they are perpendicular in every reference frame. In this sense, in the steady state every singly exited induction machine operates with an optimal relative orientation of the rotor flux and rotor current vectors. However, the defining characteristics of a field oriented drive is that this characteristic is maintained during transient conditions as well. It is this feature which results in the high transient performance capabilities of this class of drive [25]. In both direct and indirect field oriented control the method to achieve the condition that the rotor flux

and rotor current vectors are always perpendicular is twofold [25,27]. The first part of strategy is to insure that

$\lambda'_{qr}=0$…… (3.5)

The second is to insure that

$i'_{dr}=0$…… (3.6)

Clearly, if (3.5) and (3.6) hold during transient conditions, then by (3.4) the rotor flux linkage and rotor current vectors are perpendicular during those same conditions. By suitable choice of θ_e on an instantaneous basis, (3.5) can always be satisfied by choosing the position of the synchronous reference frame to put all of the rotor flux linkage in the d axis. Satisfying (3.6) can be accomplished by forcing the d axis stator current to remain constant. To see this, consider the d axis rotor voltage equation.

$0 = R'_r i'_{dr} - (\omega_e - \omega_r)\lambda'_{qr} + P\lambda'_{dr}$…… (3.7)

by suitable choice of reference frame (3.5) is achieved; therefore λ'_{qr} can be set to zero in (3.7) to yield

$0 = R'_r i'_{dr} + P\lambda'_{dr}$…… (3.8)

Then, substitution of the d axis rotor flux linkage equation (2.53) in to (3.8) and rearranging yields

$Pi'_{dr} = -\frac{R'_r}{L_{rr}} i'_{dr} - \frac{L_m}{L_{rr}} Pi_{ds}$…… (3.9)

Equation (3.9) is a stable first order differential equation with respect to i'_{dr} with Pi_{ds} as input. If i_{ds} keep constant then i'_{dr} will go to, and stay at zero, regardless of other transient that may be taking place.

Some implications of (3.5) and (3.6) are the following

$\lambda_{ds} = L_{ss}\, i_{ds}$…… (3.10)

$\lambda'_{dr} = L_m i_{ds}$ …… (3.11)

$T_e = -(\frac{3}{2})(\frac{P}{2})(\lambda'_{dr} i'_{qr})$…… (3.12)

$i'_{qr} = -\frac{L_m}{L_{rr}} i'_{qs}$…… (3.13)

Combining (3.12) and (3.13) yields

$T_e = (\frac{3}{2})(\frac{P}{2})\frac{L_m}{L_{rr}}\lambda'_{dr} i_{qs}$…… (3.14)

From q axis voltage

$\omega_e - \omega_r = \omega_{sl} = \frac{-R'_r i'_{qr}}{\lambda'_{dr}}$…… (3.15)

Combining (3.11), (3.13) and (3.15) yields

$$\omega_e - \omega_r = \omega_{sl} = \frac{R'_r i_{qs}}{L_{rr} i_{ds}} \quad \text{(3.16)}$$

3.2.1 Direct vector control of induction motor

In the direct vector control method, the position of the rotor flux is identified and the torque component current is applied 90 degrees ahead spatially in the direction of the rotation. The magnitude of the rotor flux linkage, which is chosen as the orientation of the flux linkage, should be controlled simultaneously by the flux component current together with the torque component current from the stator of the induction motor [29].

Direct vector control method determines the magnitude and position of the rotor flux vector by direct flux measurement or by a computation based on terminal conditions. It also called flux feedback control method in which required information regarding the rotor flux is obtained by means of direct flux measurement or estimation [35].

The instantaneous position of the rotor flux linkage can be identified by measurement of air gap flux by Hall Effect sensors, measurement of air gap voltage by sensing coils, estimation of the rotor flux linkage with terminal voltages and line current [29]. Alternatively, in place of flux sensors, the flux models can also be used for which the stator currents and voltages become the feedback signals and the rotor flux angle is given as its estimated output [35].

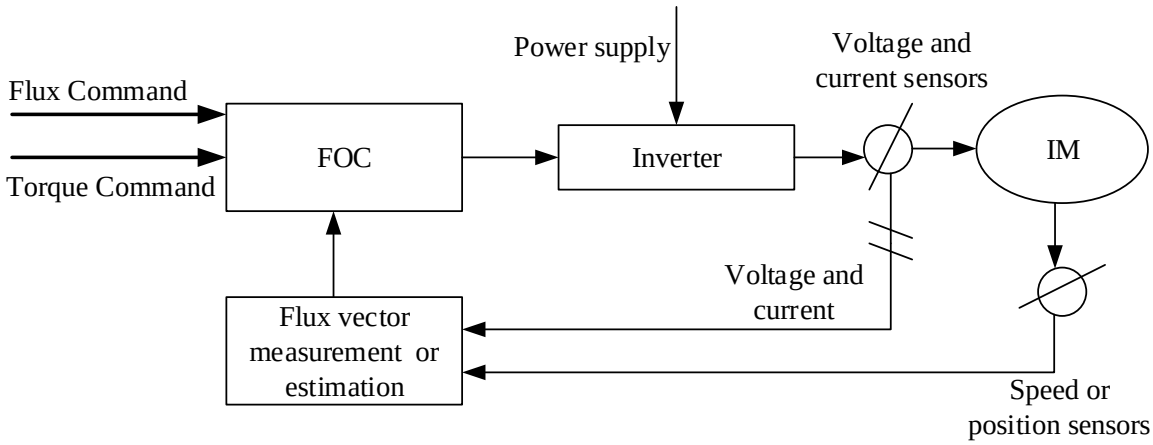


Figure 3.3 Simplified direct FOC (34)

Some disadvantages direct FOC are

- Fixing of number of sensors is a tedious job.
- The sensors increase the cost of the machine.
- Drift problem exist because of temperature.
- Poor flux sensing at lower temperature.

These disadvantages lead to another technique called indirect vector control technique.

3.2.2 Indirect vector control of induction motor

The major disadvantage of direct vector method is the need of so many sensors. Fixing so many sensors in a machine is a tedious work as well as costlier. Due to this the scheme is prevented from being used. Several other problems like drift because of temperature, poor flux sensing at lower speeds also persists. Due to these disadvantages and some more related ones, indirect vector control is used. In indirect vector control technique, the rotor position is calculated from the speed feedback signal of the motor. This technique eliminates most of the problems, which are associated with the flux sensors as they are absent.

The indirect FOC is essentially the same as that for the direct FOC, but the method for obtaining the rotor flux angle is different [28]. The rotor flux angle in the indirect FOC can be obtained from the measured rotor speed (ω_r) and calculated slip frequency (ω_{sl}).

The algorithm of indirect vector control is as follows

1. The induction motor is feed by a variable frequency, variable voltage SVPWM inverter as shown in figure 1.1. The motor speed ω_r is compared with the reference speed and the error is produced which is feed to the speed controller. The output of speed controller is electromagnetic torque $T_e{}^*$.
2. The quadrature axis stator current reference $i_{qs}{}^*$ is calculated from electromagnetic torque reference $T_e{}^*$ as

$$i_{qs}{}^* = \left(\frac{2}{3}\right)\left(\frac{2}{P}\right)\left(\frac{L_{rr}}{L_m}\right)\left(\frac{T_e{}^*}{\lambda^* r}\right) \quad (3.17)$$

Where

$\lambda^* r$ is reference nominal flux linkage

Since, $\lambda'_{\mathrm{qr}} = 0$

$$\lambda'_{\mathrm{dr}} = \lambda'_{\mathrm{r}} \quad (3.18)$$

3. The direct axis stator current reference $i_{ds}{}^*$ is obtained from reference rotor flux input $\lambda^*{}_{dr}$.

$$i^*{}_{ds} = \frac{\lambda^*{}_{dr}}{L_m} \quad (3.19)$$

4. The rotor flux position θ_e required for coordinate transformation is obtained from the rotor speed ω_r and slip frequency ω_{sl} is calculated as

$$\theta_e = \int \omega_e dt = \int (\omega_r + \omega_{sl}) dt = \theta_r + \theta_{sl} \quad (3.20)$$

ω_{sl} is calculated by (3.15) or (3.16) using reference current quantities and motor parameters.

5. i^*_{ds} and i^*_{qs} are reference for current controller. The output of current controller is voltage. V^*_{ds} and V^*_{qs} voltage references are converted to $V^*_{\alpha s}, V^*_{\beta s}$, using Clarke transformation and feed to SVPWM inverter to produce the inverter gating signals.

The main role of the speed controller is to keep the motor speed equal to the speed reference input in steady state and to provide a good dynamic response during transients.

3.3 Genetic Algorithm optimization technique

The most popular evolutionary algorithm is the Genetic Algorithm (GA), invented by John Holland at the University of Michigan in 1975 [40]. The basic element of GA is the chromosome. This contains the generic information for a given solution and is typically coded as a binary string. Initially; population of chromosomes created randomly represent number of solutions to a given problem. A fitness function, which is in effect a performance index, is used to select the best solutions in the population to be parents to the offspring's that will comprise the next generation. The fitter the parent, the greater the probability of selection. This emulates the evolutionary process of "survival of the fittest" [37].GA repeatedly modifies a population of individual solutions.

The basic steps of genetic algorithm are as follows.

1. **Start**: Generate a random population of n chromosomes which have a proper form for the problem.
2. **Run fitness function**: Evaluate the fitness function using each chromosome in the population.
3. **Result test**: Judge the n evaluated results. If conditions are satisfied, then GA stops and outputs the best chromosome of current population.
4. **Selection**: Select two or more parent chromosomes from a population according to their fitness (the better the fitness of a chromosome, the greater is its chance of being selected).
5. **Crossover**: With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, the offspring is an exact copy of the parents.
6. **Mutation**: With a mutation probability mutate new offspring at each locus (position in chromosome).
7. **Replace**: Place new offspring in the old population to create a new population and use a new generated population for a further run of the algorithm.
8. **Loop**: Go to step 2.

To implement the GA optimization, first the fitness function must be defined. The fitness function is the objective function that needs to be minimized. In this thesis the objective function contains the function that represents the square of the error between desired and actual

output. Besides the fitness function the number of variables to be optimized should also be specified. The variables to be optimized are PI speed controller's gains (K_p, K_i) and λ in case of fractional order controller to ensure optimal control performance for the induction motor. The block diagram for the entire system is given below.

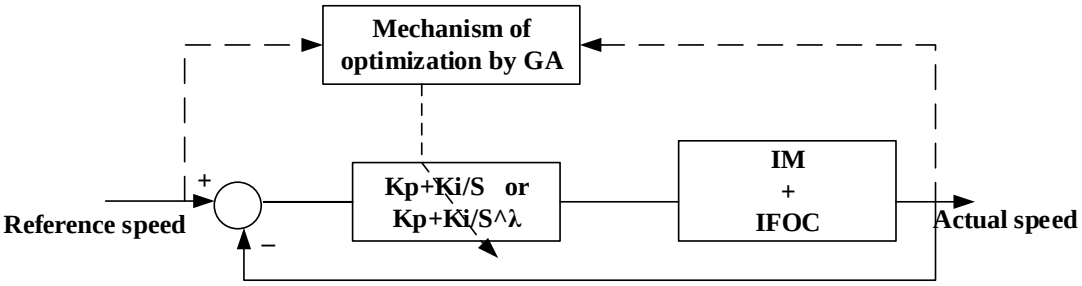


Figure 3.4 Structure of IOPI and FOPI controller optimization using GA technique

Since the objective is to minimize the error between the desired output and the actual output, the fitness function selected is an integral square error (ISE), defined as

J=$\int_0^t e(t)^2 dt = \int_0^t (\omega(t)^* - \omega(t))^2 dt$……..…..…(3.21)

Where $e(t)$is the error, $\omega(t)^*$ and $\omega(t)$ are reference and actual speed respectively.

3.4 Design of FOPI and IOPI speed controllers using GA optimization technique

One of the main challenges in the design of controller for real time applications is getting the optimal values of controller parameters, which play a vital role in determining the performance and optimality of the controller. Commonly, trial and error approach is employed for selecting the parameters of controller, which is not only burdens of design but also results in non-optimal response. Hence, to find the optimal values of controller parameters, genetic algorithm is formulated and applied for control of induction motor. The transfer function of PI controller is

$G_c(s) = K_p + \frac{K_i}{S}$……..……..…… (3.22)

Mostly for industrial applications, integer order controllers are used for controlling purpose. Now day's fractional order PID (FOPID) controller is used for industrial application to improve the system control performances. The most common form of a fractional order PID controller is the $PI^{\lambda}D^{\mu}$ controller [6]. FOPID controller provides extra degree of freedom for not only the need of design controller gains but also design orders of integral and derivative. The orders of integration and derivation are not necessarily integer, but any real numbers. As shown in figure 3.5, the FOPID controller generalizes the conventional integer order PID controller and expands it from point to plane. This expansion could provide much more flexibility in PID control design. The transfer function of such a controller has the following form [42]

$G_c(s) = K_p + \frac{K_i}{S^\lambda} + K_d S^\mu$…….. (3.23)

By making μ=0, and K_d=0 fractional order PI will result as

$G_c(s) = K_p + \frac{K_i}{S^\lambda}$……..…….. (3.24)

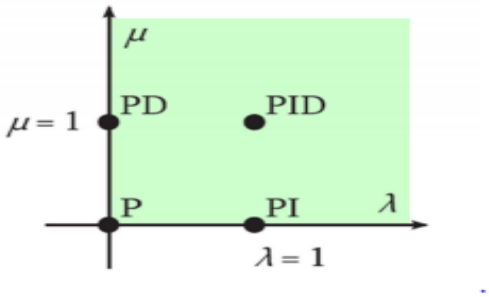
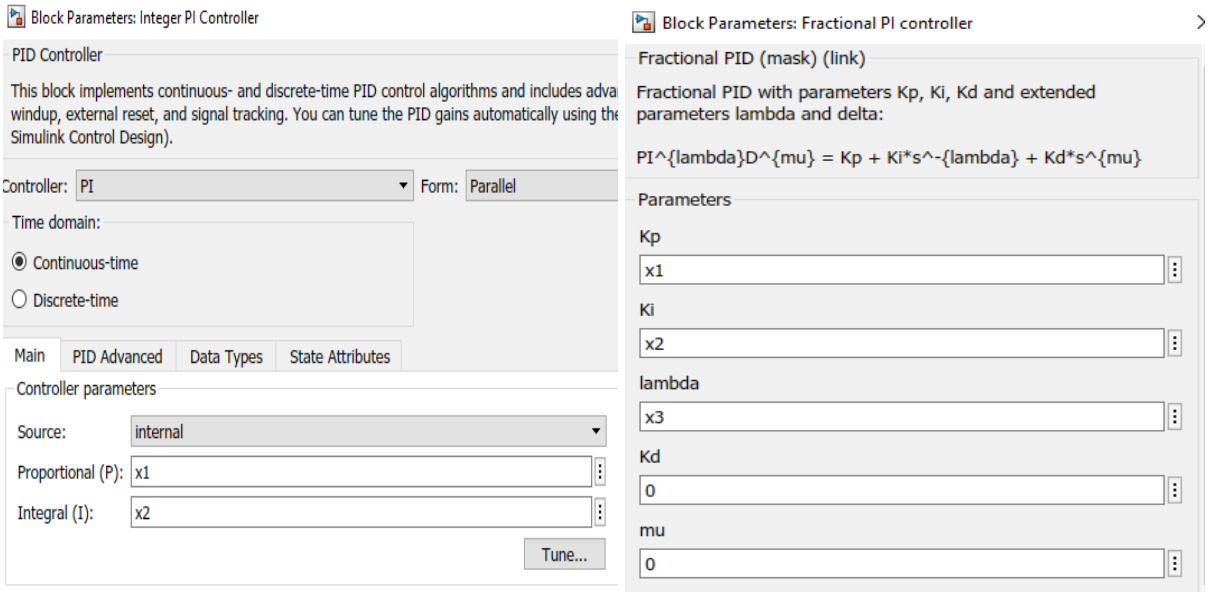


Figure 3.5 General form of a fractional order PID controller

It is Clear, by selecting λ = 1 and μ = 1, a classical PID controller can be recovered. Using λ = 1, μ = 0, and λ = 0, μ = 1, respectively corresponds to the conventional PI & PD controllers. All these classical types of PID controllers are special cases of the $PI^\lambda D^\mu$ controller.

Fractional and integer order PI controllers are optimized by Genetic Algorithm function 'ga' of MATLAB 2017a.The control systems are simulink models. In order to call the simulink model from MATLAB, the simulink model must first be built, and then a MATLAB programming functions are used to call the simulink models. Parameters of the integer order PI controller consist of two variables, namely, proportional gain K_p=x1 and integral gain K_i=x2, which are set in the 'integer PI Controller' block as shown in figure 3.6a. Parameters of the fractional order PI controller consist of three variables, namely, proportional gain K_p =x1, integral gain K_i=x2, and order of integration λ=x3 which are set in the 'fractional PI Controller' block as shown in figure 3.6b. The Simulink model used for determination of controller's parameters is figure 4.1 in the next chapter and MATLAB codes for genetic algorithm optimization is written in appendix B.



(a) (b)

Figure 3.6 Configuration of parameters (a) PI controller, (b) PI^λ controller

After giving the parameters listed in table 3.1 below to GA, the PI and PI^{λ} controllers can be easily tuned and thus system performance can be improved. Parameters of PI and PI^{λ} speed controllers are obtained according to the procedure of GA optimization technique using the following MATLAB commands

[x,fval] = ga(@integer_PI,2 ,[],[],[],[],lb, ub,[],options); *%for PI controller*

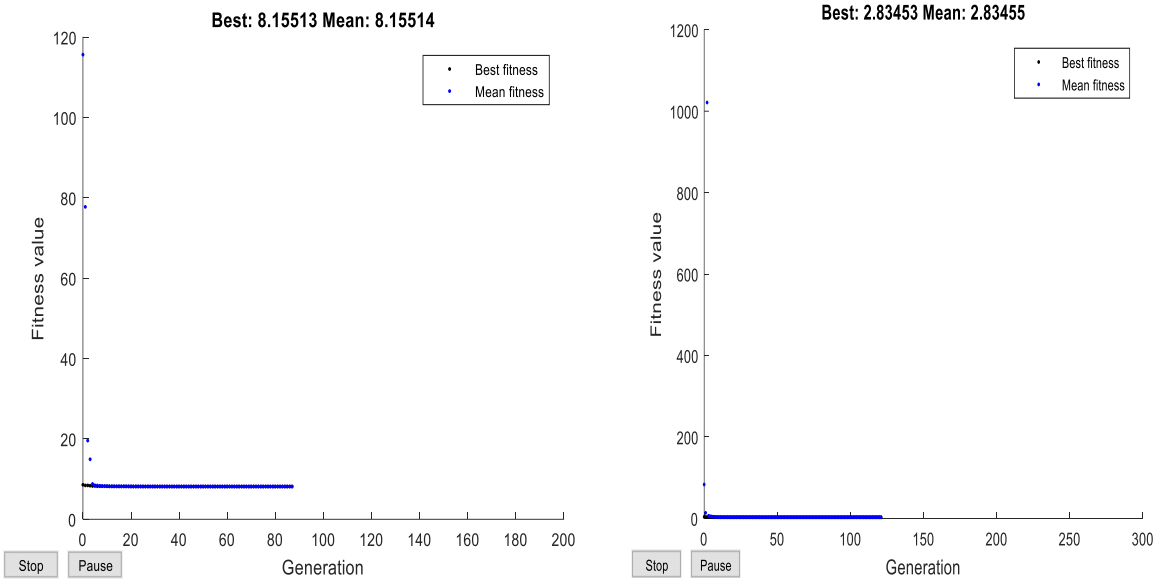
[x,fval] = ga(@fractional_PI,3 ,[],[],[],[],lb, ub,[],options); *%for PI^{λ}controller*

Where x is the PI and PI^{λ} parameters (K_p and K_i and λ in case of PI^{λ} controller), fval is fitness function value and *@integer_PI* and *@fractional_PI* are the function names. Because of initial chromosomes in the genetic algorithm are randomly created, the program may need to be run several times in order to obtain satisfactory results.

