Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

DSP Based fuzzy logic Speed Control of Permanent Magnet Synchronous Motor

By:

HAJI ALI

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Computer Engineering (Control)

Advisor:

Dr. Mengesha Mamo

December 2017

Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirement for the Degree of Master of Science in Electrical and Computer Engineering (Control Engineering)

By:

HAJI ALI

Approval by Board of Examiners

Dr: _____	_____	_____
Chairman Department of Graduate Committee	Signature	Date
Dr.Mengesha Mamo	_____	_____
Advisor	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Declaration

I, the undersigned, declared that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other University and all sources and materials used for the thesis is acknowledged.

HAJI ALI _____

Name Signature

Addis Ababa, Ethiopia _____

Place Date of Submission

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

Dr.Mengesha Mamo _____

Advisor's Name Signature

ACKNOWLEDGMENT

Next to Allah-The Almighty God, I would like to express my sincere gratitude to my advisor Dr. Mengesha Mamo for his continuous support of my thesis, for his patience, motivation, enthusiasm, and most importantly, his friendship. His guidance helped me in all time of the research and writing of this thesis. Beside my advisor, I would like to thank all Stuff and Laboratory Assistants of Electrical and Computer Engineering Department for their assistance and encouragement throughout my work. Last but not least, I would like to thank my family for their great support during my study.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

List of Acronyms

AAiT	Addis Ababa Institute of Technology
AC	Alternating current
ADC	Analog digital convertor
CCS	Code Composer Studio
d, q	synchronous reference frame quantities (d-axis, q-axis)
DC	Direct current
DSPs	digital signal processors
DTC	Direct Torque Control
EMF	Electro Motive Force
FOC	field oriented control
HVMTRPFCKIT	High Voltage Motor Control and PFC Kit
IM	induction motor
IPark	inverse Park
IPMSM	Interior permanent magnet synchronous motor
ISR	Subroutine interrupts
JTAG Joint	Test Action Group
MI	Modulation index
MIMO	multiple input multiple output
MF	Membership functions
mmf	magneto-motive force
PI	Proportional Integral
PM	permanent-magnet
PMSM	permanent magnet synchronous motor
PWM	Pulse Width Modulation
QEP	Quadrature Encoder Pulse
RPM	revolution per minute
FIS	Fuzzy Inference System
FLC	Fuzzy Logic control

SMPMSM	surface mount permanent magnet motor
SRM	switched reluctance machines
SVPWM	Space vector pulse width modulation
THD	total harmonic Distortion
V/F	volt per frequency
α, β	α-axis and β-axis stationary frame quantities

ABSTRACT

In this thesis works, fuzzy logic controller is designed for PMSM in order to control the desired reference speed with robustness. Proportional, Integral (PI) controllers are also designed for direct and quadrature components of the current in order to control the flux and torque. The performance of the proposed fuzzy logic controller (FLC) for SMPMSM drive has to be simulated in MATLAB/SIMULINK environment.

The FLC operation has been verified by simulation on MATLAB/SIMULINK and the open loop experimental investigation is implemented using Texas Instrument, Piccolo TMS320F28035 control card on general purpose PMSM. From performance comparison simulation results of sudden load change and step change in speed reference, overshoot is improved from 3.5% to 1% and 5% to 3% respectively using proposed FL controller. Using FL speed controller maximum steady state error of 0.04 rad/s at time of 0.1 sec has been achieved. These demonstrate that the performance of proposed controller is better than that of convectional PI controller.

Keywords: PMSM, Fuzzy logic controller, PI Controller, FOC and TMS320 F28035 DSP

CHAPTER ONE
INTRODUCTION

1.1 Background

Permanent magnet synchronous motor drives are becoming more popular in industries and are replacing induction motor drives due to their high performances: high torque density, high efficiency and small size. PMSM have no windings in the rotor instead they have rotating permanent magnets made of neodymium-boron- iron , Alnico or samarium cobalt that retain their magnetic property. The magnets can be mounted on the surface of the rotor for medium speed operations or placed internally inside the rotor for high speed operations [1].

The conventional control theory such as proportional integral (PI) controller which relies on mathematical model has been successfully and widely used in industrial applications due to its simple control structure, easy of design and low cost. The controller is usually applied to the control of a large variety of simple and linear processes. However, it has not been as widely used with complex and nonlinear systems [2]. To overcome the disadvantages of conventional controller of motor drive, it would be desirable to design a well controller. The fuzzy logic control (FLC) is seemed to be a suitable controller in terms of high dynamic response under the variation of load torque and step change in speed reference [3].

In the last few years, fuzzy logic has a growing interest in many motor control applications due to its non-linearity handling features and independence of the plant modeling. The Fuzzy Logic Controller (FLC) operates in a knowledge-based way. Fuzzy-based control method has the ability to deal with system nonlinearity and its control performance is less affected by load variations. Fuzzy techniques uses a linguistic if then rule base which is designed by taking advantage of system qualitative aspects and expert knowledge [4]. Fuzzy logic control is a non-mathematical decision algorithm that is based on an operator's experience.

In this Thesis, a fuzzy logic controller has been designed and simulated on MAT LAB/SIMULINK for PMSM drive. Open loop experiment using Texas Instruments DSP TMS320F28035 Control Card was attempted.

1.2 Problem statement

To achieve desire level of performance all motors require suitable speed controllers. In case of permanent magnet motors, usually speed control is achieve by using proportional-integral (PI) controller. Although conventional PI controllers are widely used in the industry due to their simple control structure and ease of implementation, these controllers pose difficulties where there are some control complexity such as nonlinearity, load disturbances and parametric variations. Moreover PI controllers require precise linear mathematical models. As the PMSM machine has nonlinear model, the linear PI may no longer be suitable.

Fuzzy Logic (FL) approach applied to speed control leads to improve dynamic behavior of the motor drive system and an immune to load disturbance and parameter variations. These controllers are inherently robust to load disturbances. The basic reason for the popularity of Fuzzy Logic Controllers is that its logical resemblance to a human operator. It operates on the foundations of a knowledge base which in turn rely upon the various if then rules, similar to a human operator. Unlike other control strategies, this is simpler as there is no complex mathematical knowledge required. The FLC requires only a qualitative knowledge of the system thereby making the controller not only easy to use, but also easy to design modification.