The following table 3.1 Summarizes the genetic algorithm parameters chosen for the tuning of PI and PI^{λ}controllers parameters.

Table 3.1 Parameters of GA

GA property	Values/Descriptions
Population size	60
Selection function	Tournament selection
Mutation function	Uniform mutation
Crossover function	Intermediate
Number of generation	200 for PI and 300 for PI^{λ}
Cross over probability	0.8
Mutation probability	0.1
Tolerance	10e-5



(a) (b)

Figure 3.7 Best evaluated outputs of fitness function of (a) PI controller,(b) PI^λ controller

The parameters of PI and PI^{λ} speed controllers obtained according to the procedure of optimization by the technique of GA are given below in table 3.2. The PI parameters for the current controllers are tuned by trial and error. The tuned parameters of current controllers for

direct and quadrature axis components are K_p =25 and K_i =2.75 for both PI and PI^{λ} speed controllers.

Table 3.2 PI and PI^λ controller gain values

PI controller			
Parameter	K_p	K_i	
Value	10.14	34.48	
PI^{λ} controller			
Parameter	K_p	K_i	λ
Value	11.89	29.31	0.817

The chromosomes for the PI controller parameters are [x1, x2] and for PI^{λ} controller are [x1,x2,x3]. Parameters of GA use values of the 'ga' function which are listed in table 3.1. When the performance needs to be improved, the parameters may be changed by '*gaoptimset'* function.

3.5 Concept of intelligent Control

3.5.1 Need for Intelligent Control

Intelligence is defined as the ability to comprehend, reason, and learn. An intelligent control system has the ability to comprehend, reason, and learn about processes, disturbances and operating conditions in order to optimize the performance of the process under consideration. Intelligent control techniques are generally classified as expert system control, fuzzy logic control, neural network control, and genetic algorithm. Intelligent induction motor control thus refers to the control of an induction motor drive using the above artificial intelligence techniques. The difficulties that arise in induction motor control can be classified under three categories. These are the following [26].

A. Complex computation: In induction motor control, application of conventional control theory and control algorithm often results in complex computations.

B. Nonlinearity: The presence of nonlinearities in an induction motor drive makes the control problem complicated. Current research efforts in nonlinear control theory focus on differential geometric methods and attempt to extend well known results in linear control theory to the nonlinear domain.

C. Uncertainty Certain essential information required in the mathematical model of the induction motor drive system, such as load, exact values of machine parameters, and noise, is unknown.

Recently, intelligent techniques like FL, ANN and ANFIS are introduced in order to overcome the above difficulties.

3.5.2 Fuzzy logic control

Fuzzy logic is one form of artificial intelligence. It is argued that human thinking does not always follow crisp 'yes no' logic, but is often vague, uncertain, indecisive, or fuzzy. The main characteristic of the fuzzy logic technique is to use the fuzzy rule sets and the linguistic representation of a human's knowledge to describe the controlled plant or to construct the fuzzy controller. A fuzzy logic controller consists of fuzzification, fuzzy inference with rulebase and database, and defuzzification [26].

The basic structure of a fuzzy logic control system is shown in figure 3.8 below [36, 37, 38, 39].

The fuzzification process

Fuzzification is the process of mapping inputs to the FLC in to fuzzy set membership values in the various input universes of discourse. Decisions need to be made regarding

(a) Number of inputs

(b) Size of universe of discourse

(c) Number and shape of fuzzy sets

The size of universe of discourse will depend upon the expected range (usually up to the saturation level) of the input variables. The number and shape of fuzzy sets in a particular universe of discourse is a tradeoff between precision of control action and real time computational complexity.

The purpose of this fuzzification step is to make the input physical signal compatible with the fuzzy control rule base in the core of the controller.

The fuzzy inference system

The basic structure of a fuzzy inference system consists of three conceptual components: a rule base, which contains a selection of fuzzy rules; a database (or dictionary), which defines the membership functions used in the fuzzy rules; and a reasoning mechanism, which performs the inference procedure upon the rules and given facts to derive a reasonable output or conclusion. The fuzzy rulebase consists of a set of antecedent-consequent linguistic rules. The rule base is constructed using a priori knowledge from either one or all of the following sources

a) Physical laws that govern the plant dynamics
b) Data from existing controllers
c) Imprecise heuristic knowledge obtained from experienced experts: it does not require the mathematical model of the system.

The role of the inference engine in the FLC is key to make the controller work and work effectively. The job of the "engine" is to create the control actions, in fuzzy terms, according to the information provided by the fuzzification module and the set point tracking requirement. The most important two types of fuzzy inference methods are **Mamdani** and **Sugeno or Takagi-Sugeno-Kang** method of fuzzy inference systems.

The Mamdani fuzzy inference system consists of a set of antecedent-consequent linguistic rules of the form; if x is A and y is B then z is C where A and B are fuzzy sets in the antecedent and C is fuzzy set in the consequent which needs defuzzification.

The Sugeno fuzzy inference system (also known as the TSK fuzzy inference system) was proposed by Takagi, Sugeno, and Kang in an effort to develop a systematic approach to generating fuzzy rules from a given input output data set. A typical fuzzy rule in a Sugeno fuzzy inference system has the form

if x is A and y is B then z = f (x, y)

Where A and B are fuzzy sets in the antecedent, while z = f (x, y) is a crisp function in the consequent. Usually f (x, y) is a polynomial in the input variables x and y, but it can be any function as long as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule.

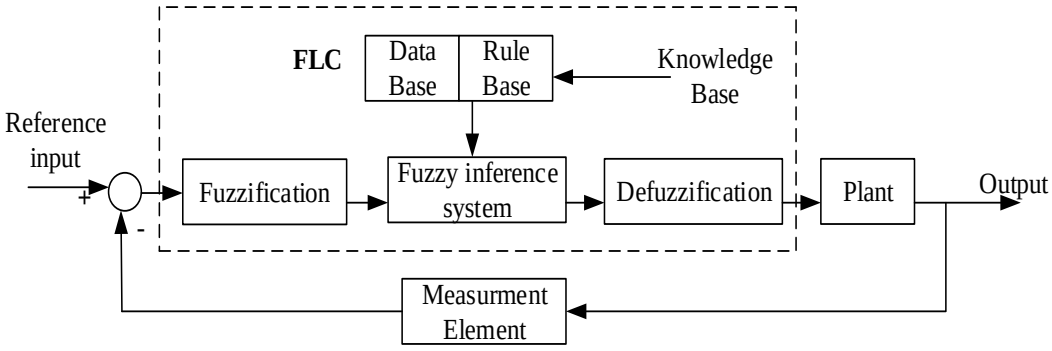


Figure 3.8 Fuzzy logic control system

Defuzzification

Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representative value. The defuzzification module is the connection between the control rule base and the physical plant to be controlled, which plays the role of a transformer mapping the controller outputs (generated by the control rule base in fuzzy terms) back to the crisp values that the plant can accept. Hence, in a sense the defuzzification module is the inverse of the fuzzification module. There are two common techniques for defuzzifying

1. **Center of mass**: This technique takes the output distribution and finds its center of mass to come up with one crisp number. This is computed as follows

$$z = \frac{\sum_{j=1}^{q} z_j \, u_c(z_j)}{\sum_{j=1}^{q} u_c(z_j)} \text{……...……..(3.25)}$$

$$\text{Crisp control signal} = \frac{\text{Sum of first moments of area}}{\text{Sum of areas}} \text{…….……(3.26)}$$

where z is the center of mass and uc is the membership in class c at value zj.

2. **Mean of maximum**:This technique takes the output distribution and finds its mean of maxima to come up with one crisp number. This is computed as follows:

$$z = \sum_{j=1}^{l} \frac{z_j}{l} \text{……...……...……...……. (3.27)}$$

where *z* is the mean of maximum, *zj* is the point at which the membership function is maximum, and *l* is the number of times the output distribution reaches the maximum level.

3.5.3 Artificial Neural Network control

Neural network is the most generic form of AI for emulation of human thinking compared to fuzzy logic. The neural network is famous for its learning ability and arbitrary approximation to any continuous function [26].

The Artificial Neural Network is a collection of simple processors connected together. Each processor can only perform a very straightforward mathematical task, but a large network of them has much greater capabilities and can do many things which one on its own cannot. Figure 3.9 shows the basic idea [40].

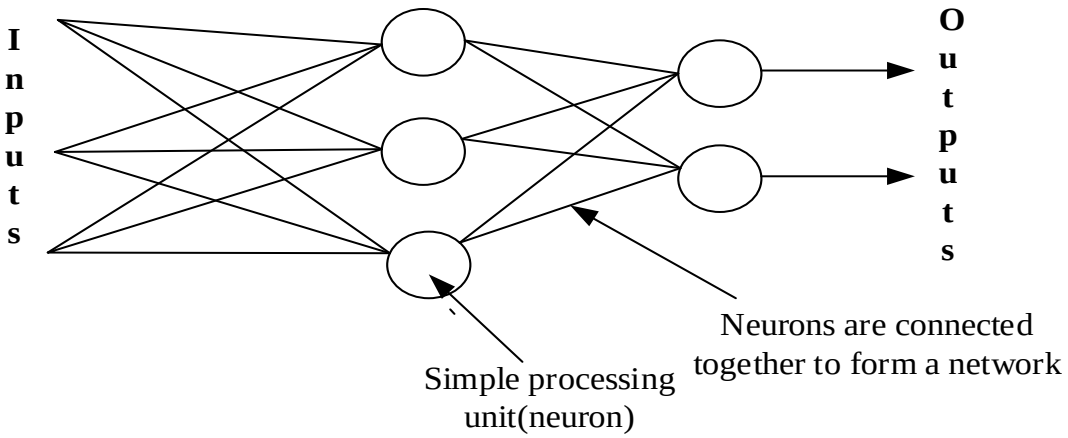


Figure 3.9 Neural network consists of many simple processing units connected together [40]

Artificial neural network has the following potential advantages for intelligent control [37, 40]

- ✓ They learn from experience rather than by programming.
- ✓ They have the ability to generalize from given training data to unseen control.
- ✓ They are fast, and can be implemented in real time.
- ✓ They fail gracefully rather than catastrophically.

The basic model of a single artificial neuron consists of a weighted summer and an activation (transfer function) as shown in figure 3.10.

Figure 3.10 shows that the inputs to the neuron are represented by *i* (in the biological sense, these are the activities of the other connecting neurons or of the outside world, transmitted through the dendrites). Each input is weighted by a factor which represents the strength of the synaptic connection of its dendrite, represented by *w*, the sum of these inputs and their weights is called the activity or activation of the neuron and is denoted as *S*. In mathematical terms:

$S = i_1 w_1 + i_2 w_2 + i_3 w_3 + i_4 w_4$ ……..(3.28)

So, the neuron takes its inputs, weights them according to how strong the connection is and if the total sum of the weighted inputs is above a certain threshold, the neuron "fires", just like the biological one. As explained later, such a neuron learns by changing its weights. Early Artificial Neural Networks used simple binary outputs in this way. However, it was found that a continuous output function was more flexible. Such as sigmoid function.

$O = \frac{1}{1+e^{-S}}$.....(3.29)

This function simply replaces the threshold Function and produces an output which is always between zero and one, it is therefore often called a Squashing or Activation Function. Other Activation Functions are also sometimes used, including: Linear, Logarithmic and Tangential Functions; however, the Sigmoid Function is the most common.

The preceding expression may be formalized for a neuron of *n* inputs

$S = i_1 w_1 + i_2 w_2 + i_3 w_3 + \ldots\ldots + i_n w_n$ ……..……...……..(3.30)

Which may be written conveniently as

$S = \sum_{x=1}^{x=n} w_x i_x$……..……. (3.31)

or, if the inputs are considered as forming a vector $\bar{I}$, and the weights a vector or matrix $\bar{W}$.

$S=\bar{I}.\bar{W}$……...…….(3.32)

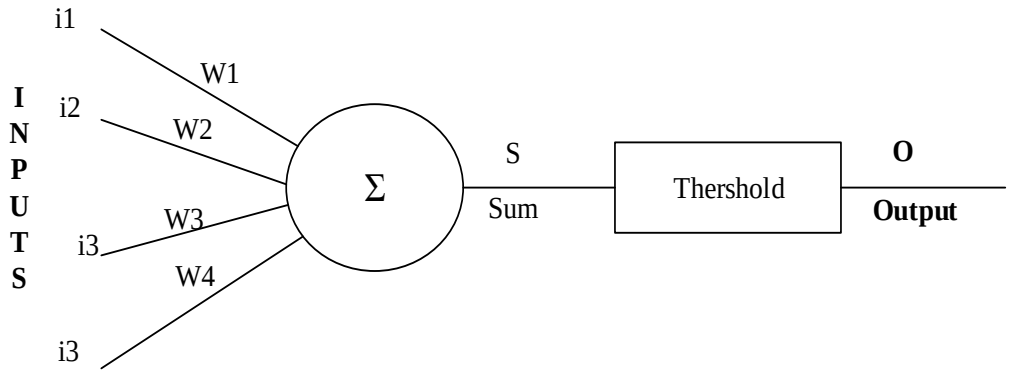


Figure 3.10 a basic Artificial Neuron [37]

A Neural Network consists of a number of these neurons connected together by the following network architectures.

Feedforward networks

ANN is a network of single neuron joined together by synaptic connections. Figure 3.11 shows a three-layer feedforward neural network. The feedforward network shown in figure 3.11 consists of a three neuron input layer, a two neuron output layer and a four neuron hidden layer. All neurons in a particular layer are fully connected to all neurons in the subsequent layer. This is generally called a fully connected multilayer network, and there is no restriction on the number of neurons in each layer, and no restriction on the number of hidden layers.

Feedback (recurrent) networks

This type of artificial neural network adds feedback connections to the network (the outputs are feedback into the inputs) and these connections are capable of interesting behaviors which we might not expect of them, in particular they can hold memories.

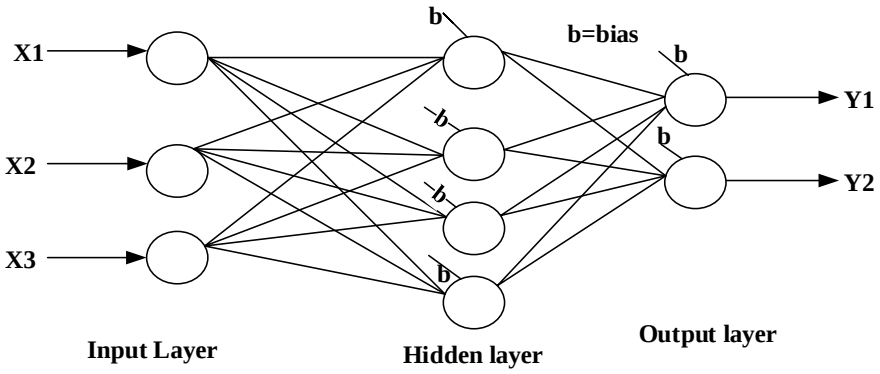


Figure 3.11 Three-layer feedforward neural network [37]

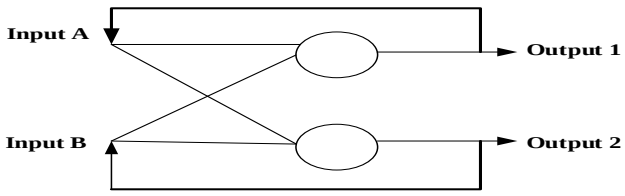


Figure 3.12 Feedback (recurrent) networks [40]

Learning in the context of a neural network is the process of adjusting the weights and biases in such a manner that for a given input, the correct response, or outputs are achieved. There are two types of learning. These are the following.

Supervised Learning: the network is presented with training data that represents the range of input possibilities, together with the associated desired outputs. The weights are adjusted until

the error between the actual and desired outputs meets some given minimum value. In this thesis Supervised Learning is used since input and output data of PI controller is available.

Unsupervised Learning: Also called open loop adaptation because the technique does not use feedback information to update the network's parameters. Applications for unsupervised learning include speech recognition and image compression.

Learning algorithms include backpropagation and hybrid learning Algorithm.

Backpropagation Learning Algorithm: The backpropagation algorithm is a supervised learning method for training ANN. It uses a gradient descent optimization method.

Hybrid learning Algorithm: It is combination of backpropagation algorithm and lest square estimator. Although we can apply backpropagation or steepest descent to identify the parameters in an adaptive network, this simple optimization method usually takes a long time before it converges. We may observe, however that an adaptive network's output is linear in some of the network's parameters; Thus we can identify these linear parameters by the linear least squares method. This approach leads to a hybrid learning rule which combines steepest descent and the least square estimator for fast identification of parameters. In this thesis hybrid learning algorithm is used.

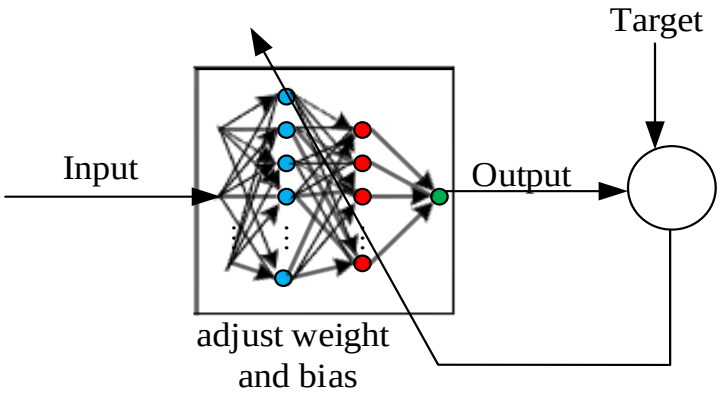


Figure 3.13 Training process of neural network [26]

Training of neural network continues until

- ✓ The performance index reaches an acceptable low value
- ✓ The maximum iteration count (number of epochs) has been exceeded
- ✓ The training time period has been exceeded.

3.5.4 Adaptive Neuro Fuzzy inference system

Neuro fuzzy control combines the mapping and learning ability of an artificial neural network with the linguistic and fuzzy inference advantages of fuzzy logic that have the ability to self-modify their membership function to achieve a desired performance. Thus, a neuro fuzzy controller has the potential to outperform conventional ANN or fuzzy logic controller. The adaptive network based fuzzy inference system (ANFIS) controller employs a Tagaki Sugno Kang (TSK) fuzzy inference system [37].

The basic ANFIS architecture is shown in figure 3.14. Square nodes in the ANFIS structure denote parameter sets of the membership functions of the TSK fuzzy system. Circular nodes are static (non modifiable) and perform operations such as product or max/min calculations. A hybrid learning rule is used to accelerate parameter adaptation. This uses sequential least squares in the forward pass to identify consequent parameters, and backward pass to establish the premise parameters [37].

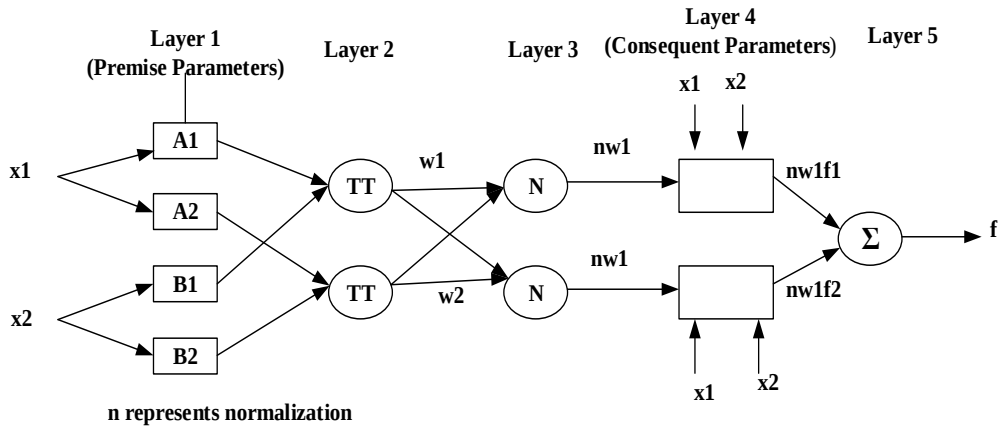


Figure 3.14 ANFIS architecture [37]

If the fuzzy inference system has inputs x1 and x2 and output f as shown in figure 3.14 then first order TSK rulebase becomes

Rule 1: if x_1 is A_1 and x_2 is B_1 then $f_1=p_1x_1+q_1x_2+r_1$

Rule 2: if x_1 is A_2 and x_2 is B_2 then $f_2=p_2x_1+q_2x_2+r_2$

For n membership functions

Rule n: if x_1 is A_n and x_2 is $B_n$then $f_n=p_nx_1+q_nx_2+r_n$

Where

$A_1 \ldots . A_n, B_1 \ldots . B_n$ membership functions and $p_1 \ldots . p_n, q_1 \ldots . q_n$ and $r_1 \ldots . r_n$ are constants within the consequent functions. The membership type can be any type of membership function which are available in fuzzy logic toolbox. For this thesis, typical membership function is generalized bell membership function which is given by:

$$\mu_A(x) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \text{……..……..……..……..(3.33)}$$

Where a, b and c are the parameter sets that are referred to as premise parameters which are optimized during training process.

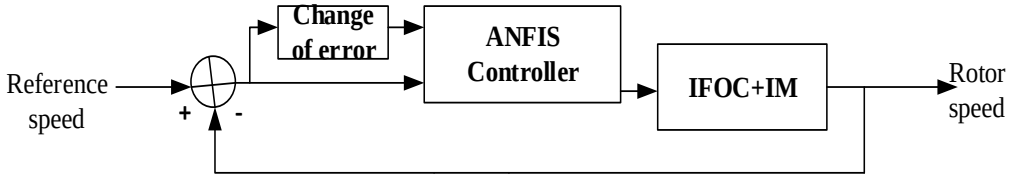


Figure 3.15 ANFIS control scheme for speed control of IFOIM

ANFIS has **five layers** with the following respective functions

Layer 1 Contains adaptive nodes that require suitable premise membership functions.

Each node generates the membership degree of linguistic terms that are given by the following equations. Where i is number of the nodes, and A and B are the linguistic terms.

$y_{1,i} = \mu_{Ai}(x_i)$ ……(3.34a)

$y_{1,i} = \mu_{Bi}(x_i)$……..……..……..……(3.34b)

Layer 2 Undertakes a T norm operation.

$y_{2,i} = w_i = \mu_{Ai}(x_1)\mu_{Bi}(x_2) \ldots\ldots\ldots \mu_{Pi}(x_n)$…….…….....… (3.35)

i=1, 2……, n

Layer 3 Calculates the ratio of the firing strength of the rules.

$y_{3,i} == \frac{w_i}{\sum_{i=1}^{n} w_i}$ …….……...…….....……. (3.36)

Layer 4 Generates the linear consequent functions.

$f_n = p_n x_1 + q_n x_2 + r_n$…….……..……..(3.37)

Layer 5 Sums all incoming signals.

$y_{5,i} = f = \sum_{i=1}^{n} \overline{w}_i f_i = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}$……..……..……..……..…… (3.38)

ANFIS is functionally equivalent to a Takagi Sugeno fuzzy inference system. By using stipulated input output training data pairs, ANFIS tunes the membership functions and other associated parameters by backpropagation gradient descent and least square type method. Such methodologies make the ANFIS modeling more systematic and less reliant on expert knowledge. Every node in layer one and layer four have an adaptive node with updated parameters in order to achieve a desired input output mapping. These parameters are updated according to given training data and gradient based learning procedure, while, the nodes in layers two and three are fixed with no parameters [41].

In learning process of ANFIS, the premise parameters (on layer one) and consequent parameters (on layer four) should be tuned to optimum mathematical relation between inputs and outputs. ANFIS uses a two-pass learning cycle. In the forward pass the algorithm uses least squares method to identify the consequent parameters. In the backward pass the errors are propagated backward and the premise parameters are updated by gradient descent. An initial fuzzy inference system is first created and improved through the learning process. ANFIS tunes the initial fuzzy inference system by a hybrid algorithm which, combining the gradient decent back propagation and least square optimization techniques. At each epoch the error is calculated. The stopping criteria is the same as neural network.

3.6 Design of Adaptive Neurofuzzy Inference System

3.6.1 Training of ANFIS controller using data from fractional order PI controller

ANFIS based modelling combines the transparent linguistic representation of fuzzy systems with the learning ability of neural networks so that, they can be trained to perform an input/output mapping. In designing, we cannot decide what the membership function must be, just by merely looking at the data. ANFIS permits the parameters to be automatically adjusted as a result; the membership functions capture the dynamics of data. The set of rules ANFIS provides is indicative of the underlying system and hence is valuable information to gain further insight into the process model. When FIS is launched as a controller, the special requirement is that the refining of parameters of membership functions should be done in such a way that the best performance of the plant is guaranteed.

The speed error and rate of change of speed error are inputs and command torque is output of neurofuzzy controller. Sugeno fuzzy model with five layer ANN structure is used in the proposed controller. The following figure 3.16 shows Input/output of ANFIS controller.

Figure 3.16 Input/output of ANFIS controller

Speed error=Input1=Reference speed −Actual (rotor) speed

Change of speed error = input2 = ΔSpeed error = Δ(Reference speed −Actual (rotor) speed)

Where, Δ indicates change.

In order to generate adaptive neurofuzzy system, one hundred fifty thousand (150,000) data have been considered. Out of these data, one hundred five thousand (70% of total data) have

been considered as training data set and remaining forty five thousand (30% of total data) have been considered as checking data. Since ANFIS involves the combined properties of neural network and fuzzy logic, it provides a method for the fuzzy modeling procedure to learn information about a data set, in order to compute the membership function parameters that allow the associated fuzzy inference system (FIS) to track the given input/output data. In order to develop the FIS structure, a network type structure similar to that of neural network which maps inputs through input membership function and associated parameters and through output membership functions and associated parameters to outputs can be considered.

The parameters associated with the membership functions will change through the learning process. The computation of these parameters is done by a hybrid learning algorithm. In this work, number of membership functions considered during the development of ANFIS is seven and type of input membership functions used are generalized bell functions. Linear membership function type is selected for output, totally 49 rules are generated, and the number of generations considered is 60. The plot of membership functions for inputs, surface and rule views are shown in B.6. of appendix B.

The performances of ANFIS models of both training and checking data were evaluated and the best training/checking data set was selected according to RMSE. In order to find the ideal ANFIS system, we trained the system with a variety of settings for items such as data set sample, epoch number, membership function type and number, and number of inputs to achieve the best performance. Using a given input/output data set, the MATLAB toolbox function *anfis* constructs a fuzzy inference system (FIS) whose membership function parameters are tuned using hybrid learning algorithm method.

The data were divided into two separate sets: the training data set and the checking data set. The training data set was used to train the ANFIS, whereas the checking data set was used to verify the accuracy and the effectiveness of the trained ANFIS model for the adaptation of learning content. Two pair data sets were made with different combinations of 70% and 30% of the samples to improve the generalization properties of the adopted ANFIS.

The training and checking data pairs were gathered from the simulation result of the block shown in figure 4.1 using fractional PI controller. After collecting the input/output data, the data divided in to training and checking data sets randomly. Once the input/output data pairs have been collected, the first step was loading training and checking data pairs to the anfis GUI of MATLAB toolbox, then initialize the FIS was the second step (the initialization step includes

the number and type of input and output membership functions and also number of epoch numbers). After initializing FIS and specifying the number of epochs the next step was train the training data pairs until the minimum training and checking errors are recorded. If the errors are acceptable the final step was testing the training data using checking data to validate the model. After loading the training data and checking data sets in Neurofuzzy Designer, the following results have been observed. Figure 3.17. shows the training and checking data sets after loading in ANFIS editor toolbox.

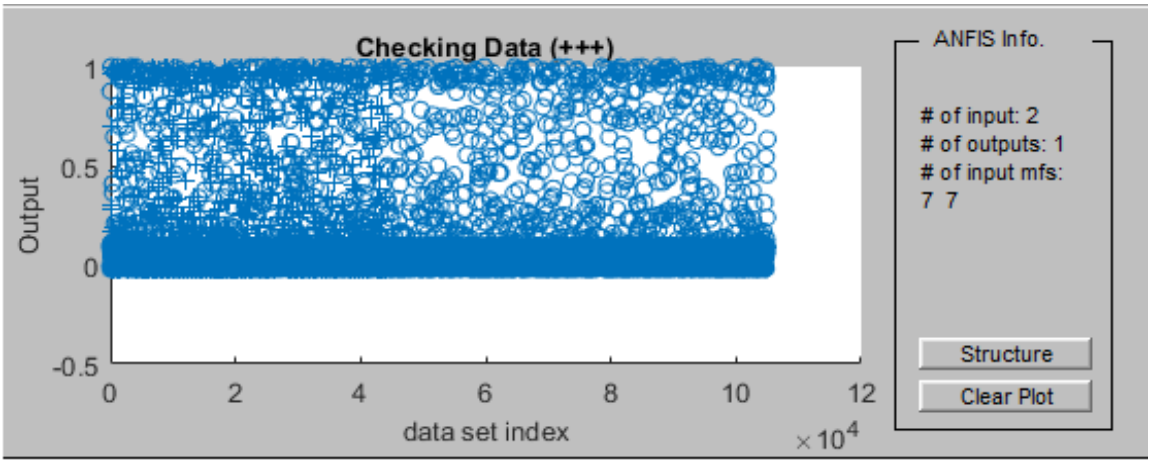
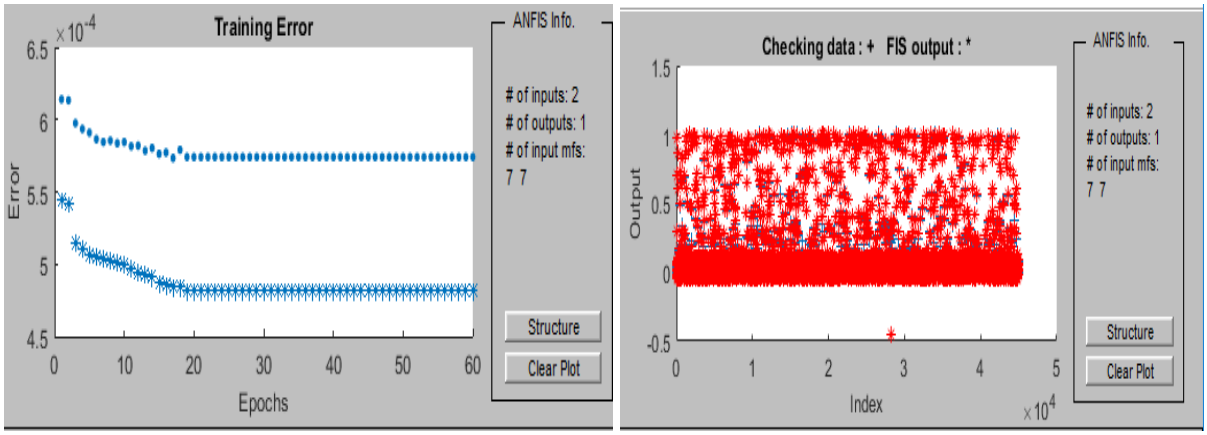


Figure 3.17 Training and checking data sets

Checking data is used for testing the generalization capability of the fuzzy inference system at each epoch. The checking data has the same format as that of the training data. This data set is used to validate the fuzzy inference model. This validation is done by applying the checking data to the model and then seeing how well the model responds to this data. When the checking data option is used using the ANFIS Editor GUI, the checking data is applied to the model at each training epoch. The FIS membership function parameters are computed using the ANFIS editor GUI when both training and checking data are loaded. The checking data is similar enough to the training data that the checking data error decreases as the training begins. The checking error is the difference between the checking data output value, and the output of the fuzzy inference system corresponding to the same checking data input value, which is the one associated with that checking data output value. The checking error records the RMSE for the checking data at each epoch. The ANFIS Editor GUI plots the checking error versus epochs curve as the system is trained. Minimal training RMSE is 0.000482 and minimal checking RMSE is 0.000570794.When checking data is tested against the trained FIS, it looks satisfactory and it shown in figure 3.18b.



(a) (b)

Figure 3.18 a) Training and checking errors b) Testing the FIS with checking data set

As shown in figure 3.18a the training and checking errors decrease as the training starts and became constant after the number of epoch reaches about 20. The RMS errors are negligibly small, and hence the ANFIS captured the essential dynamics of the system. Figure 3.18b shows how much the training and checking data are related and they are superimposed each other. The following figure shows the structure of the proposed ANFIS controller.

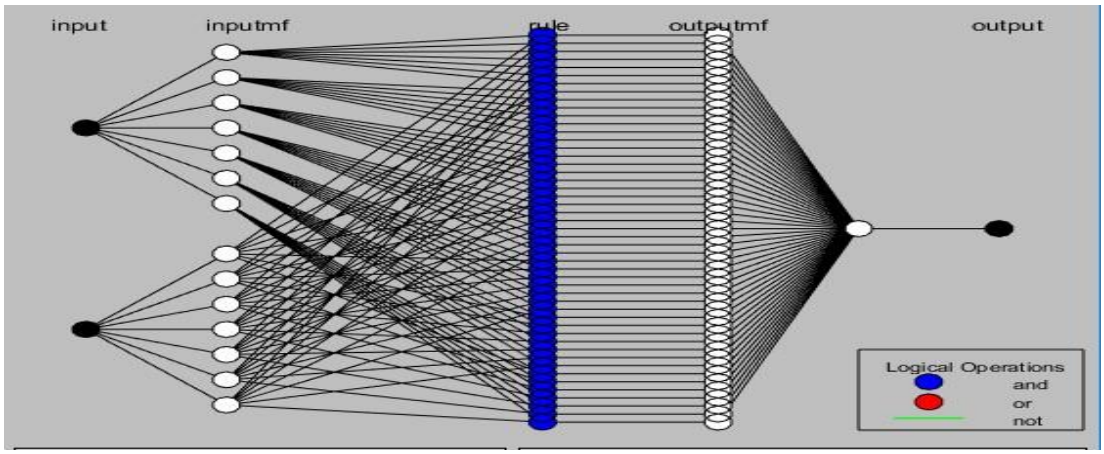


Figure 3.19 Schematic of the proposed ANFIS controller

From the ANFIS structure shown in figure 3.19, it has been observed that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters.

Table 3.3 Specifications of the developed adaptive neurofuzzy inference system

Parameters	**Description/Value**	**Parameters**	**Description/Value**
Optimization method	Hybrid	Number of outputs	1
Structure of FIS	Sugeno first order	Output MF type	linear
Or method	Max	No. of rules	49
And method	Min	No. of epochs /iterations	60
Implication method	Prod	No. of training data pairs	105,000
Aggregation method	Max	No. of checking data pairs	45,000
No. of inputs	2	No. linear parameters	147
No. of input MF	7	No. of nonlinear parameters	42
Type of input MF	Generalized bell	Total No. of parameters	189
No. of output MF	49	No. of nodes	131

CHAPTER FOUR

4. Simulation Studies and Analysis of Results

In this chapter analysis of fractional order PI, integer order PI and ANFIS speed controllers for indirect vector controlled induction motor that gives good performance for the overall drive system in terms of motor parameter variation, load variation, and reference speed variation will be discussed. Also comparison between the above mentioned controllers in terms of parameter, load torque, and reference speed variations will be discussed and analyzed. The analysis begins with simulation of induction motor without controller and validation of induction motor model. Then, analysis with controller will be follow.

4.1 Simulink Modeling

MATLAB/Simulink is a software package which is used to model, simulate, and analyze dynamic systems. Simulink has the advantage of being capable of complex dynamic system simulations, graphical environment with visual real time programming and broad selections of toolboxes. The mathematical equations presented in chapter two and three are used to model the three phase induction motor in Matlab/simulink R2017a environment. Figure 4.1 shows the complete simulink model of indirect vector speed control of three phase induction motor. The overall system simulink block includes different sub functional blocks such as IM model block, fractional or integer order PI controller blocks, coordinate transformation blocks, IFOC block, and SVPWM inverter block. Once the block diagram has been developed in MATLAB/simulink it can be simulated using any number of different solvers. These solvers can compute the internal state variables of the blocks by solving their respective ordinary differential equations in MATLAB modeling configuration parameters. The solver is significant to decrease the computation time and improve the accuracy of the simulation.