1.3 Objective

This thesis is intended to design fuzzy logic speed controller, which is artificial intelligent technique, for a permanent magnet synchronous motor drive, in conjunction with field oriented control. It is expect that this control scheme can improve performance characteristic of motor drive system and track the reference speed well under parameter uncertainties and load torque disturbance.

To achieve the main objectives the following specific objective are formulated as follows:

- ✓ Studying the characteristic and modeling of the PMSM motor drive system
- ✓ Design PI controller for controlling the inner loop
- ✓ Develop Simulation on MATLAB/SIMULINK and experimental setup for the system on DSP
- ✓ Performances analysis with FLC

1.4 Methodology

For the accomplishment of fuzzy logic speed controlling of PMSM, the following methodologies has been followed:

- Gathering & surveying related works from various literature and publication
- Study system model and Design of a fuzzy logic Controller
- Analyze the performances FLC with Mat lab/Simulink
- The open loop system algorism has been developed using C language.
- Develop experimental set up and test of open loop Control Algorithms using Texas Instruments DSP kit.

1.5 Scope and limitation

In this thesis work, design and simulation of fuzzy logic controller for PMSM was done using MATLAB/SIMULINK. And the implementation has been done on DSP kit with TMS320F28035 piccolo control card.

In AAit motor control and drive Laboratory, the PMSM has no parameter specification labeled so the value is taking from the Hamdihun's paper [5], that was done on this motor. Due to inaccessibility of the mechanical position sensors the implementation is limited to open loop system control.

1.6 Literature review

Speed control of PM (permanent magnet) motor drives has been a topic of interest for the last twenty years. And different papers have been published reporting different controller design for such drives.

Ahmad FikriHilmie, Farrukh Hafiz Nagi in [2] develop Pulse Width Modulation (PWM) fuzzy controller for eZ-DSP F2812 using Simulink in MATLAB. The controller is designed to process the input and gives output to control the duty cycle of PWM which control the speed of Brushless DC (BLDC) motor. The fuzzy controller that had been developed in Simulink then is implemented on eZ-DSP F2812 using C language in Code Composer Studio environment. The eZ-DSP F2812 is used as a speed controller for BLDC motor in real-time. The eZ-DSP F2812 does not produce PWM sequence for BLDC winding excitation instead an electronic commutation (EC) driver is used for this purpose, which takes input from eZ-DSP F2812 fuzzy controller. Ms. Madhavi Mallam , M. Bhanu Sri , Dr.A. Guruva Reddy in [6] are design and implement Fuzzy Logic speed control of DC motor and simulate on mat lab software. In this paper TMS320F28335 DSP is used

as the controller necessary signal conditioning components are used to ensure high processing speed and precision in the overall control system. Mahlet Legesse in [7] implemented Mamdani type Fuzzy-PI speed controller for permanent magnet synchronous motor (PMSM) on eZ-DSP F2812 by generating C code using Real Time Workshop (RTW) embedded coder in MATLAB linked to Code Composer Studio. Binita Nanda and A.N Thakur in[8] proposes a Fuzzy logic control scheme for a PMSM drive driven by Field Oriented Control (FOC) and also study the performance comparison of the proposed IPMSM drive with conventional PI controller based drive in MATLAB/Simulink environment. The inputs to the FLC are Torque error (e) and change in torque error (ce). The FLC initially converts the crisp error and change in torque error into fuzzy variables and then are mapped into linguistic labels. The inputs and output contain membership functions with five linguistic variables. G. Sakthivel, T.S. Anandhi and S.P. Natarjan in[9] proposed an approach for implementation of a Mamdani type FLC for BLDC motor on eZ-DSP F2812 interfacing with Vissim. The implementation of FLC is very straightforward by coding each component of fuzzy inference system in Vissim which is an environment for model based development of embedded controller for Texas instruments DSPs according to design specifications.

A number of other papers also present the advantage of Fuzzy logic controller techniques over PI and verified to implement sensored and sensor less control of PMSM.

1.7 Outline of the thesis

This thesis includes six chapters. The **first chapter** generally presents the introduction of the speed control of PMSM using fuzzy logic controller which is the background of the PMSM speed control, statement of the problem, objectives of the study, methodology, scope and limitation of study and different literatures related to speed control of PMSMs are reviewed.

Chapter two describes general characteristic of PMSM. Overview of PMSM, Permanent Magnet Materials, Classification of permanent magnet motor**,** advantage and different applications, mathematical modeling of PMSM, transformations and different control methods are presented.

Chapter three Design of speed controller of PMSM using fuzzy logic and current controller by PI, Space Vector PWM and position sensors are presented.

Chapter four discusses about the system simulation and results. The performance of the Fuzzy Logic Controller is evaluated by simulation study using Matlab/Simulink under different conditions.

Chapter five present hardware key feature and system software organization, experimental setup and result of PMSM open loop system.

Chapter six presents the conclusion and recommendations for future enhancements.

CHAPTER TWO

GENERAL CHARACTERISTICS OF PM SYNCHRONOUS MOTOR

2.1 Introduction

Synchronous motors are those which rotate at the speed of the stator revolving field, which is called the synchronous speed. The stator of a three-phase synchronous motor normally has a sine distributed three-phase winding. When excited with a three-phase balanced supply, a rotating magnetic field develops. Unlike induction motor synchronous motor is excited by a permanent magnet source to cause the rotor to 'lock-on' or 'synchronise with' the rotating magnetic field produced by the stator. Once the rotor is synchronised, it will run at exactly the same speed as the rotating field despite load variation, so under constant-frequency operation the speed will remain constant as long as the supply frequency is stable. The synchronous speed, is determined by the frequency of the stator supply, and the number of stator pole pairs, p.