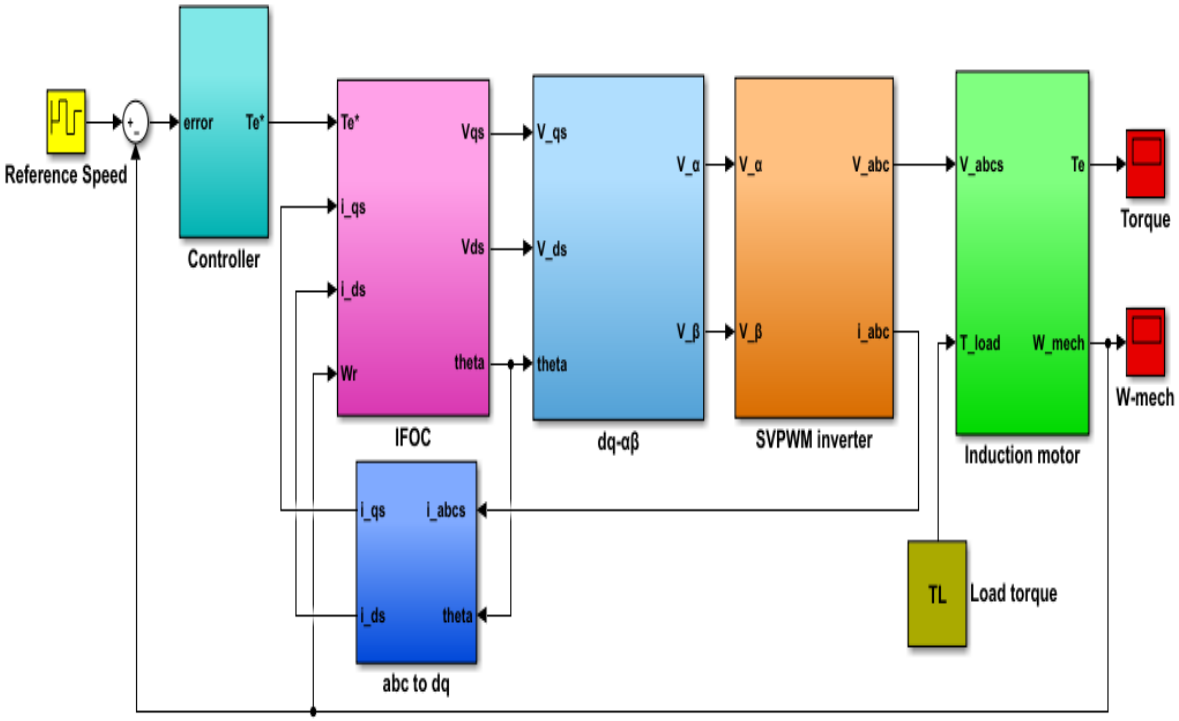


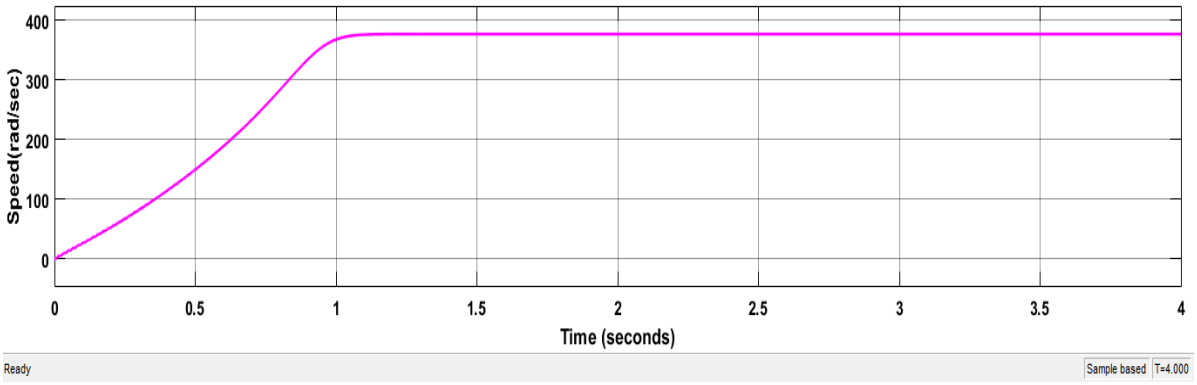
Figure 4.1 Complete simulink model of indirect vector control of three phase IM

Space vector PWM inverter is taken from MATLAB simulink library. It comprises PWM generator, IGBT voltage source inverter and DC voltage source. Space Vector Pulse Width Modulation (SVPWM) method is one of the advanced, computation intensive PWM method and possibly the best among all the PWM techniques.

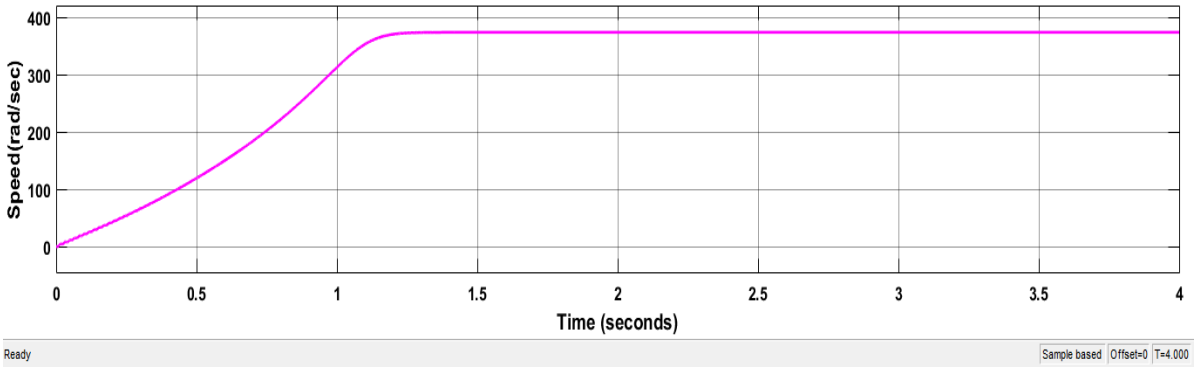
4.2 Response of induction motor without controller

Before proceeding to the controlled induction motor drive, it is important to see the uncontrolled ones. To analyze the performance of IFOC induction motor drive, a Simulink model in appendix A of figure A.1 is used. The parameters of the induction motor used for simulation is listed under motor1 in table C.1 of appendix C. The following figures(4.2a-4.4b) show the speed, torque and speed-torque characteristics simulation results of the selected induction motor without load and with load of 20Nm.

As shown in figure 4.2a the motor rotates at 376.94 rad/sec which is near synchronous speed of 377 rad/sec(3600rpm). Since the motor runs without load the rotor speed is near synchronous speed therefore, the result is acceptable. Figure 4.2b shows the speed response of IM with 20Nm load torque; the result shows 371rad/sec rotor speed which is 6rad/sec less than the synchronous speed as a result of load.



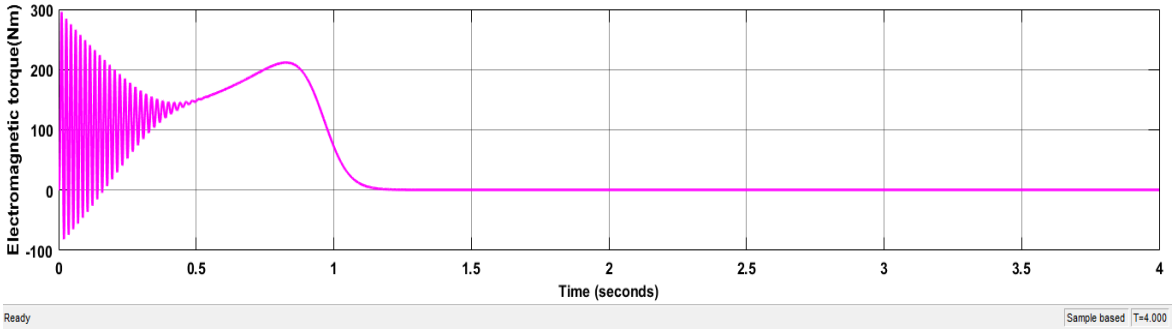
(a)



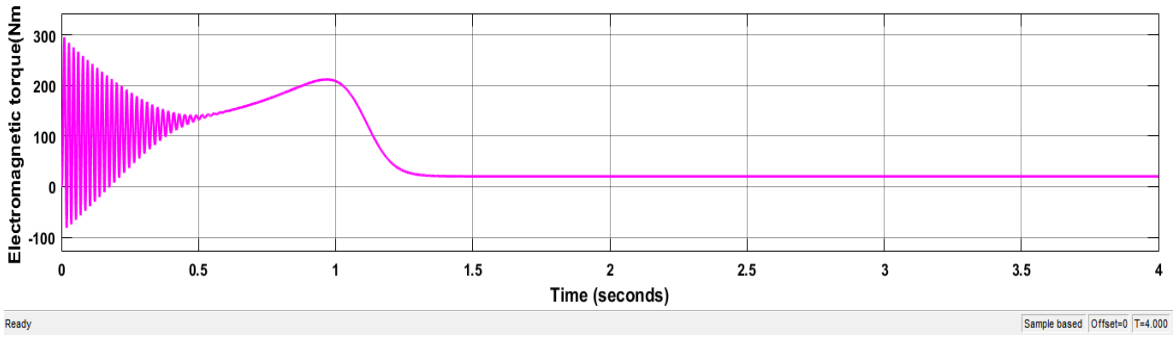
(b)

Figure 4.2 Speed response a) at no load, b) at 20Nm load torque

Figure 4.3a shows electromagnetic torque at no load. Initially, it oscillates at some value and finally it settles to 0.3769Nm which is near to zero. As shown in figure 4.3b the electromagnetic torque finally settles to 20Nm which is due to 20Nm load torque. From the principle of operation of induction motor we know that if rotor runs at the synchronous speed, then it will appear stationary to the rotating magnetic field and the rotating magnetic field will not cut the rotor. As a result, no induced current will flow in the rotor and no rotor magnetic flux will be produced, due to this no torque is generated. Using this principle, the above results are acceptable.



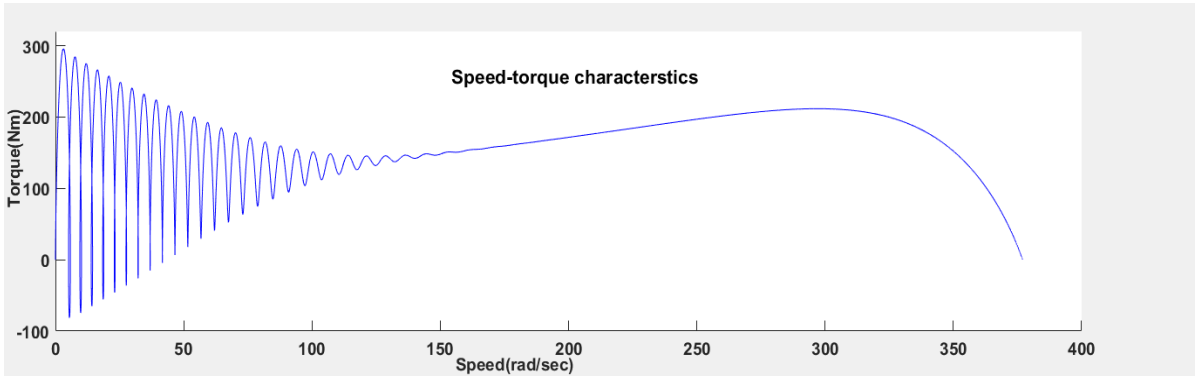
(a)



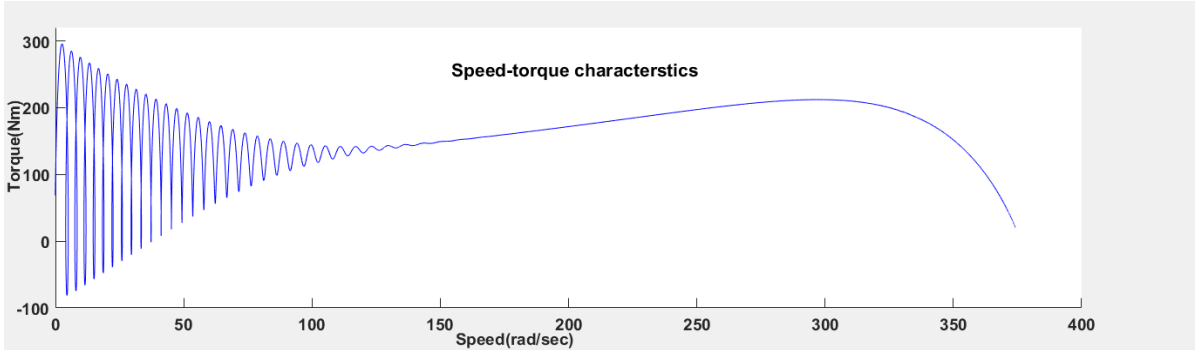
(b)

Figure 4.3 Electromagnetic torque response a) at no load , b) 20Nm load torque

As shown in figure 4.4a the speed torque graph of induction motor as the speed reaches to rated value, the electromagnetic torque becomes zero because the motor is at no load condition. On the other hand, in figure 4.4b as the speed reaches to its rated value, the electromagnetic torque reaches to 20Nm this is due to load torque of 20Nm.The above results also agree with the general principle of induction motor.



(a)



(b)

Figure 4.4 Speed-torque characteristics a) at no load , b) at 20Nm load

All the above results show that induction motor runs at constant speed near to synchronous speed at no load condition and decreases as the load is applied. To overcome this problem designing a controller is a must. In the next portion the controllers that were designed in chapter three will be applied to run the induction motor to the required speed.

4.3 Comparison of fractional order PI, integer order PI and ANFIS speed controllers

MATLAB/Simulink simulations will be performing on fractional PI and integer order PI controllers (both are GA optimized) and ANFIS controller. Motor1 is selected as a test motor whose parameters are given in table C.1 of appendix C. Four investigations will be undertake on (1) Response of IM using the three controllers at no load, (2) Reference tracking capability of controllers (3) effects of motor parameter variation and (3) effects of load torque variation.

4.3.1 Responses of IM using FOPI, IOPI and ANFIS speed controllers at no load

In order to compare the GA optimized fractional and integer order PI controllers and ANFIS speed controller (trained by input/output of GA optimized FOPI controller) MATLAB simulation results of the test motor will be analyzed. The simulation results are carried out at 50rad/sec reference speed. The controllers are evaluated based on performance criteria such as rise time, settling time, steady state error and peak overshoot.

Figure 4.5a shows the speed versus time response of FOPI, IOPI and ANFIS speed controlled IM drive at no load condition. It is summarized in table 4.1. As shown in table 4.1 the GA optimized PI control has bigger overshoot, rise time and steady state error compared to GA optimized fractional PI controller. but settling time of FOPI controller is slightly bigger than IOPI by 0.002 sec. which is almost negligible. The IOPI speed controller shows higher overshoot. This higher overshoot is reduced by using FOPI controller and with the use of ANFIS, the speed overshoot is highly reduced as shown in figure 4.5. The ANFIS controller shows a faster response in comparison with FOPI and IOPI speed controllers. ANFIS controller outperforms both FOPI and IOPI controllers in all performance criteria.

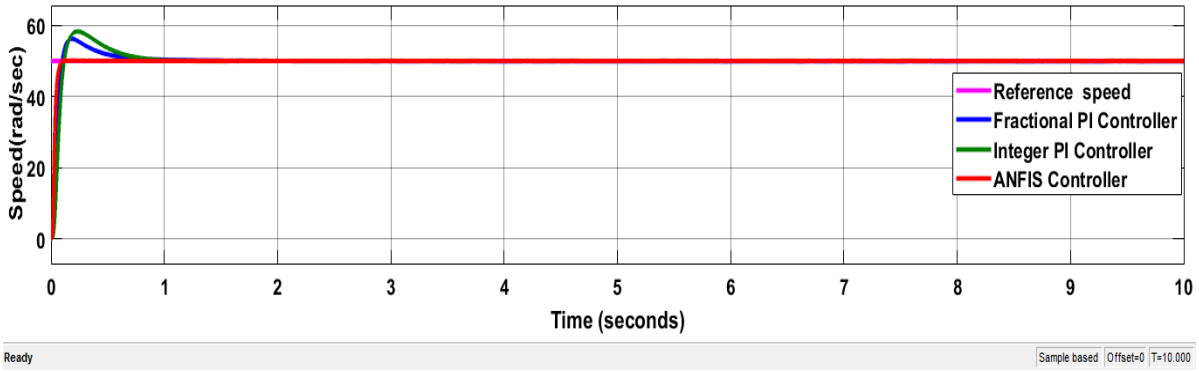


Figure 4.5 Speed response of FOPI, IOPI and ANFIS control system at no load

ANFIS tries to speed up the performance of the IM drive. Further, it can also be observed that using the ANFIS control, the system stabilizes in a very less time compared to the IOPI and FOPI controllers because of the training process of the ANN involved and the proper selection of the rule. As observed from figure 4.6 IOPI controller have higher error initially than FOPI and ANFIS controllers. Also FOPI still have higher error than ANFIS controller. ANFIS have very small overshoot and less steady state error, these show the effectiveness of this controller.

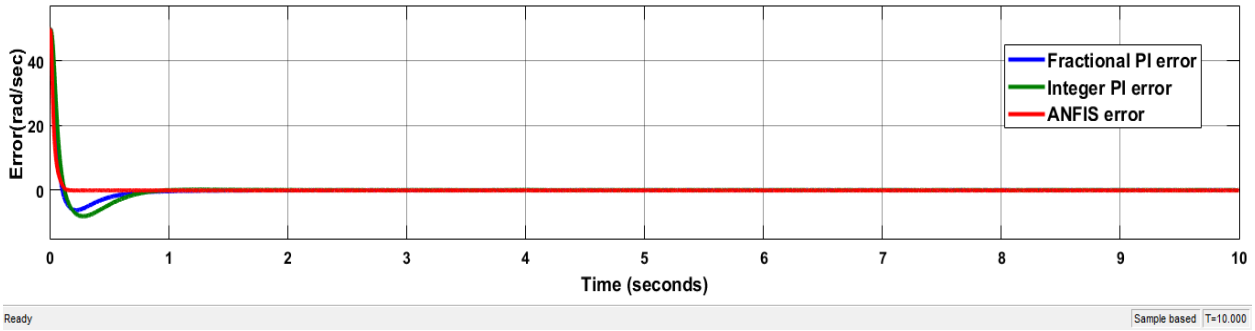


Figure 4.6 Speed error of FOPI, IOPI and ANFIS control system at no load

Figure 4.7 shows stator current response of IM drive without load. The initial peak current values decrease in the order of ANFIS, FOPI, and IOPI controllers. That is ANFIS have high initial current but, as initial current increases settling time decreases as seen in figure 4.7. Since the peak values are high for a fraction of microseconds, it doesn't affect the motor operation. From the simulation result i_{ds} is constant only i_{qs} have peak value. As explained in chapter three the strategies for both direct and indirect field oriented control method to achieve the condition that the rotor flux and rotor current vectors are always perpendicular are insure $\lambda'_{qr} = 0$ and $i'_{dr} = 0$. The first strategy is satisfied by choosing the position of the synchronous reference frame to put all of the rotor flux linkage in the d axis and the second strategy can be accomplished by forcing the d axis stator current to remain constant. That is why the d axis current is constant. The electromagnetic torque is controlled by quadrature axis

stator current and flux is controlled by direct axis stator current. The rotor flux is held at its nominal value of 0.986 weber.

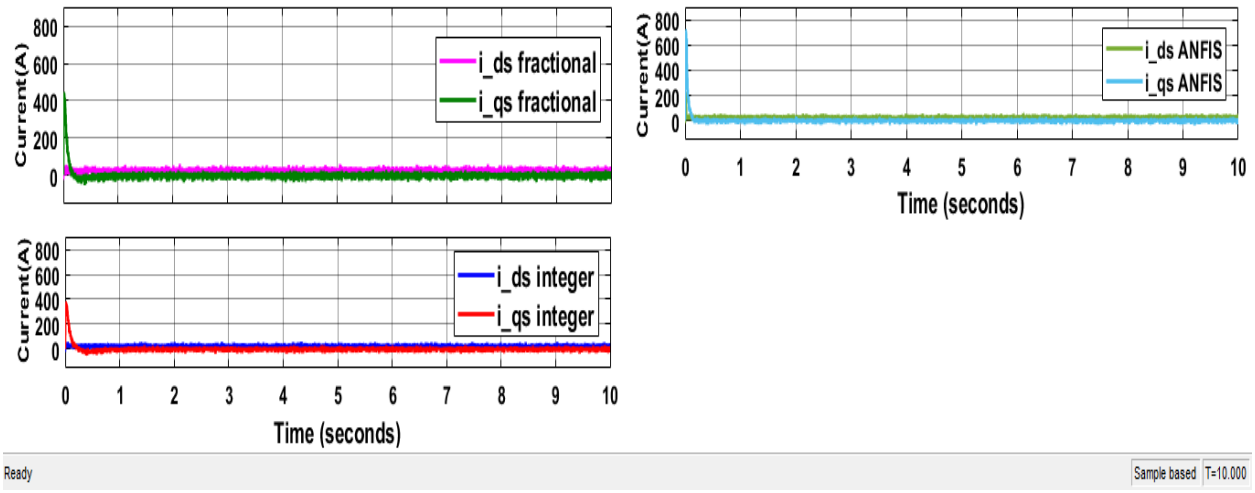


Figure 4.7 Stator current response of FOPI, IOPI and ANFIS control system at no load

Electromagnetic torque response in figure 4.8 shows the same form as quadrature axis stator current since, i_{qs} is a torque producing component of stator current. The electromagnetic torque of FOPI controller settles in less time than IOPI controller but, ANFIS controller settles in less time than FOPI controller. Once the machine reaches the set speed of 50 rad/s, the average torque of the machine becomes nearly zero. Even though the peak electromagnetic torque at initial is high in FOPI controller, it oscillates around zero fast compared to IOPI controller.

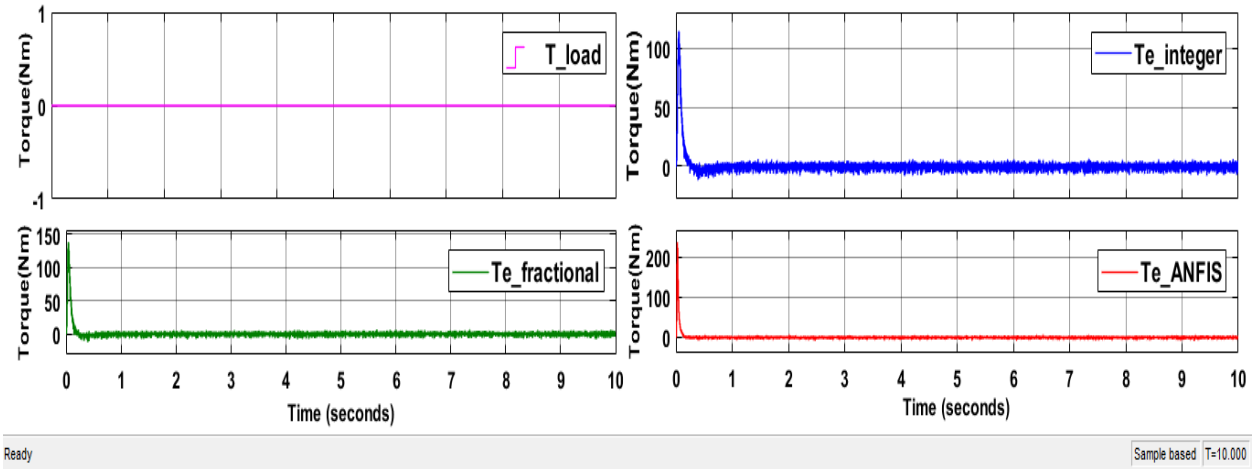


Figure 4.8 Torque response of FOPI,IOPI and ANFIS control system at no load

Table 4.1 Performance of FOPI, IOPI, ANFIS controllers at no Load

Performance criteria	ANFIS	Fractional order PI	Integer order PI
Rise time(sec)	0.058764	0.060919	0.074766
Settling time(sec)	0.15	0.962	0.96
Steady state error (rad/sec)	0.01	0.015	0.02
Overshoot(%)	0.496	13.068	15.698

In general, results show that FOPI has improved system performance, regarding system error, overshoot and rising time compared with IOPI controller. Electromagnetic torque, rotor speed and stator phase currents of the system have good transient and steady state response. ANFIS controller outperforms the two controllers in all performance criteria. The above figures show these properties.

4.3.2 Tracking capability of controllers for reference speed variation

The simulation results are carried out for different reference speeds at no load condition. The considered reference speeds are below and above 50rad/sec which was used as a reference speed to find the GA optimized FOPI and IOPI controller parameters to see the performance of controllers at low and high speed. Reference speeds that are taken for simulation are 0,30,20,40,70 in rad/sec applied sequentially at 2sec time interval. Figure 4.9 shows that the ANFIS controller has the ability to follow the speed of the induction motor drive effectively as compared with FOPI and IOPI controllers. The result shows that the speed using ANFIS is better in overshoot, settling time, rise time and steady state error as before. Also, FOPI controller has better tracking capability than IOPI controller. For 0 to 2sec time interval the reference speed is 0 rad/sec, the controllers can track this reference speed effectively this shows better capability of controllers at lower speed. The figure also shows that we can operate the motor in wide speed range.

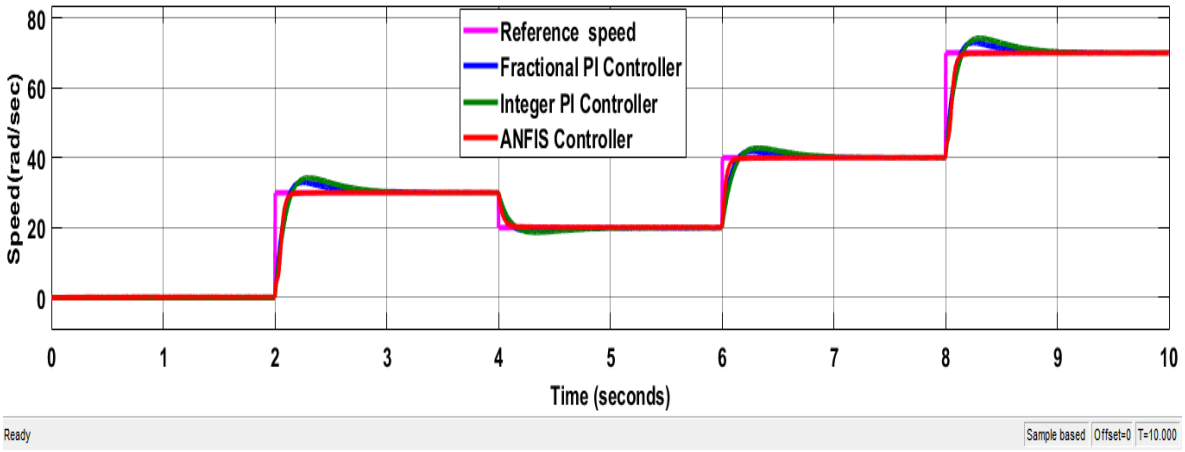


Figure 4.9 Speed tracking at no load with reference speed variation

Figure 4.10 shows the speed error response of FOPI, IOPI and ANFIS controllers. As the figure shows speed error using ANFIS controller settles to its minimum value faster compared to other controllers. FOPI controller has less error compared to IOPI controller.

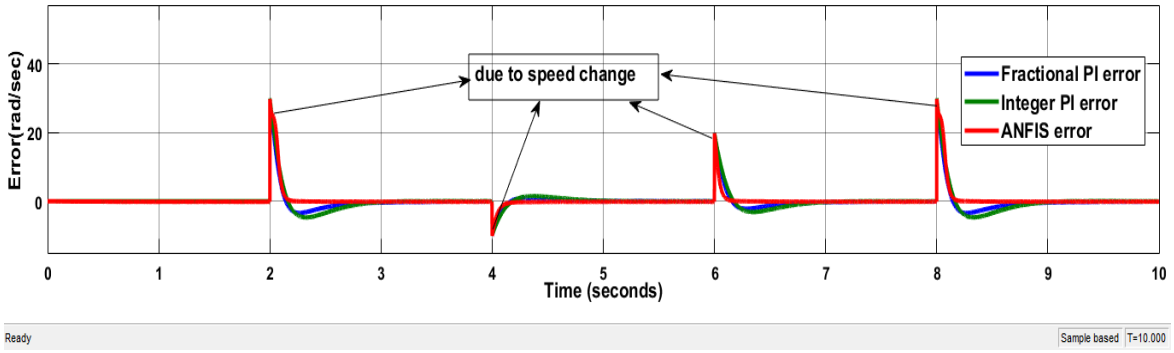


Figure 4.10 Speed error at no load with reference speed variation

Figure 4.11 shows the stator current responses of FOPI, IOPI and ANFIS controllers for different reference speed. The direct axis current does not affect by reference speed variation only the quadrature axis current is affected. As explained in previous section the quadrature axis current is a torque producing component of stator current. Therefore, it is affected by command torque which is the output of the controller. The command torque in turn affected by reference speed. On the other hand, the direct axis current is flux producing component which kept at its nominal value. As a result, direct axis current is constant. ANFIS has large peak quadrature axis current than FOPI and IOPI controllers but, it settles faster. So, it does not affect operation of the drive.

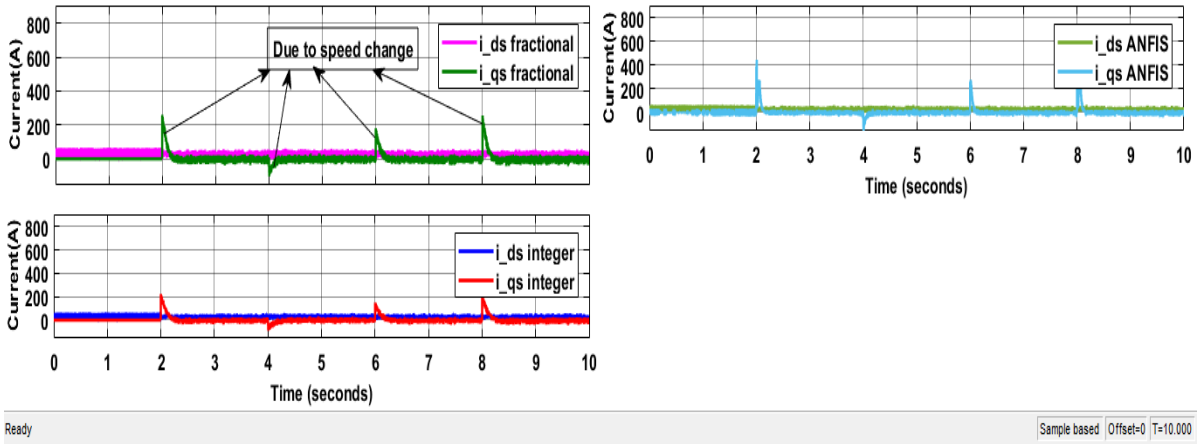


Figure 4.11 Stator currents at no load with reference speed variation

As figure 4.12 shows the electromagnetic torque responses of IM drive using IOPI, FOPI and ANFIS controllers have the same form as quadrature axis stator current response. The torque response of controllers settles fast near to zero which indicates the motor running at no load condition.

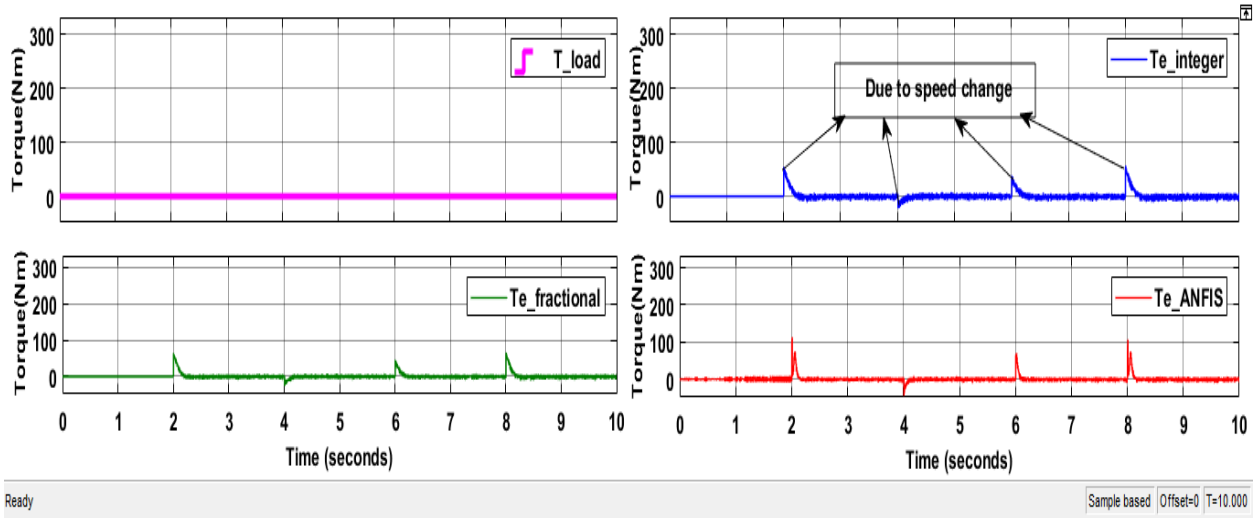


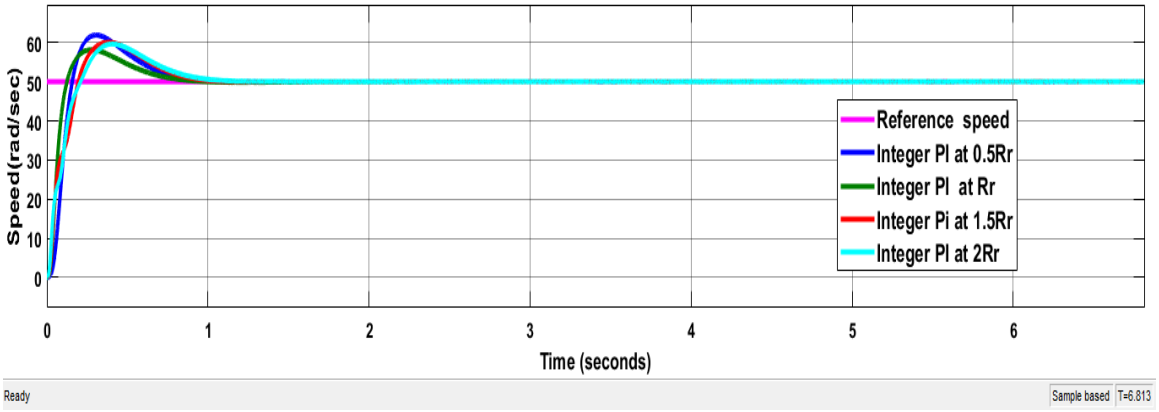
Figure 4.12 Electromagnetic torque at no load with reference speed variation

4.3.3 Effect of motor parameter variation

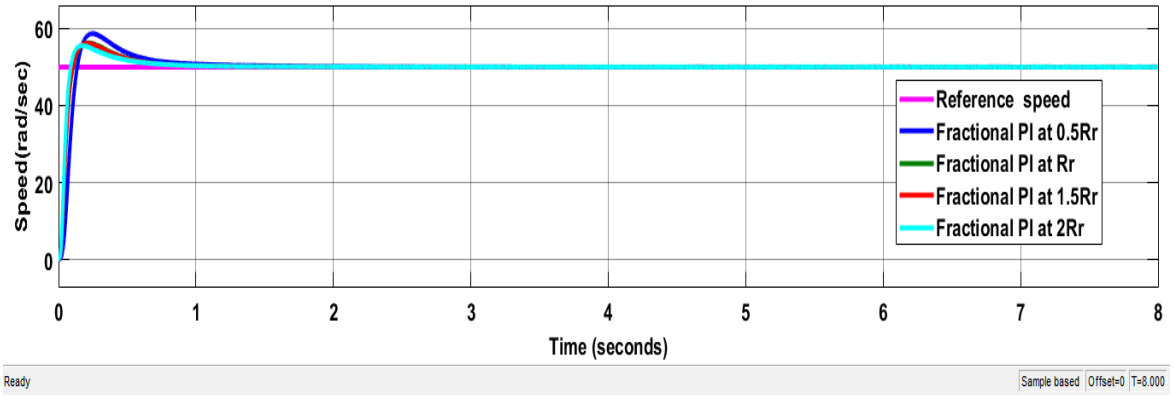
The control of induction motor for high performance is a challenging problem because they exhibit significant non linearity and it is known that uncertainties of motor parameters and unknown external disturbances can degrade the performance of the drive. To see the effect of motor parameter variation on the performance of IM drive, assume that the parameters such as inductance and pole are remains constant, and it is evident that the speed response of the system is not affected by the variation. The resistance of IM changes because of temperature variation caused by machine losses. As the resistance value varies by temperature, the performance of controllers can be deteriorating. The performance of decoupling control methods based on machine models can be influenced by a mismatch between the parameters value being used in the controller and the actual machine parameters. Let's see the variation in resistance of stator and rotor windings based on the following different cases.