The synchronous speed (in rev/min) is given by the expression

$$N_s = \frac{120f}{nP} \qquad 2.1$$

Where f is the supply frequency, Ns is Synchronous speed and np is the pole number of the winding.

2.2 Overview of permanent magnet synchronous motor

A permanent magnet synchronous motor (PMSM) is a motor that uses permanent magnets to produce the air gap magnetic field rather than using electromagnets. These motors have significant advantages, attracting the interest of researchers and industry for use in many applications.

2.2.1 Permanent Magnet Materials

The properties of the permanent magnet material will affect directly the performance of the motor and proper knowledge is required for the selection of the materials and for understanding PM motors. The earliest manufactured magnet materials were hardened steel. Magnets made from steel were easily magnetized. However, they could hold very low energy and it was easy to demagnetize. In recent years other magnet materials such as Aluminum Nickel and Cobalt alloys (ALNICO), Strontium Ferrite or Barium Ferrite (Ferrite), Samarium Cobalt (First generation rare earth magnet) (SmCo) and Neodymium Iron-Boron (Second generation rare earth magnet) (NdFeB) have been developed and used for making permanent magnets. The rare earth magnets are categorized into two classes: Samarium Cobalt (SmCo) magnets and Neodymium Iron Boride

(NdFeB) magnets. SmCo magnets have higher flux density levels but they are very expensive. NdFeB magnets are the most common rare earth magnets used in motors these days [10].

2.2.2 Classification of permanent magnet motor

2.2.2.1 Direction of field flux

PM motors are broadly classified by the direction of the field flux. The first field flux classification is radial field motor meaning that the flux is along the radius of the motor. The second is axial field motor meaning that the flux is perpendicular to the radius of the motor.

Radial Field Motors

The core of a PMSM are the permanent magnets (PMs), which are placed in the rotor to provide the magnetizing flux. The magnets can be placed in two different ways on the rotor. Depending on the placement they are called either as surface permanent magnet motor or interior permanent magnet motor [11].

Surface mounted PM motors have a surface mounted permanent magnet rotor. Each of the PM is mounted on the surface of the rotor, making it easy to build, and specially skewed poles are easily magnetized on this surface mounted type to minimize cogging torque. The permeability of the permanent magnet is almost that of the air, thus the magnetic material becoming an extension of the air gap. For a surface permanent magnet motor Ld=Lq[12].

Interior PM Motors have interior mounted permanent magnet rotor. Each permanent magnet is mounted inside the rotor. It is not as common as the surface-mounted type but it is a good candidate for high-speed operation. There is inductance variation for this type of rotor because the permanent magnet part is equivalent to air in the magnetic circuit calculation. These motors are considered to have saliency with q axis inductance greater than the d axis inductance (Ld<Lq).

2.2.2.2 Flux density distribution

PM motors are classified on the basis of the flux density distribution and the shape of Current excitation. They are PMSM and PM brushless motors (BLDC). The PMSM has a Sinusoidal-shaped back EMF and is designed to develop sinusoidal back EMF waveforms.

They have the following:

I. Sinusoidal distribution of magnet flux in the air gap
II. Sinusoidal current waveforms
III. Sinusoidal distribution of stator conductors.

BLDC has a trapezoidal-shaped back EMF and is designed to develop trapezoidal back EMF Wave forms. They have the following:

I. Rectangular distribution of magnet flux in the air gap
II. Rectangular current waveform
III. Concentrated stator windings.

2.3 Why concentrate this study on PMSM?

The significant advantages of PMSM attracting researchers and industries make it highly competitor to other motors like Induction Motors & DC Motors. PMSMs have many advantages. To mention some;

- They have high torque to inertia (lower weight). That is better dynamic performance than conventional one,
- High power density,
- High efficiency (this motor do not require field windings, they do not have field circuit copper loss) and reliability,
- Avoidance of brushes and slip rings makes the machine less audible noise, longer life, sparkles (no fire hazard) and is used for high speed applications.

Even though PM machines have aforementioned merits, they have the following demerits:

- They have complex control,
- The risk of demagnetization of poles which may be caused by large armature currents. Demagnetization of can also occurs due to excessive heating and also when the rotor is overloaded for long period of time,
- Extra ampere turn cannot be added to reduce the armature reaction,
- There is a problem of maintenance of rotor magnet.

2.4 Applications

Among many applications of PM synchronous motor, here are the following:

Industry

- Industrial drive
- Machine tools
- Servo drive
- Robots

Public life

- Hating , ventilations and air conditioning
- Auto bank machine
- Environmental control system
- Automatic vending machine

Domestic life

- Kitchen equipment
- Bath room equipment
- Washing machine and cloth dryer
- Vacuum cleaner

Information and office equipment

- Computer
- Printer
- Scanner
- Auto visual aids

Defense force

- Missile
- Radar system

Aero space

- Rocket
- Space shuttle
- satellite

2.5 Modeling of pm drive system

PMSM is very similar to the standard wound rotor synchronous machine except that the PMSM has no damper windings and excitation is provided by a permanent magnet instead of a field winding. Hence the d, q model of the PMSM can be derived from the well-known model of the synchronous machine with the equations of the damper windings and field current dynamics removed [13].

2.5.1 Mathematical Modeling of PMSM

The two axis mathematical modeling of PMSM is required for proper control and simulation of the system. The d-q model has been developed on rotor reference frame as shown in Figure 2.1. At any time t, the rotating rotor d-axis makes and angle θr with the fixed stator phase axis and rotating stator mmf makes an angle α with the rotor d-axis. Stator mmf rotates at the same speed as that of the rotor.

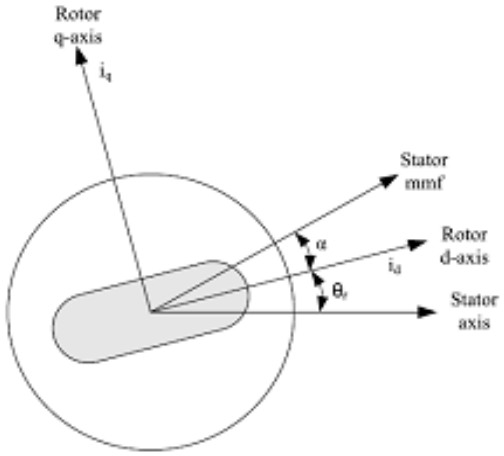


Figure 2.1: Motor axis

The model of PMSM without damper winding has been developed on rotor reference frame using the following assumptions:

1) Saturation is neglected.
2) The induced EMF is sinusoidal.
3) Eddy currents and hysteresis losses are negligible.
4) There are no field current dynamics

Mathematical modeling of PMSM [12][14][15].

Voltage equations are given by:

$$V_q = R_s I_q + \omega_r \lambda_d + \frac{d\lambda_q}{dt} \qquad 2.2$$

$$V_d = R_s I_d - \omega_r \lambda_q + \frac{d\lambda_d}{dt} \qquad 2.3$$

Where, V_d and Vq are the (d, q) axis voltages, iq and id are the (d, q) axis currents. L_d and L_q are the d, q axis inductances, while Rs is the stator winding resistance.

Flux Linkages are given by

$$\lambda_q = L_q I_q \quad 2.4$$

$$\lambda_d = L_d I_d + \lambda_f \quad 2.5$$

Substituting Equations (2.4) and (2. 5) in to Equations (2.2) and (2.3) and Arrange in matrix form

$$\begin{pmatrix} Vq \\ Vd \end{pmatrix} = \begin{pmatrix} R_s + \rho L_q & \omega_r L_d \\ -\omega_r L_q & R_s + \rho L_q \end{pmatrix} \begin{pmatrix} I_q \\ I_d \end{pmatrix} + \begin{pmatrix} \omega_r \lambda_f \\ 0 \end{pmatrix} \quad 2.6$$

For surface mounted PMSM The developed torque motor is being given by

$$T_e = \frac{3}{2} p \lambda_f I_q \quad 2.7$$

The mechanical torque equation

$$T_e - Tl = J\frac{d\omega}{dt} + B\omega \quad 2.8$$

$$\theta_r = \frac{P}{2}\theta_m \quad 2.9$$

2.5.2 Equivalent Circuit of Permanent Magnet Synchronous Motor

Equivalent circuits of the motors are used for study and simulation of motors. The equivalent circuit model of permanent magnet synchronous machine in the rotor d-q reference frame in [12] is shown in Figure 2.2. The motor equivalent circuit can be also derived from the dynamic d-q modeling using equation 2.6.

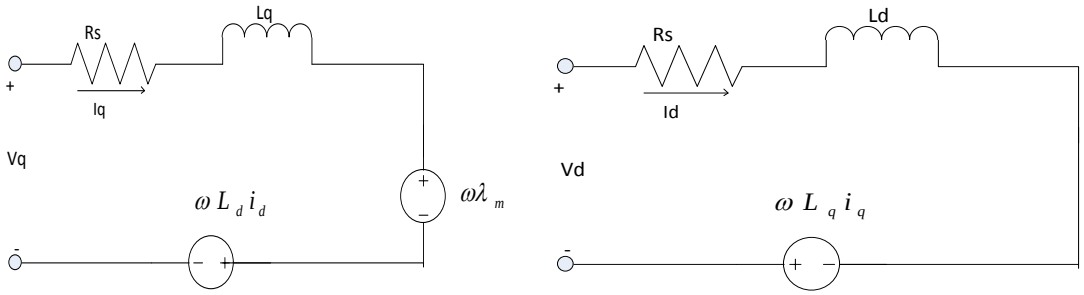


Figure 2.2: Equivalent circuit of PMSM

2.5.3 Transformation

Transforming from a stator reference frame of a PMSM into a rotary reference frame is used to get the time in varying values of the motor parameters. In FOC, the components are measured in the

rotating reference frame. Hence the measured stator currents have to be transformed from the three phase time variant stator reference frame to the two axis rotating d-q rotor reference frame.

Clarke transformation

The three-phase quantities are translated from the three-phase reference frame to the two-axis orthogonal stationary reference frame using Clarke transformation. The Clarke transformation is expressed by the following equations [16][17].

$$i_\alpha = i_a$$
$$i_\beta = (i_a + 2i_b)/\sqrt{3} \qquad 2.10$$

Where,

$$i_a + i_b + i_c = 0 \qquad 2.11$$

Park transformation

The two-axis orthogonal stationary reference frame quantities are transformed into rotating reference frame quantities using Park transformation. The Park transformation is expressed by the following equations [16][17].

$$i_d = i_\alpha \cos\theta + i_\beta \sin\theta$$
$$i_q = i_\beta \cos\theta - i_\alpha \sin\theta \qquad 2.12$$

Inverse Park transformation

The quantities in rotating reference frame are transformed to two-axis orthogonal stationary reference frame using Inverse Park transformation. The Inverse Park transformation is expressed by the following equations [16][17].

$$V_\alpha = V_d \cos\theta - V_q \sin\theta$$
$$V_\beta = V_q \cos\theta + V_d \sin\theta \qquad 2.13$$

Inverse Clarke transformation

The transformation from a two-axis orthogonal stationary reference frame to a three-phase stationary reference frame is accomplished using Inverse Clarke transformation. The Inverse Clarke transformation is expressed by the following equations [16][17].

$$V_a = V_\alpha$$
$$V_b = (-V_\alpha + \sqrt{3}V_\beta)/2$$
$$V_c = (-V_\alpha - \sqrt{3}V_\beta)/2 \qquad 2.14$$

The zero sequence components are important if three phase system is not balanced. If it is, the zero sequence components are neutral.

2.5.4 Control methods for PMSM

Vector and scalar control are the two broad categories of controlling the PMSM. Scalar control of the PMSM is also called Voltage/Hz control, in which magnitude of the voltage (V) is varied according to the frequency (f) in a constant ratio V/f. This method is open loop control as it does not uses any feedback such as rotor position or speed. So Voltage/Hz method does not have control over the torque. The V/f method is simple but its dynamic performance is poor and it will introduce high torque ripple [18].