- ✓ Stator resistance variation (50% decrease,50% and 100% increase)
- ✓ Rotor resistance variation (50 decrease,50% and100% increase)

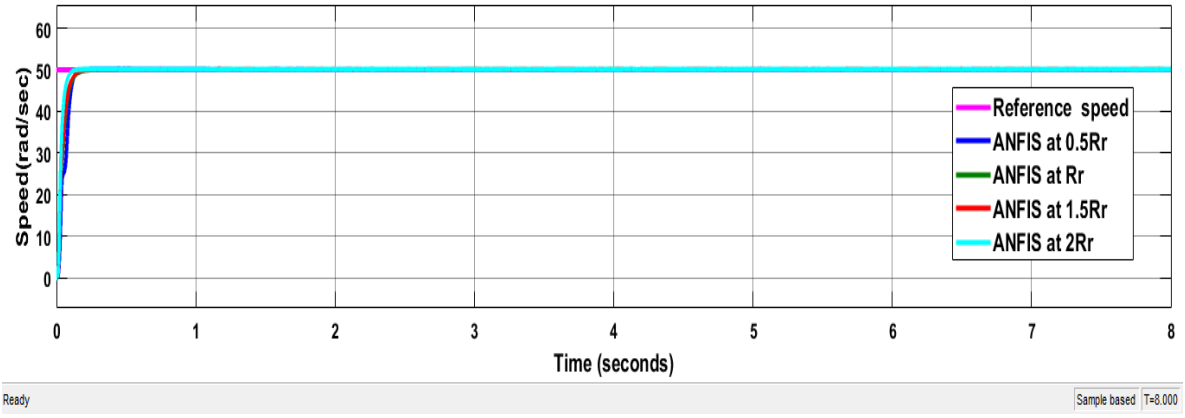
Figure 4.13(a-c) below show the speed responses of IM drive due to rotor resistance variations without load at reference speed of 50rad/sec. Figure 4.13a shows the speed response using IOPI controller while figure 4.13b and 4.13c show speed responses of IM drive using FOPI and ANFIS controllers respectively. As the resistance increases from 0.5Rr (-50% of Rr) to 2Rr(+100%Rr), there are changes in rise time, settling time, steady state error and peak overshoot of IM drive using all controllers. But the amount of changes are varying from controller to controller. Let take 1.5Rr (50% increase) to see how the controllers respond and how much the performance criteria are changed numerically. The performance criteria at Rr are listed in table 4.1. The change in rising times are 0.072167 sec,0.00185 sec and 0.00593 sec using IOPI, FOPI and ANFIS respectively. This shows IOPI controller has high rising time change compared to FOPI and ANFIS controllers. FOPI controller exhibits lower rising time than IOPI and ANFIS controllers. When we come to percentage overshot using Rr and 1.5Rr; 2.821% using IOPI controller,0% using FOPI controller, and using ANFIS controller 0.031% changes are recorded. All in all, the simulation results show FOPI and ANFIS controllers are less sensitive to rotor resistance variation while IOPI controller is more affected by rotor resistance variation.



(a)



(b)

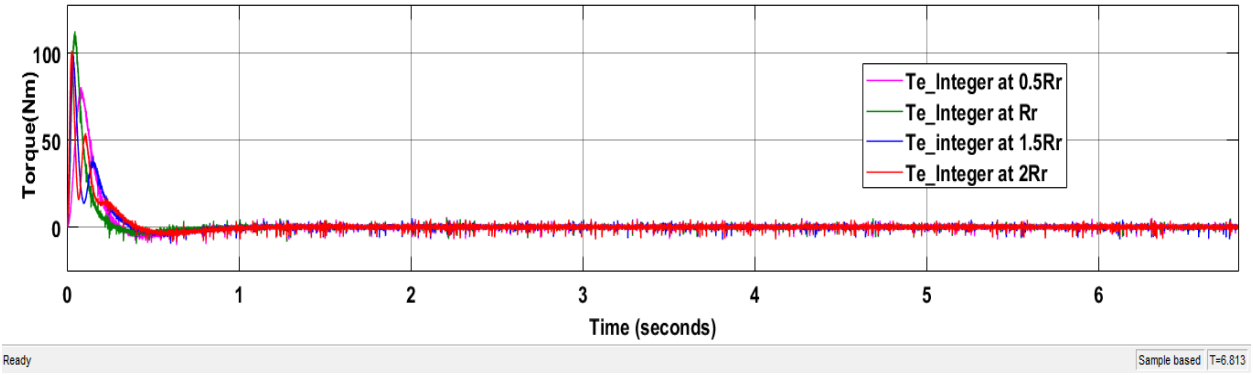


(c)

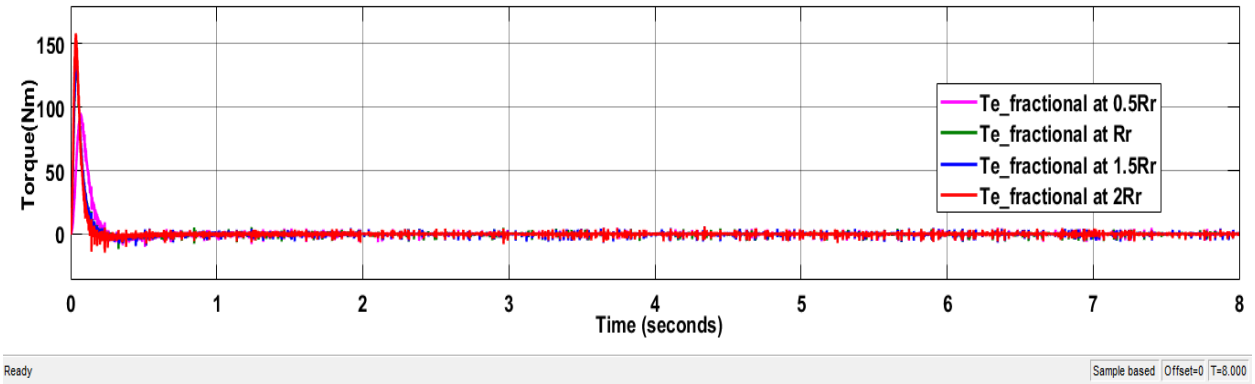
Figure 4.13 Speed response at no load with Rr variation using a) IOPI, b) FOPI, c)ANFIS

Figure 4.14(a-c) show the electromagnetic torque response of IOPI, FOPI, and ANFIS controllers at different values of rotor resistance without load. As shown in figure 4.14a the electromagnetic torque results of IOPI controller with Rr has high peak value compared with 0.5Rr, 1.5Rr and 2Rr but, it settles fast compared to others. From figure 4.14b, the

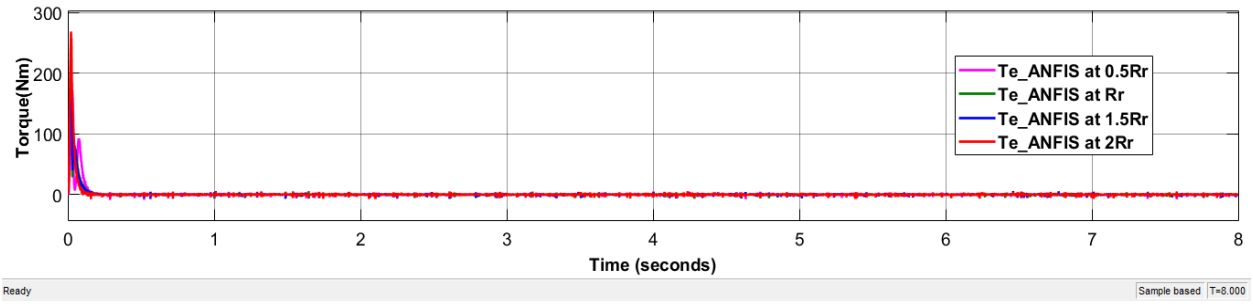
electromagnetic torque response using FOPI controller 2Rr has high peak value and settles faster than other electromagnetic torque response with 0.5Rr, Rr and 1. 5Rr.The same is true for ANFIS controller as FOPI controller as shown in figure 4.14c. The transient electromagnetic torque oscillation increases in the order of ANFIS, FOPI, and IOPI controllers. The torque with ANFIS is better in settling time, rising time, and has low oscillation peak compared to those of IOPI and FOPI controllers even if the rotor resistance is varying. Here, it is seen that the performance of the motor is almost similar in both cases of constant rotor resistance as in steady state conditions, as well as a 50% decrease,50% increase and 100% increase in rotor resistance that corresponds to the dynamic conditions of motor operation, which leads to parameter variations. The only downside of control strategy using IOPI and FOPI controllers is, the peak overshoot that occurs at the very start of the operation and high settling time. Such a sharp change in the motor speed at the start can lead to hazardous situations where the motor windings can burn, however, under ANFIS controller not only the peak overshoot is reduced near to zero, but the amount of oscillations under steady state are reduced, thus giving promising results than IOPI and FOPI controllers.



(a)



(b)



(c)

Figure 4.14 Electromagnetic torque response at no load with Rr variation using a) IOPI, b) FOPI, c) ANFIS

Figure 4.15 shows speed response of IM drive using FOPI, IOPI, and ANFIS controllers at no load with stator resistance variation. As used in rotor resistance variation; in this case also 50% decrease,50% increase and 100% increase in Rs is considered. From the simulation result stator resistance has no considerable effect on the performance of the drive using IOPI, FOPI, and ANFIS controllers.

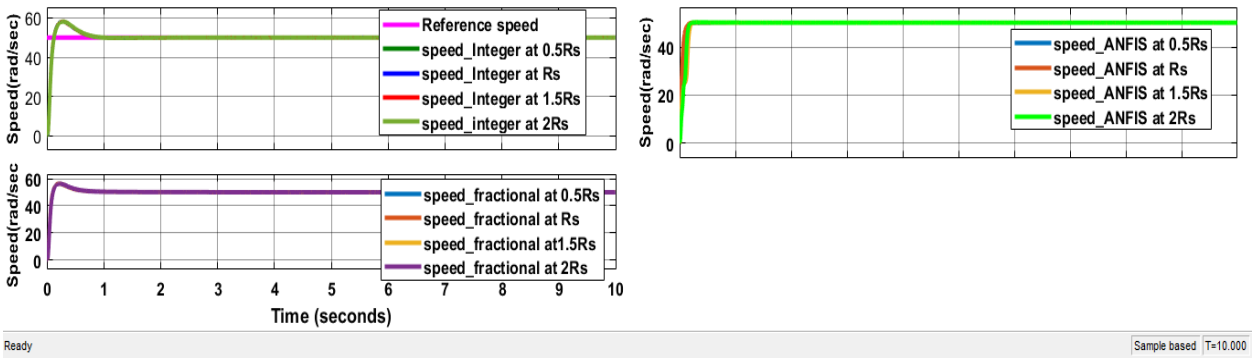


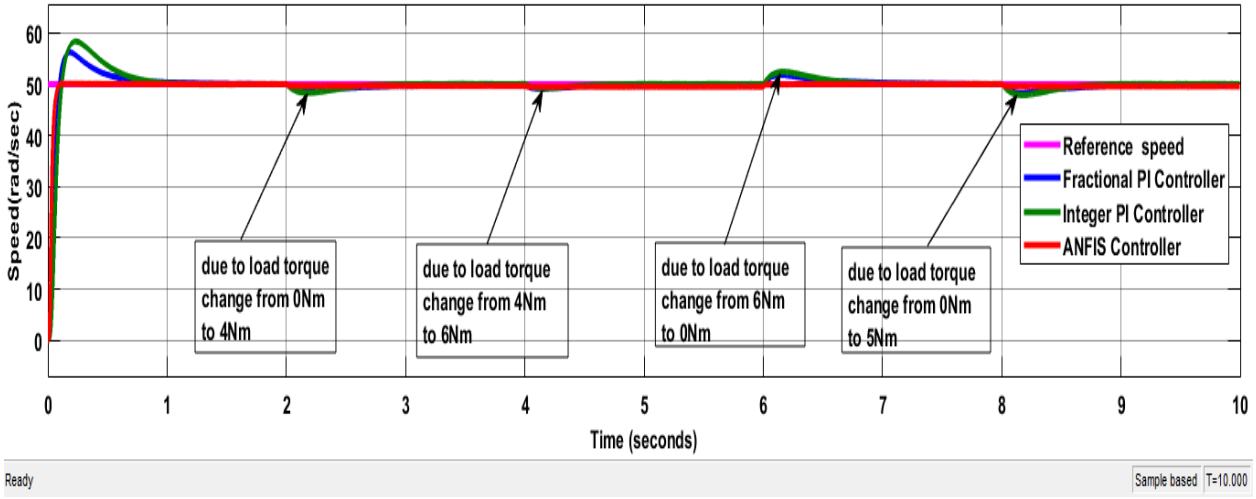
Figure 4.15 Speed response using FOPI, IOPI, and ANFIS at no load with Rs variation

4.3.4 Effect of load torque variation

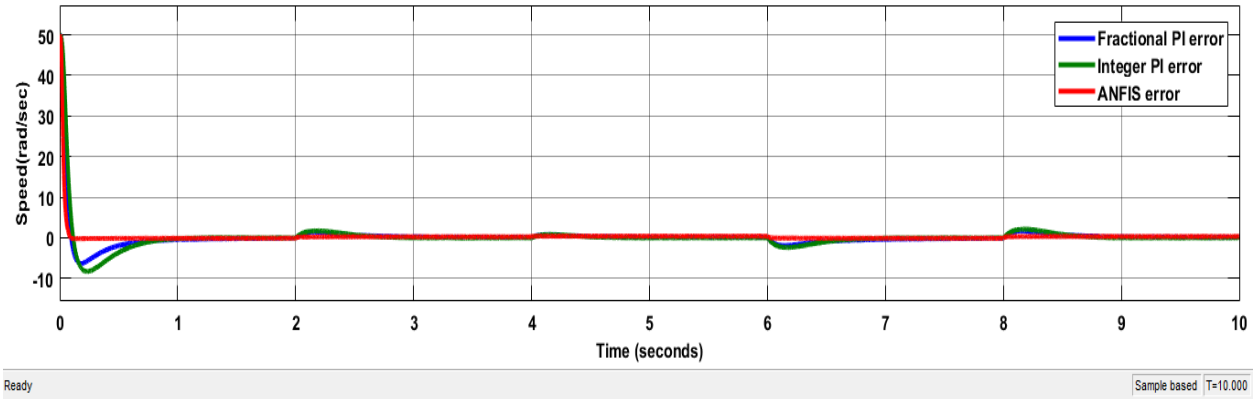
The simulation will investigate effects of load torque variation on IOPI, FOPI, and ANFIS control of IM drive. The load torque variation can be implemented by changing parameters of the repeating sequence stair parameter block in the Simulink diagram. The selected motor has rated torque of 6Nm. In order to testify the robustness of the controlled system, variable torque values of 0Nm (no load),4Nm, 5Nm, and 6Nm (full load) at 2seconds interval with 50rad/sec reference speed for total simulation time of 10seconds are considered. In the simulation, IM drive experiences sudden changes in the load torque at t=2sec, the load increases from 0Nm to 4Nm, at t=4 second, the load increases to 6Nm, and at t=6second, the load decreases to 0Nm, at t=8second, the load increases again to 5Nm.

Figure 4.16a shows the rotor speed of IOPI, FOPI and ANFIS control system with load torque variation. As inferred from the figure ANFIS control system is less sensitive to load torque variations than IOPI and FOPI control system. Also, FOPI control system is less sensitive to load torque variations as compared to IOPI control system. Figures 4.16b, 4.17(a-b) show the speed error, stator currents, and torque response with load torque variation respectively.

Figure 4.16b gives the speed error response when induction motor is commanded to follow the 50rad/sec reference speed with sudden change in torque load. The IOPI controller has high speed error at each load torque change instants. Therefore, it is true to say that the ANFIS controller is robust to load torque variations.



(a)

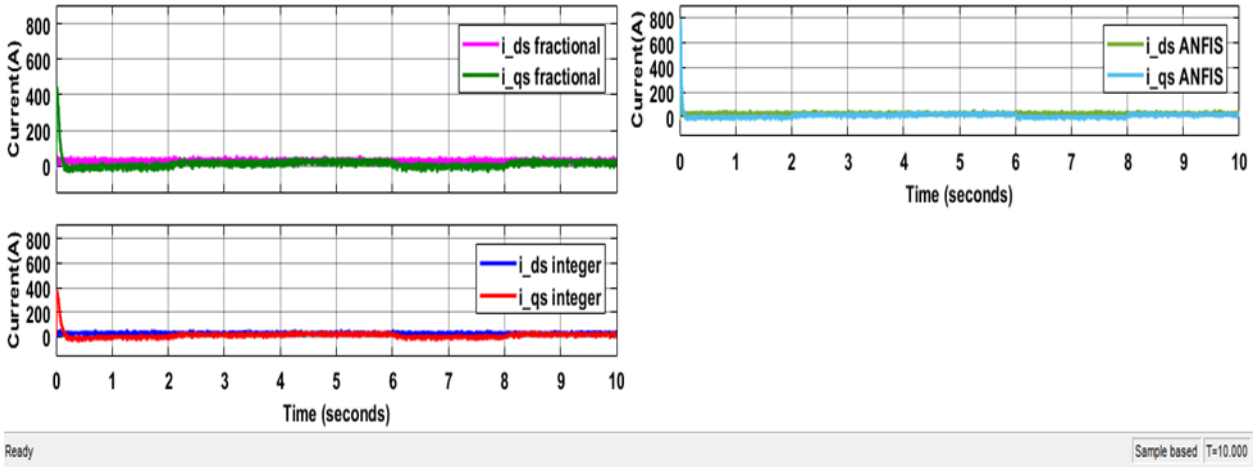


(b)

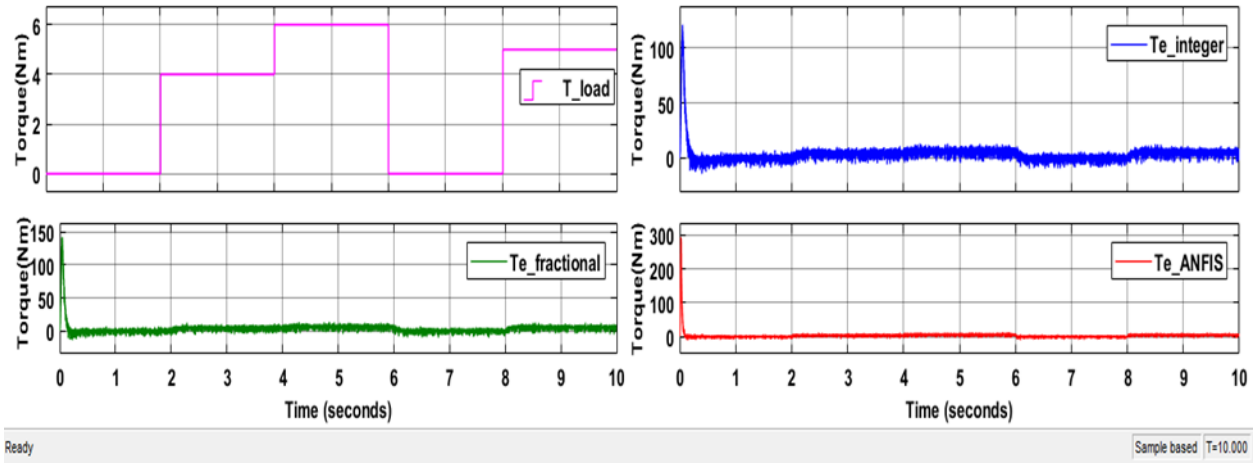
Figure 4.16 Response with load torque variation a) Speed b) Speed error

As shown in figure 4.17a only the quadrature axis stator current is changing when the load torque is changing. The direct axis current remains constant as in previous cases, But, it can be control the direct axis stator current independently at different reference value of reference flux

without affecting the quadrature axis stator current. Thus, these relationships will be indicated clearly shown that the torque and flux can be controlled independently. This proves decoupling which is the aim of vector control concept. As the previous results ANFIS controller results high initial quadrature axis current. But, it settles fast.Lastly, figure 4.17b shows the electromagnetic torque responses of IM drive using the three controllers. The simulation result shows all controllers can track the load torque. It is seen that the quadrature axis stator current increases and initially the speed decreases with increasing load and the electromagnetic torque follows the load torque. But, the speed settles back to the reference speed. This result is as expected.