Direct torque control (DTC) and field oriented control (FOC) are few of the vector control methods. In the direct torque control method, the stator flux will be adjusted based on the applied stator voltage. The change in the stator flux, changes the Te. As the torque control is directly based on the electromagnetic state of the motor, similar to the DC motor, this method is called direct torque control. The main advantage of this technique is good dynamic torque control performance [15].

In the FOC, a decoupled control of the torque and flux can be achieved by converting the measured three phase current into the d-q co-ordinate system using the park and Clarke transformations. This transformation also helps to implement the control algorithm similar to the DC motor control. Considering the rotor flux is perfectly oriented in equation (2.6) shows that by varying the Iq-current Te can be controlled. Generally the d-current is used only when the PMSM is operated in the field weakening region otherwise it is kept zero and the rotor flux will be constant due to the presence of the permanent magnet. As the FOC involves more mathematical transformations, its control algorithm is complex compared to the DTC method. But the FOC has a good steady state performance compared to the other control methods [18]. This thesis is based on FOC method.

Vector control in [19] is an elegant method to control a Permanent Magnet Synchronous Motor (PMSM), in which a field-oriented theory controls space vectors of magnetic flux and current. It is possible to set up the coordinate system to decompose the vectors into a magnetic field-generating function and a torque-generating function. The structure of the motor controller (vector control controller) is then almost the same as for a separately-excited DC motor, which simplifies the control of PMSM. This vector control technique was developed specifically to achieve a similarly dynamic performance in PMSMs.

CHAPTER THREE

CONTROLLER DESIGN OF SYSTEM

3.1 Introduction

Speed control of permanent magnet motors mainly consists of two loops, the inner loop for current and the outer loop for speed. The difference between the reference value of the rotor speed and actual rotor speed serves as the input signal to the speed controller producing an error. Based on the error, the speed controller generates a required q axis current, which corresponds to torque (fed to the inner loop current controller). The Fuzzy logic controller is designed for PMSM in order to control the desired reference speed of outer loop. Proportional, Integral (PI) controllers are also designed for direct and quadrature components of the current in order to control the flux and torque for inner loop.

The general system block diagram for speed control of PMSM using FLC controller is shown in figure 3.1. The system basically includes speed sensor (position sensor) to indicate rotor position information and Controllers (fuzzy logic controller, generation of reference currents and PI controller). The system also includes different co-ordinate transformation, such as Clarke, Park and Inverse Park for the purpose of vector control. Space vector pulse width modulation (SVPWM) is also part of the system which calculates the operation time of the inverter gates from the controlled d-q space vector voltage value for proper application voltages on the stator windings.

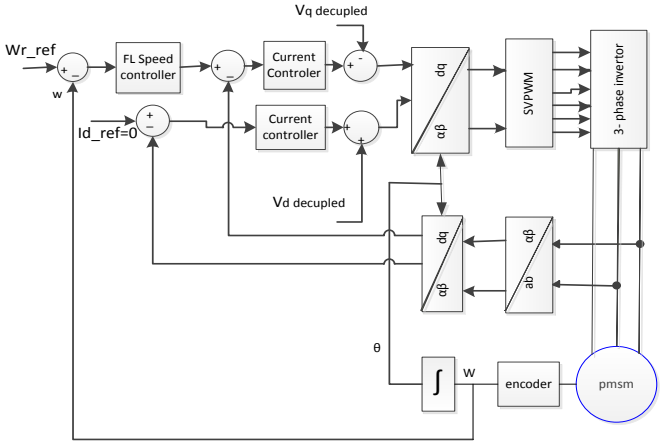


Figure 3.1: system model for speed control of PMSM using FLC controller

3.2 Fuzzy logic control

The concept of "Fuzzy Logic" was first introduced by Lotfi A. Zadeh in 1965 with a novel proposal of Fuzzy Set Theory. Fuzzy logics had been studied since the 1920s as infinite-valued logics notably by Łukasiewicz and Tarski. Fuzzy logic theory is an artificial intelligence method which has been employed to many fields like control theory to artificial intelligence [8].

Fuzzy logic has rapidly become one of the most successful of today's technology for developing sophisticated control system. With it aid complex requirement so may be implemented in amazingly simple, easily minted and inexpensive controllers. The past few years have witnessed a rapid growth in number and variety of application of fuzzy logic. The application range from consumer products such as cameras ,camcorder ,washing machines and microwave ovens to industrial process control ,medical instrumentation ,and decision-support system .many decision-making and problem solving tasks are too complex to be understand quantitatively however ,people succeed by using knowledge that is imprecise rather than precise. Fuzzy logic is all about the relative importance of precision. Fuzzy logic has two different meanings .in a narrow sense, fuzzy logic is a logical system which is an extension of multi valued logic .but in wider sense fuzzy logic is synonymous with the theory of fuzzy sets. Fuzzy set theory is resembles approximate reasoning in it use of approximate information and uncertainty to generate decisions.

Several studies show, both in simulations and experimental results, that Fuzzy Logic control yields superior results with respect to those obtained by conventional control algorithms thus, in industrial electronics the FLC control has become an attractive solution in controlling the electrical motor drives with large parameter variations like machine tools and robots.

The advantages provided by a FLC are listed below:

- It provides a hint of human intelligence to the controller.
- It is cost effective.
- No mathematical modeling of the system is required.
- Linguistic variables are used instead of numerical ones.
- Non-linearity of the system can be handled easily.

These advantages allow fuzzy controllers can be used in systems where description of the process and identification of the process parameters with precision is highly difficult. Hence it provides a fuzzy characteristic to the control mechanism.

3.3 Components of FLC

The inputs to a Fuzzy Logic Controller are the processed with the help of linguistic variables which in turn are defined with the aid of membership functions. The membership functions are chosen in such a manner that they cover the whole of the universe of discourse. To avoid any discontinuity with respect to minor changes in the inputs, the adjacent fuzzy sets must overlap each other. The basic internal structure of fuzzy logic controller block diagram is shown in Fig.3.2 and usually consists of:

I. A Fuzzification unit which maps measured inputs of crisp value into fuzzy linguistic values to be used by a fuzzy reasoning mechanism. In FLC, the input is always a crisp numerical value limited to the universe of the input variable and the output is fuzzy degree of membership in the qualifying linguistic set (always between 0 and 1).

II. A Knowledge rule base which is the collection of expert control knowledge required to achieve the control objective. Each rule of the FLC is characterized with an IF part, called the antecedent, and with a THEN part called the consequent. The antecedent of a rule contains a set of conditions and the consequent contains a conclusion. If the conditions of the antecedents are satisfied, then the conclusions of the consequent apply.

III. A Fuzzy reasoning mechanism (inference engine) that performs various fuzzy logic operations to infer the control action for the given fuzzy inputs. The Inference Mechanism provides the mechanism for referring to the rule base such that the appropriate rules are fired. The two most commonly used inference procedures in FLC are Mamdani's Max-Min and Max-Algebraic Product (or Max-Dot) composition. In this thesis a Mamdani's Max-min composition inference method is used.

IV. A Defuzzification unit which converts the inferred fuzzy control action into the required crisp control values to be entered into the system process. Remember, the fuzzy conclusion or output is still a linguistic variable, and this linguistic variable needs to be converted to the crisp variable via the defuzzification process. Three defuzzification techniques are commonly used, which are:

   a. Center of gravity(COG)
   b. Mean of maxima(MOM)
   c. Height Method (HM)

Unfortunately, there is no systematic procedure for choosing a defuzzification strategy. In this thesis a center of gravity defuzzification method is adopted for, which can reflect the overall inference information.

a. Center of gravity method

This procedure is the most prevalent and physically appealing of all the defuzzification methods. It is given by the algebraic expression as in equation 3.0.

$$Z = \frac{\sum_{i=1}^{n} SiFi}{\sum_{i=1}^{n} Fi} \qquad 3.0$$

Where Z is output from defuzifications, Si is the specific positions at ith fuzzy set and Fi membership degree at the positions [21].

b. The Mean of Maximum (MOM)

defuzzification method computes the average of those fuzzy conclusions or outputs that have the highest degrees. For example, the fuzzy conclusion is: the PMSM x is rotated NB. By using the MOM method, this defuzzification can be expressed as

$$MOM(NB) = \frac{\sum_{X \varepsilon T} X}{T} \qquad 3.1$$

Where T is the set of output x that has the highest degrees in the set NB. A shortcoming of the MOM method is that it does not consider the entire shape of the output membership function, and it only takes care of the points that have the highest degrees in that function. For those membership functions that have different shapes but the same highest degrees, this method will produce the same result.

c. The Height Method (HM)

This defuzzification method is valid only for the case where the output membership function is an aggregated union result of symmetrical functions. This method can be divided into two steps. First, the consequent membership function Fi can be converted into a crisp consequent x= fi where Fi is the center of gravity of Fi. Then the COG method is applied to the rules with crisp consequents, which can be expressed as

$$x = \frac{\sum_{i=1}^{m} Wifi}{\sum_{i=1}^{m} Wi} \qquad 3.2$$

Where wi is the degree to which the ith rule matches the input data. The advantage of this method is its simplicity. Therefore many neuro-fuzzy models use this defuzzification method to reduce the complex of calculations.

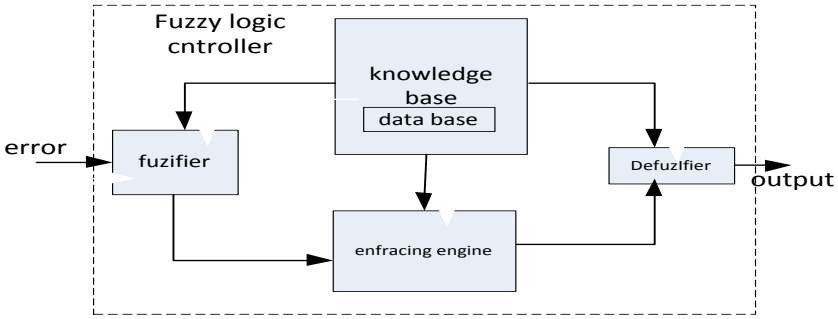


Figure 3.2: internal structure of fuzzy logic control

3.4 Fuzzy logic controller design for PMSM

The most important things in fuzzy logic control system designs are the process design of membership functions for inputs, outputs and the process design of fuzzy if-then rule knowledge base.

- To apply heuristic knowledge in the FLC, inputs, output and universe of discourse are defined first.

The Fuzzy Controller that has been developed in this thesis is show in Figure A.1 of appendix A. For the PMSM drive, speed error (e) and change in speed error (Δe) are taken as the two inputs for the fuzzy controller and one output $\Delta i_q(k)$. The input and output are described by:

$$e(k) = \omega r(k) - \omega a(k) \qquad 3.3$$

$$\Delta e(k) = e(k) - e(k-1) \qquad 3.4$$

$$i_q{}^*(k) = \Delta i_q(k) + i_q(k-1) \qquad 3.5$$

Proposed in [20], the maximum speed range that will not damage the motor is assumed to be ± 150 rad/sec. the possible error range is between -300 rad/sec and 300 rad/sec. Therefore, the universe of discourse of error (E) is defined to span between -300 rad/sec and +300 rad/sec. The

universe discourse of change in error is defined to be in range of ±3.7 rad/sec. For change in a reference current $\Delta i_q(k)$, the maximum and minimum defined values are +8 and -8 respectively.

The input and output of FLC are scaled with different coefficient GE, GCE and GCU respectively. The value of scaling factor chosen for error of speed, change error of speed and first difference of q- axis current are shown in appendix A. These scaling factors can be constant or variable and play an important role for FLC design in order to achieve a good response in both transient and steady state. In this thesis, these scaling factors are constant and are selected by trial and error.

- The second step in designing fuzzy logic controller is defining fuzzy membership function and rule.