(a)



(b)

Figure 4.17 Response with load torque variation a) Stator current b) Electromagnetic torque

CHAPTER FIVE

5. Conclusion and Recommendation

5.1 Conclusion

In this thesis, mainly four tasks have been performed. These are investigation of fractional order model of induction motor, design of GA optimized fractional and integer order PI controllers, design of ANFIS controller trained by data from GA optimized fractional order PI controller for indirect vector control of induction motor drive and finally, comparative analysis based on MATLAB/Simulink results of the above three controllers have been performed.

Analysis of speed and torque simulation results at no load with different values of differentiation order (α) below and above integer differentiation order i.e."1" have been performed to investigate fractional order model of IM. For fractional orders FOMCON toolbox was added to MATLAB and used for simulation. Different evaluation criteria such as peak torque, settling time of speed and torque, final value of speed and torque at no load condition are taken to select the optimized differential order (α). Based on analysis of simulation results integer order model is the optimized representation of IM.

GA optimized fractional and integer order PI controllers are developed for the integer order model of IM that has been modeled with some assumptions.GA optimized fractional order PI controller compared with that of conventional GA optimized PI controller performance. The simulation results obtained have confirmed the good dynamic performance and robustness of fractional order controller during the transient period and sudden load changes. This is due to fractional order PI controller allows us to adjust integration order λ in addition to the proportional and integral constants where the values of λ lie between 0 and 1. This gives extra freedom to operator in terms of extra knob i.e. order of integration. This provides more flexibility and opportunity to better adjust the dynamical properties of the control system. Simulation results show that fractional order controller has better performance than integer order controller for integer order modelled IFOC induction motor.

Due to the incorporation of the ANFIS controller in IM drive control, it was observed that the speed reaches the desired value quickly in a lesser time as compared with FOPI and IOPI controllers. Another advantage of the ANFIS is that its speed of operation is much faster than FOPI and IOPI; the tedious task of training of membership functions is done in ANFIS. Collectively, these results show that the ANFIS controller provides faster settling times, has

very good dynamic response and good stabilization. It is concluded that the proposed intelligent controller has shown superior performance than that of the fixed parameter fractional and integer order PI controllers.

5.2 Recommendations

ANFIS controller is used as an intelligent tool to design FLC. It helps to generate and optimize membership functions as well as the rule base from the simple data provided. ANFIS combine the learning power of neural network with knowledge representation of fuzzy logic. Even though, ANFIS eliminates unwanted trial and error method to determine parameters of membership functions using layers of ANN, it does not determine which type of membership function is suitable for the data provided. Therefore, additional investigations are needed to determine the type of membership function used in ANFIS controller.

In this thesis GA optimization technique is applied to determine the optimized parameters of fractional and integer order PI controllers, but there are other optimization techniques like Local Search Optimization (LSO), Sequential Quadratic Programming (SQP), and Particle Swarm optimization (PSO) those may improve the performance of controllers. The thesis can be also extended for hard ware implementation using Digital signal processing (DSP) and Field programmable gate array (FPGA).

References

[1]. P. Vas, “Sensorless Vector and Direct Torque Control”, London, U.K.: Oxford Science Publication,1998.

[2]. R. Krishnan, “Electric motor drives modeling, analysis and control”, PHI Pvt. Ltd., New Delhi, 2003.

[3]. Kumar, Rajesh, Gupta, R.A. and Singh, Bhim. “Intelligent Tuned PID Controllers for PMSM Drive -A Critical Analysis”, in the Proc. of IEEE International Conference on Industrial Technology (ICIT2006), Mumbai, INDIA, pp. 2055-2060, December 2006.

[4]. Z. Ibrahim and E. Levi, “A comparative analysis of fuzzy logic and PI speed control in high-performance AC drives using experimental approach,” IEEE Transl. on Ind. Appli, Vol. 38, No.5, pp. 1210–1218, Sep/Oct. 2002.

[5]. M.V. Aware, A.G. Kothari and S.O. Choube, “Application of adaptive neuro fuzzy controller (ANFIS)for voltage source inverter fed induction motor drive,” in the Proc. of IPEMC 2000 Conf., Vol. 2, pp. 935–939, 15-18 Aug 2000.

[6]. I. Podlubny,Fractional-order systems and PIλDμ-controllers, IEEE Trans. on Automatic Control, vol. 44, pp. 208-214, 1999.

[7]. M.Yahyazadeh, M. Haeri, Application of fractional derivative in control functions, India Conference. Annual IEEE, vol.1, pp. 252-257, 2008.

[8]. Hilfer R. “Applications of fractional calculus in Physics”, World Scientific, Singapore, 2000.

[9]. K. F. Man, K. S. Tang, and S. Kwong, “Genetic algorithms: concepts and applications,” IEEE Transactions on Industrial Electronics, vol. 43, no. 5, pp. 519–534, 1996.

[10]. Serein Al-Ratrout, Ashraf Saleem and, Hisham Soliman.” Optimization Methods in Fractional Order Control of Electric Drives” Trans. on Automatic Control, vol. 13, pp. 12-57, December, 2015.

[11]. Rajesh Kumar, R. A. Gupta and Rajesh S. Surjuse’’Adaptive Neuro Fuzzy Speed Controller for Vector Controlled Induction Motor Drive” Asian Power Electronics Journal, Vol. 3, No. 1, pp.8-14,Sept 2009.

[12]. Ramchandra Bhosale, Bindu R,” Indirect Vector Control of Induction motor using Fuzzy Logic Controller” IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE),

[13]. Aleksei Tepljakov, Eduard Petlenkov and Juri Belikov.,” Fractional Order Modeling and Control Toolbox for MATLAB” January 2011.

[14]. E. Petlenkov, J. Belikov, A.Tepljakov,and M. Halás, “Design and implementation of fractional-order PID controllers for a fluid tank system,” in Proc. of the 2013 American Control Conference (ACC), Washington DC, USA, pp. 1780–1785, June 2013.

[15]. Mehdi Dalir and Majid Bashour”Applications of Fractional Calculus” Applied Mathematical Sciences, Vol. 4, no. 21, pp.1021 – 1032.dec 2010.

[16]. Aleksei Tepljakov,” Fractional order Modeling and Control of Dynamic Systems”, Estonia, TUT press, 2015.

[17]. Kenneth S. Miller and Bertram Ross.An Introduction to the Fractional Calculus and Fractional Differential Equations, United States of America, John Wiley & Sons, Inc.,1993.

[18]. Carl M. Bender and Steven A.Orszag, Advanced Mathematical Methods for Scientists and Engineers, United States of America, Mcgraw-Hill Book Company,1978.
[19]. Ivo Petras, Fractional-Order Nonlinear Systems Modeling, Analysis and Simulation, Beijing, Beijing and Springer-Verlag Berlin Heidelberg,2011.
[20]. Riccardo Caponetto, Giovanni Dongola and Luigi Fortuna,Ivo Petras, Fractional Order Systems Modeling and Control Applications,Singapore,World Scientific Publishing Co. Pte. Ltd., 2011.
[21]. Aleksei Tepljakov," Fractional order Calculus based Identification and Control of Linear Dynamic Systems",Estonia ,Grant no. 8738,2011.
[22]. Zhuo Li," Fractional Order Modeling and Control of Multi input multi output Processes", License CC BY-SA 4.0, Jan 01, 2015.
[23]. Y. Chen, Petras, and D. Xue, "Fractional order control a tutorial," Proc. of the 2009 American Control Conference, (ACC '09), St. Louis, MO, USA, June 2009, pp. 1397–1411.
[24]. A. Oustaloup, P. Melchior, P. Lanusse, O. Cois, and F. Dancla, "The CRONE toolbox for Matlab," Proc. of the Computer-Aided Control System Design (CACSD), IEEE International Symposium, 2000, pp. 190–195.
[25]. Paul C.Krouse,Oleg Wasynczuk,Scott and D.Sudhoff,Analysis of electric Machinery and drive Systems, United States of America ,John Wiley and sons,2002.
[26]. Tze-Fun Chan and Keli Shi,Applied Intelligent Control Of Induction Motor Drives, Singapore ,John Wiley & Sons (Asia) Pte Ltd,2011.
[27]. Chee Mun ong, Dynamic Simulation of Electric Machinery Using Matlab/Simulink,upper Saddle river New Jersey,Princel Hall PTR,1998.
[28]. BinWu,Yongqiang Lang,Navid Zargari, and Samir Kouro,Power Conversion and Control of Wind Energy Systems, United States of America, John Wiley & Sons, Inc.,2011.
[29]. Seung-Ki Sul, Control of Electric Machine Drive Systems, United States of America, John Wiley & Sons, Inc.,2011.
[30]. Benoît Robyns and Philippe Degobert,Vector Control of Induction Machines Desensitization and Optimization through Fuzzy Logic, Verlag London, Springer, 2012.
[31]. R. J. LEE, P. PILLAY and R. G. HARLEY, "D, Q Reference Frames for the Simulation of Induction Motors", Electric Power Systems Research, 8 (1984/85), pp 15 -26, Received April 9, 1984.
[32]. Rakesh Singh Lodhi and Payal Thakur."Performance and Comparison Analysis of Indirect Vector Control of Three Phase Induction Motor". International Journal of Emerging Technology and Advanced Engineering, vol.3, pp. 716-724, Oct. 2013.
[33]. B. Srinu Naik." Comparison between Direct and Indirect Field Oriented Control of Induction Motor". International Journal of New Technologies in Science and Engineering, Vol.1, pp.110-131, Jan. 2014.
[34]. Venu Gopal B T,"Comparison between Direct and Indirect Field Oriented Control of Induction Motor". International Journal of Engineering Trends and Technology (IJETT), Vol.43, pp.364-369, January 2017.
[35]. Bose B.K, Modern Power Electronics and AC Drives 4th Edition, United States of America., John Wiley & Sons, Inc., 2004.

[36]. Jyh-Shing Roger Jang, Chuen-Tsai Sun and Eiji Mizutani, Neuro Fuzzy and Soft Computing A Computational Approach to Learning and Machine Intelligence, United States of America, Prentice-Hall, Inc.,1997.

[37]. Roland S.Burns, Advanced control Engineering, Boston, Johannesburg, Butterworth Heinenann(BH),2001.

[38]. Guanrong Chen and Trung Tat Pham, Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy Control Systems, United States of America, CRC,2001.

[39]. S. N. Sivanandam, S. Sumathi and S. N. Deepa, Introduction to Fuzzy Logic using MATLAB, Verlag Berlin Heidelberg, Springer, 2007.

[40]. Christopher MacLeod, An Introduction to Practical Neural Networks and Genetic Algorithms for Engineers and Scientists, the author retains copyright of this material.

[41]. Mohamed Watany, Manar A. Eltantawie, Shawki A. and Abouel seoud "Application of an Adaptive Neuro Fuzzy Inference System for Low Speed Planetary Gearbox Vibration Control". Journal of Low Frequency Noise, Vibration and Active Control, vol.34, pp.323-342, April 2015.

[42]. Arijit Biswas, Swagatam Das, Ajith Abraham and Sambarta Dasgupta, "Design of fractional-order $PI^{\lambda}D^{\mu}$ controllers with an improved differential evolution", Engineering Applications of Artificial Intelligence, Volume 22, Issue 2, pp. 343-350, March 2009.

Appendices

Appendix A. Expanded Simulink model of figure 4.1

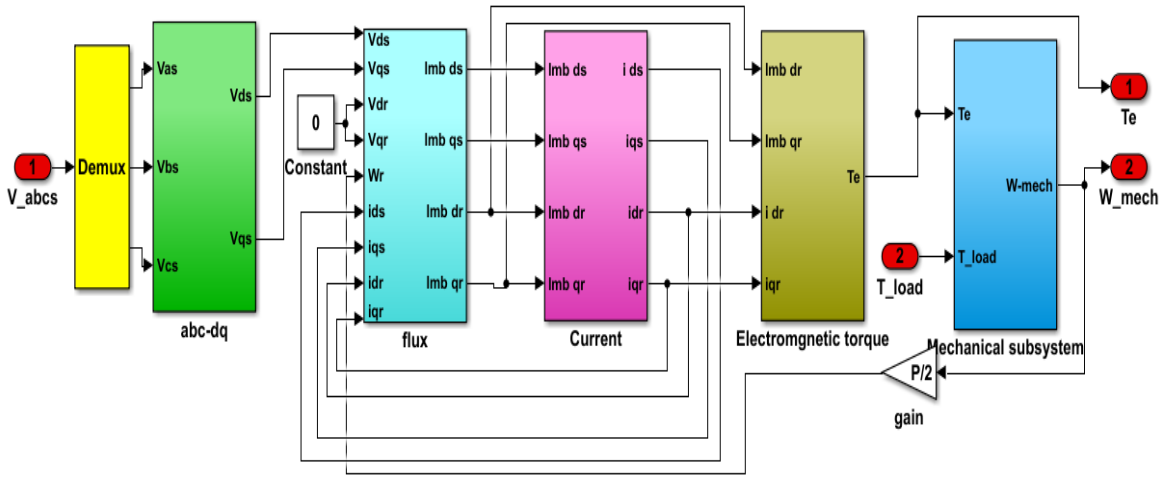


Figure A.1 Simulink model inside induction motor block

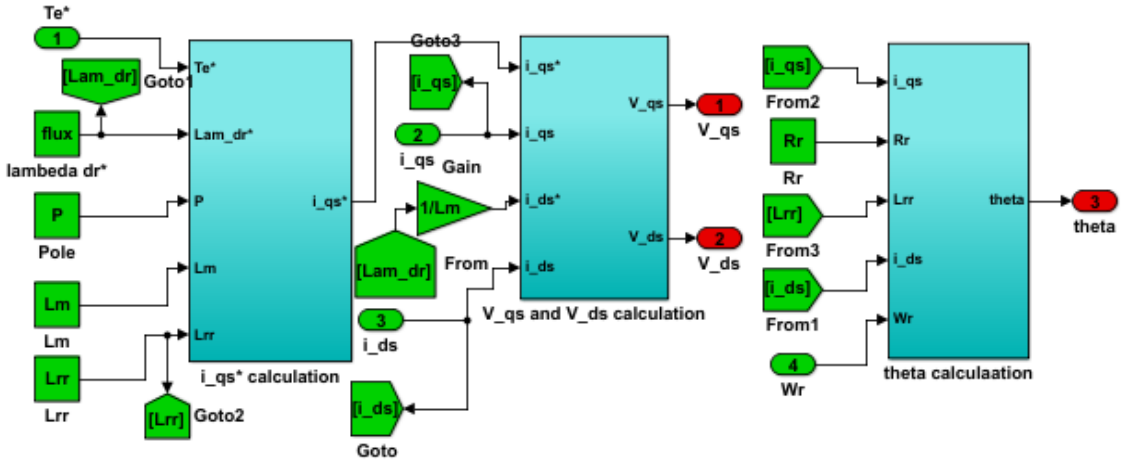
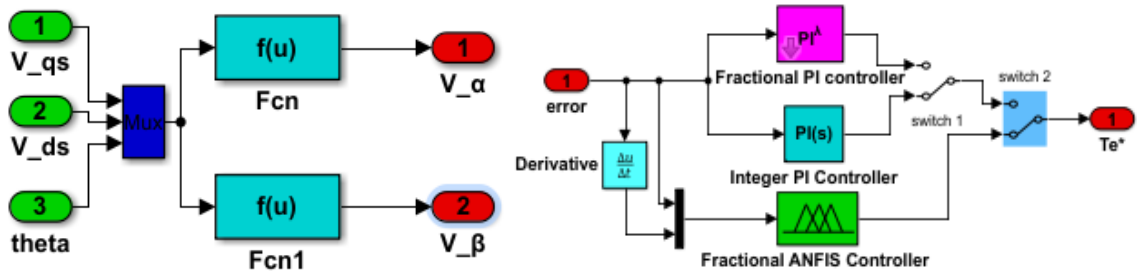


Figure A.2 Simulink model inside IFOC block



(a) (b)

Figure A.3 a). Clarke transformation, b). Simulink model inside controller block

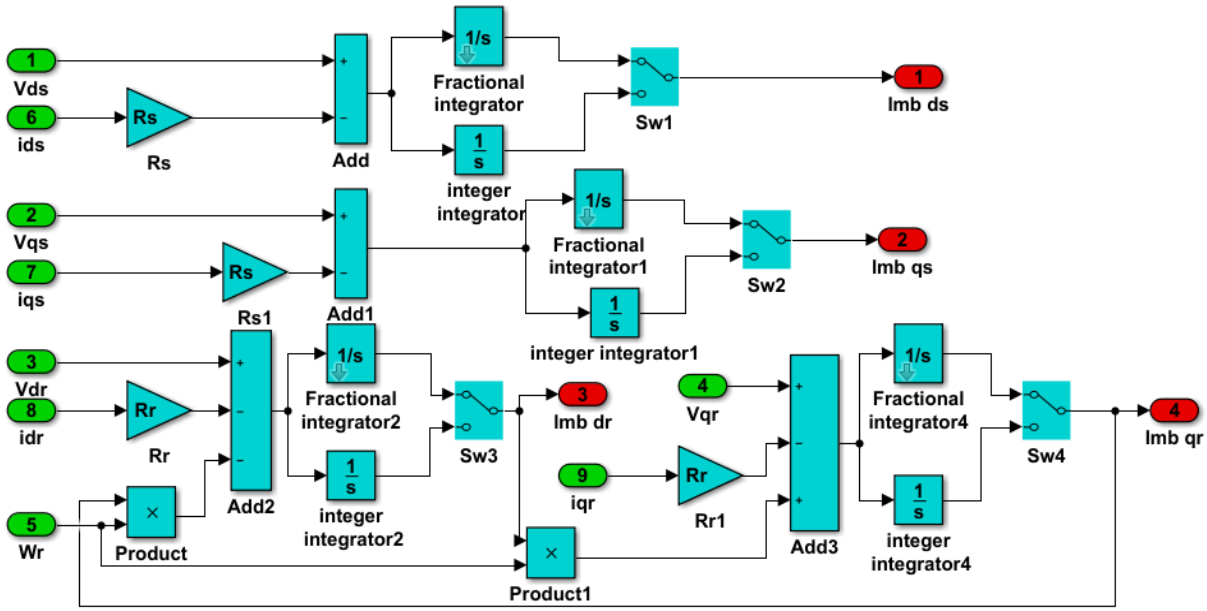


Figure A.4 Simulink model inside flux block to investigate Fractional order model of IM