To perform fuzzy computation, the inputs and outputs must be converted numerical or "crisp" value into linguistic forms. the term such as "Small" and "Big" are used to quantized the inputs and outputs values to linguistic values. In this thesis, the linguistic terms that used represent input and output values are defined bay five fuzzy variables as shown in table 3.I.

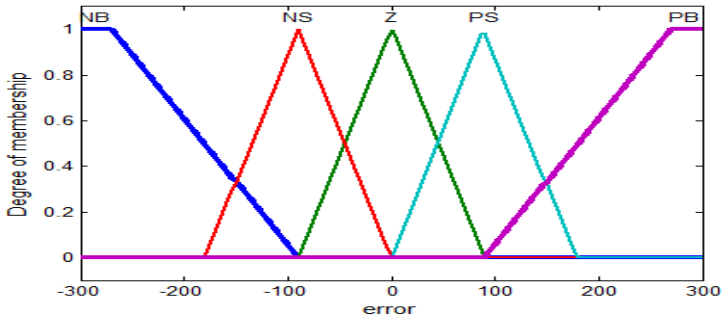
Table 3. I: Fuzzy linguistic term

Terms	Definition
NB	negative big
NS	negative small
ZE	Zero
PS	positive small
PB	positive big

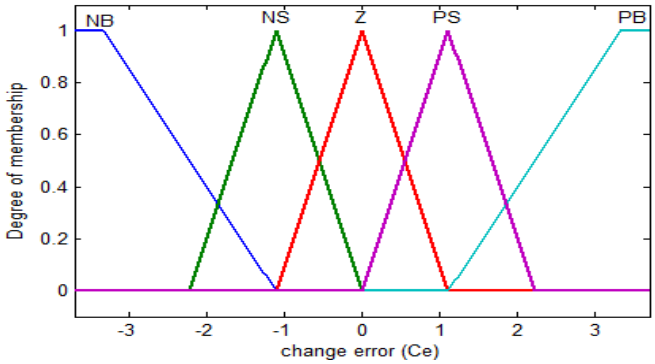
The Fuzzy membership functions can contain several fuzzy sets depending on the number of linguistic term used. Each fuzzy set represents one linguistic term. The number for indicating how much a crisp value can be a member in each fuzzy set is called a degree of membership. One crisp value can be converted to be partially in many fuzzy sets, but the membership degree in each fuzzy set may be different.

The shape of the fuzzy sets on the two extreme ends of the universe of discourse is taken as trapezoidal whereas all other intermediate fuzzy sets are triangular. The membership functions used for particular FLC are shown in figure 3.3(a)-3.3(c). All membership functions are symmetrically spaced over the universe of discourse. For performing fine-tuning to improve efficient of controller, the adjacent of each fuzzy set value should overlap at least 50%. Figure

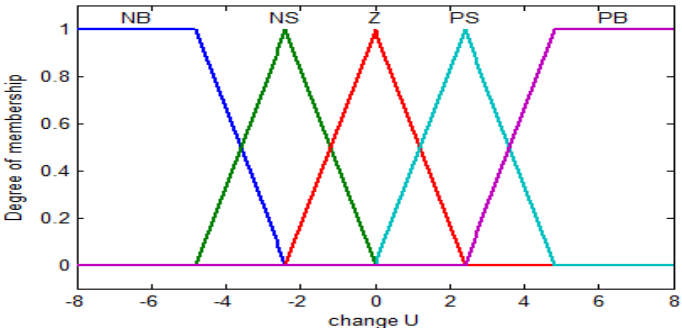
3.3(d) is show a mesh plot of relationship between error and change in error on the input side, and controller output on the output side. The plot results from a rule base with five rules.



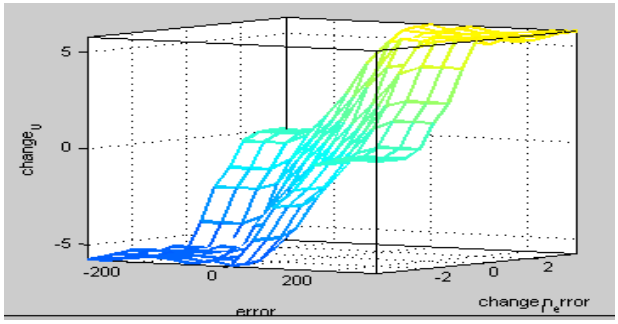
(a)



(b)



(c)



(d)

Figure 3.3: membership function and Control surface

Table 3.II: fuzzy rule of base matrix

e \ Δe	PB	PS	ZE	NS	NB
PB	PB	PB	PB	PS	ZE
PS	PB	PB	PS	ZE	NS
ZE	PB	PS	ZE	NS	NB
NS	PS	ZE	NS	NB	NB
NB	ZE	NS	NB	NB	NB

The fuzzy rule-base is given in Table 3. II. There are total 25 (number of linguistic variables to the power of input variables, $5^2 = 25$) rules to achieve desire speed trajectory. For Convenience, The Rules have been written In Matrix Form and should be Interpreted As:

IF 'speed error is NS' AND 'change in speed error is PS' THEN 'change in q-axis reference current is ZE'

IF 'speed error is PB' AND 'change in speed error is PS' THEN 'change in q-axis reference current is NB'

Referring to table 3.II above, a negative value of error implies that the process output Y is above the reference, because the error is computed as error=reference-Y (output). A positive value implies a process output is below the reference. A negative value of change in error means that the process output increases while a positive value means it decreases.

Certain regions in the table are especially interesting. The center of the table corresponds to the case where the Error is zero; the process is on the reference. Furthermore, the change in error zero here, so the process stays on the reference. This position is the stable point where the process has settled on the reference. The diagonal (orthogonal to the main diagonal) of the table is zero; those are all the pleasant states, where the process is either stable on the reference or approaching the reference. If the processes move away a little from the zero diagonal, due to noise or a disturbance, the controller will make small corrections to get it back. In case the process is far from the

reference and also moving away from it, in the upper left and lower right corners. Here the controller calls for drastic changes.

- The third procedure is adjusting fuzzy membership function and rule

In order to improve performance of the FLC, membership functions and rule can be adjusted.