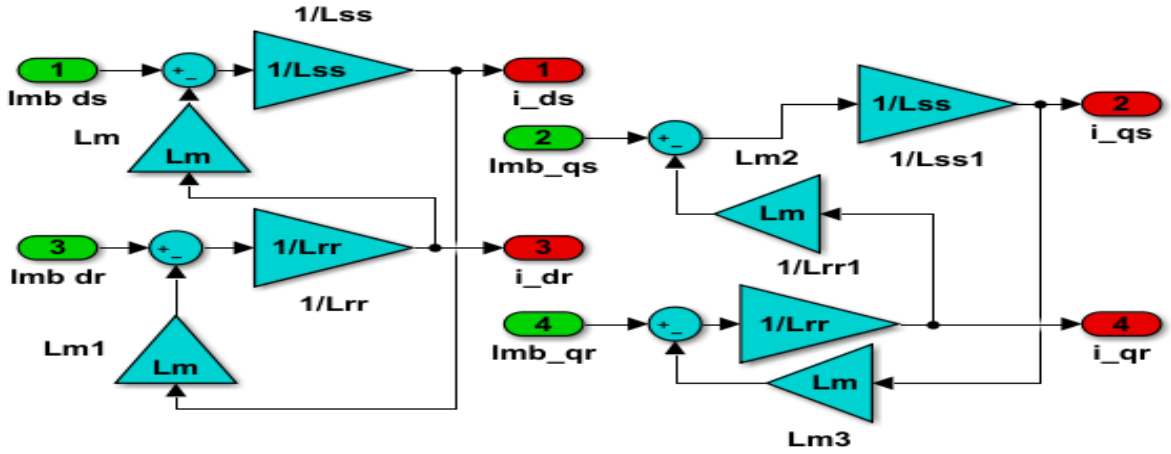


Figure A.5 Current calculations

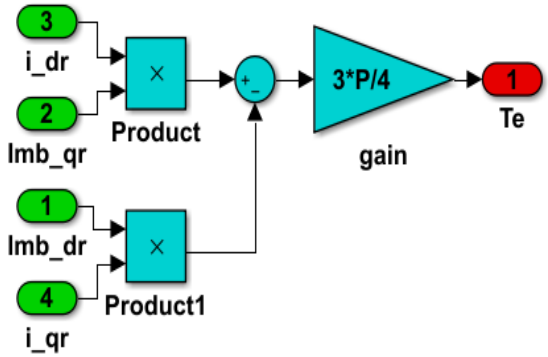


Figure A.6 Electromagnetic torque calculation

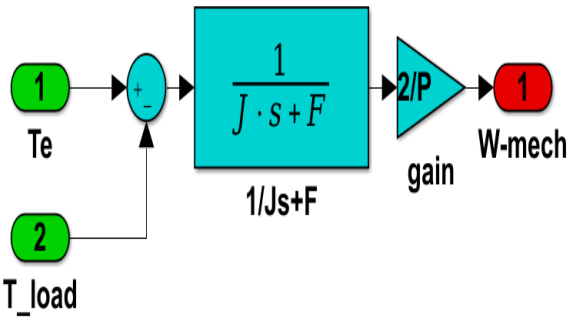


Figure A.7 Mechanical subsystem

Appendix B. MATLAB Codes of Genetic algorithm optimization and ANFIS controller Design

B.1. Programming a function to run the simulink model of FOPI controller

```
function s=fractional_PI(x) %function name x as input&s as output
assignin('base','x1', x(1));% Exchange variable value in 'caller'
and 'base' workspaces
assignin('base','x2', x(2));
assignin('base','x3', x(3));
[tout,xout,yout]=sim('FOPI_IM',5);% Perform a Simulink model for
5 seconds
z=yout;
[m,n] = size(z); % Calculating number of rows and number of
columns
V=0;
ref=50; %reference speed
for i=1:siz(z)
    V=V+(ref-z(i))^2;
end
s=V; %Calculating square error of the plant output and command
reference this is fitness function
end
```

B.2. Commands to run the simulation of FOPI controller for GA optimization

```
clc
clear all
options = gaoptimset(@ga); %Get GA parameters @ga Default values
of options
options=gaoptimset(options,'PlotFcns',{@gaplotbestf},'Display','i
ter');
options = gaoptimset(options,'PopulationSize',60);
options = gaoptimset(options, 'EliteCount',2);
options = gaoptimset(options, 'CrossoverFraction',0.8);
options = gaoptimset(options, 'Generations',300);
options=gaoptimset(options,'MutationFcn',@mutationadaptfeasible;
options= gaoptimset(options,'SelectionFcn',@selectionstochunif);
lb = [0;0;0];% Lower bounds of x1,x2,x3
```

```
ub = [20;40;1];%upper bound on x1,x2,x3
[x,fval,exitflag]=ga(@fractional_PI,3,[],[],[],[],lb,ub,[],option
s);% Perform GA,
assignin('base','x1',x(1));%Store final values to base workspace
assignin('base','x2',x(2));
assignin('base','x3',x(3));
```

B.3. Programming a function to run the simulink model of IOPI controller

```
function s=integer_PI(x) %function name x as input s as output
assignin('base','x1',x(1)); % Exchange variable value in 'caller'
and 'base' workspaces
assignin('base','x2',x(2));
[tout,xout,yout]=sim('IOPI_IM',5); % Perform a Simulink model for
5 seconds
z=yout;
[m,n] = size(z); % Calculating number of rows and number of
columns
V=0;
ref=50;  %command reference
for i=1:siz(z);
    V=V+(ref-z(i))^2;
End
s=V; %Calculating square error
end
```

B.4. Commands to run the simulation of IOPI controller for GA optimization

```
clc
clear
options = gaoptimset(@ga); %Get GA parameters @ga Default values
of options
options=gaoptimset(options,'PlotFcns',{@gaplotbestf},'Display','i
ter');
options = gaoptimset(options,'PopulationSize',60);
options = gaoptimset(options, 'EliteCount',2);
options = gaoptimset(options, 'CrossoverFraction',0.8);
options = gaoptimset(options, 'Generations',200);
options=gaoptimset(options,'MutationFcn',@mutationadaptfeasible;
```

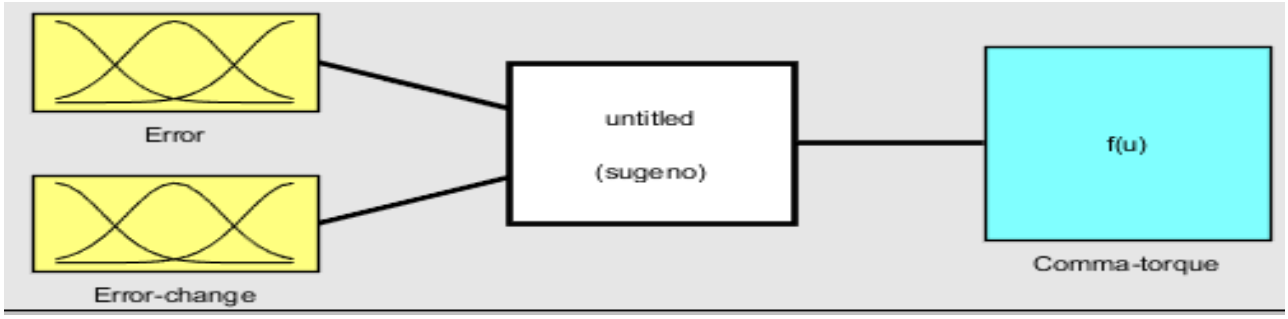
```
options= gaoptimset(options,'SelectionFcn',@selectionstochunif);
lb = [0;0]; % Lower bound on x1,x2
ub = [20;40]; % upper bound on x1,x2
[x,fval,exitflag] = ga(@integer_PI,2,[],[],[],[],lb,
ub,[],options);% Store final values to base workspace
assignin('base','x1',x(1));%Store final values to base workspace
assignin('base','x2',x(2));
```

B.5. MATLAB Code for ANFIS controller design using data from FOPI controller

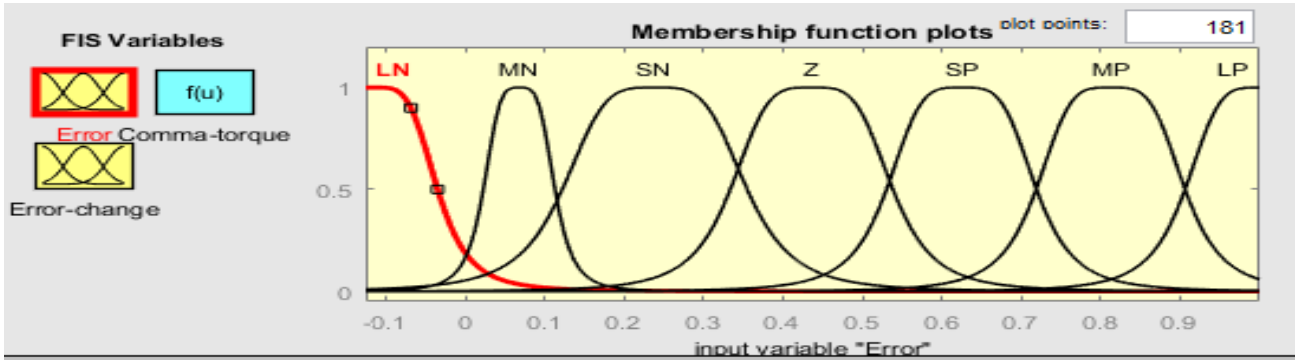
```
alldata=(noload;load)% data from no-load and with load condition
data=alldata(randsample(size(alldata,1),150000),:);%random data
selection
error=data(:,1);% error
error_change=data(:,2);% change in error
command_torque=data(:,3);% output command torque
max1=max(error);% maximum error
max2=max(error_change);%maximum change of error
max3=max(command_torque);%maximum torque or o/p of controller
input1=error/max1;%normalize error input between -1 and 1
input2=error_change/max2;%normalize error change between -1and 1
output=command_torque/max3;%normalize command torque b/n -1and1
nordata=[input1 input2 output];%normalized data between -1 and 1
traindata=nordata(1:105000,:); % training data pair
checkdata=nordata(105001:end,:); % checking data pair
```

B.6. ANFIS membership functions, rule and surface viewers

Membership functions for input and output variables have been chosen with generalized bell shaped and linear respectively as shown in Figure B.1. Universe of discourse of input and output variables are divided in to seven fuzzy sets. Those are LN (Large Negative), MN (Medium Negative), SN (Small Negative), Z (Zero), LP (Large Positive), MP (Medium Positive), SP (Small Positive).



(a)



(b)

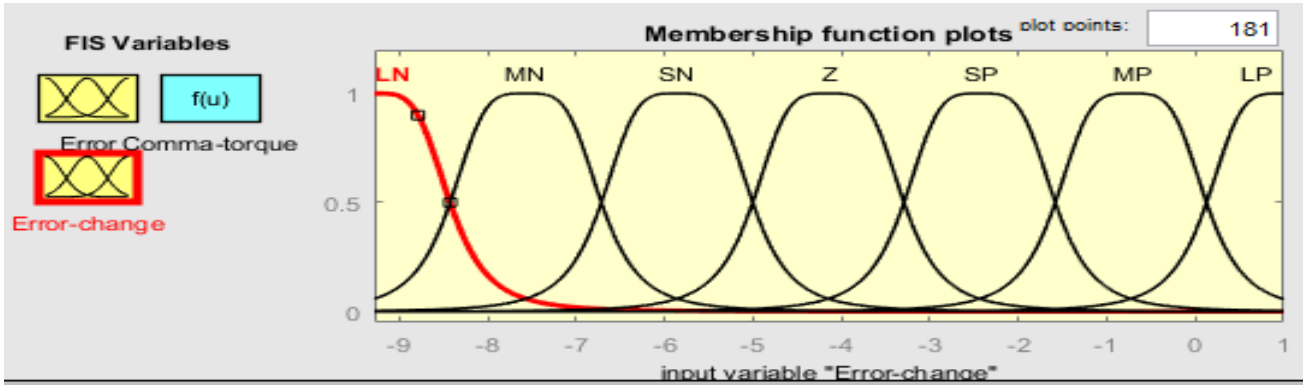
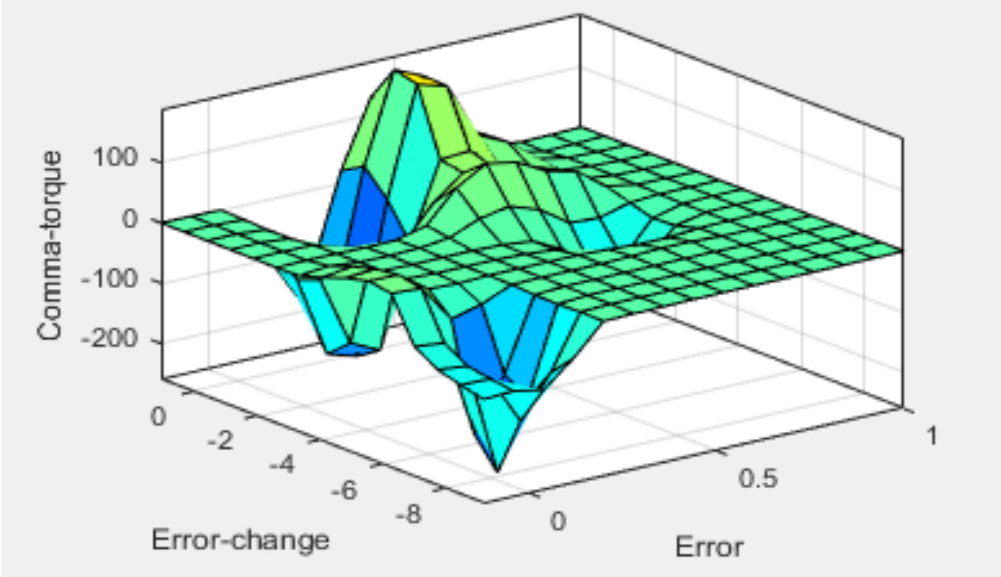
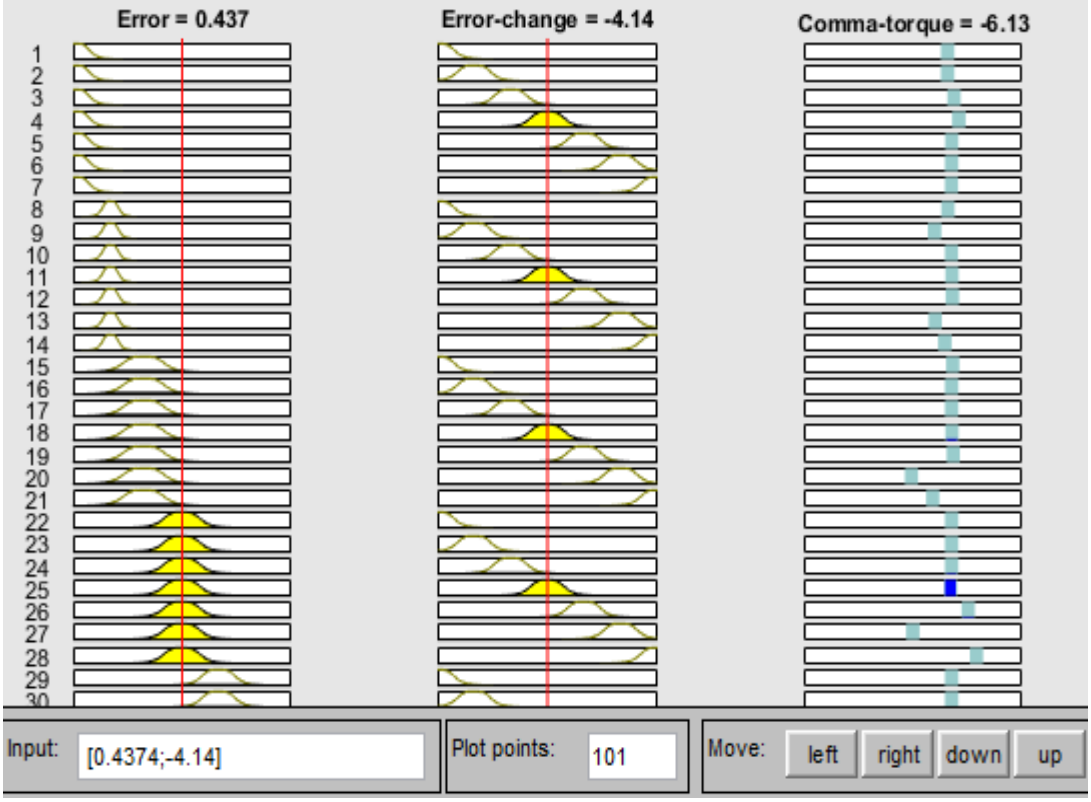


Figure B.1. a) FIS property editor b) Error MFs c) Error change MFs

(c)



(a)



(b)

Figure B.2 a) Surface viewer b) Rule viewer

The Surface Viewer is used to display the dependency of the output on any one or two of the inputs. It generates and plots an output surface map for the system. The output is presented by a 3D surface model.

Appendix C. Induction motor parameters

Table C.1 Parameters of Squirrel-cage induction motor used for simulation

Parameters	**Motor 1**	**Motor 2**	**Motor 3**
Power(KW)	2.24	7.457	7.5
Supply voltage (phase),Vs	230	230	220
Number of poles, P	2	4	6
Frequency(Hz)	60	60	60
Stator Resistance, Rs(ohm)	0.288	0.6837	0.282
Stator Leakage inductance, Lls(H)	0.0013	0.004152	0.001358
Rotor Resistance, Rr(ohm)	0.158	0.451	0.151
Rotor Leakage inductance, Llr(H)	0.0006	0.00415	0.000711
Mutual inductance, Lm(H)	0.0412	0.1486	0.03943
Inertia (Kg.m^2)	0.001	0.05	0.4
Friction factor, F(Nms^{-1})	0.4	0.00814	0.124
Rated rotor speed(rad/sec)	372.8	184.3	121.48