3.5 Design of current controller (PI)

Proportional integral (PI) controllers are the most commonly used closed-loop controllers in the industry. PI parameters affect the following system dynamics with respect to closed-loop step response.

- Rise time - time taken for the output to rise beyond 90% of the desired level
- Overshoot - how much higher the peak level is compared to the steady-state level
- Settling time - time taken by the system to converge to its steady state
- Steady-state error - the difference between the steady-state output and the desired output

The integral parameter almost eliminates steady-state error in the output. Lower Ki values slowly push the motor speed to the expected set point, and higher Ki values can cause hunting around the set point speed. The proportional parameter provides fast response to sudden changes in load, controlling the rise time.

For decoupled torque and flux control the rotor flux and torque current have to be properly controlled, which makes current regulator very important part of the field oriented based control system. Since the PMSM is operated using field oriented control, it can be modeled like a dc motor. In PMSM System the motor has current controllers which make the current loop. The current control is performed by the comparison of the reference currents with the actual feed currents.

The parallel form structure of PI controller representation is given by:-

$$Gc(s) = Kp(s) + Ki\frac{1}{s} \qquad 3.6$$

Kp: proportional gain

Ki : integral gain

3.5.1 iq's PI controller design

Here, the design objective is to determine the PI's gains Kp and Ki (or Ti) so as to achieve a good closed loop response. The transfer function between torque component, iq and input voltage, Vq in q-axis is:

$$Gp(s) = \frac{i_q}{V_q} = \frac{1/R_s}{\tau_e s + 1} \qquad 3.7$$

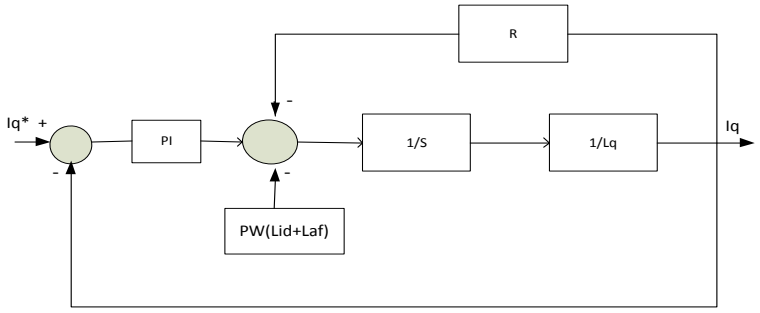


Figure 3.4 :q-axis current closed loop T(s) with PI controller block diagram

For the design of PI-controller (G_c) of current loop in q-axis, the block diagram in Figure 3.4 is used.

Open loop transfer function

$$Gopen(s) = Gc(s)Gp(s) \qquad 3.8$$

The overall closed loop transfer function between the torque component iq and its iq* is given as:

$$T(s) = \frac{i_q}{i_{q^*}} = \frac{Gc(s).Gp(s)}{1 + Gc(s).Gp(s)} \qquad 3.9$$

$$T(s) = \frac{Ki/L\,[\frac{Kp}{Ki}S + 1]}{S^2 + [\frac{R + Kp}{L}]S + \frac{Ki}{L}} \qquad 3.10$$

The equation of standard 2nd order closed loop system is given as:

$$T_s(s) = \frac{{\omega_0}^2}{S^2 + 2\xi\omega_0 S + {\omega_0}^2} \qquad 3.11$$

The PI controller can be designed by comparing the denominator of equation (3.10 &3.11) for inner loop (current loop).

$$S^2 + [\frac{R + Kp}{L}]S + \frac{Ki}{L} = S^2 + 2\xi\omega_0 S + {\omega_0}^2 \qquad 3.12$$

Where ω_0 is the natural frequency of the closed loop system (loop bandwidth) and ξ is the loop attenuation. The proportional and integral gains of the PI controller can be therefore calculated from (Equation 3.12) as:

$$Kp = 2\xi\omega_0 L - R \quad 3.13$$

$$Ki = L\omega_0^{\,2} \quad 3.14$$

After calculation and adjustment, the values of Kp and Ki were selected to be 20 and 18 respectively.

3.5.2 id's PI controller design

The transfer function between flux component, id and input voltage, Vd in d-axis current is given by equation 3.13.

$$\frac{i_d}{V_d} = \frac{1/R_s}{\tau_e s + 1} \quad 3.15$$

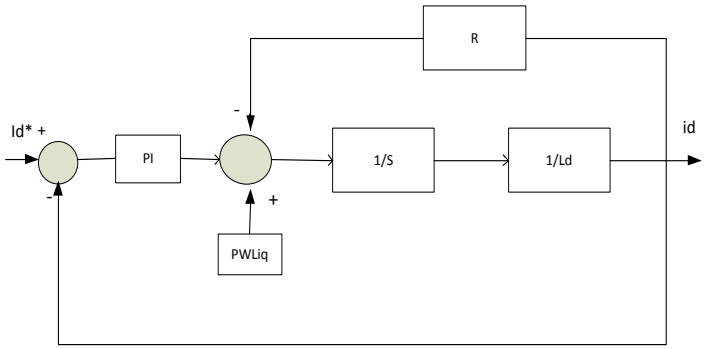


Figure 3.5: q-axis current closed loop T(S) with PI controller block diagram

In the constant torque operation region of PMSM due to the presence of the constant flux of the permanent magnet, there is no need to generate flux by means of the id current, i.e. Id=0, which decreases the stator current and increases the efficiency of the drive. In order to design the PI current regulator in d-axis it is followed the same procedure as q-axis and Kp and Ki are the same as q-axis since Lq=Ld in surface Permanent Magnet Synchronous Motor (SPMSM). For simplicity of PI controller parameter calculation, the nonlinear part of the equation is ignored.

3.6 Pulse Width Modulation (PWM)

Pulse Width Modulation technique is used to generate the required voltage or current to feed the motor or phase signals. This method is increasingly used for AC drives with the condition that the harmonic current is as small as possible and the maximum output voltage is as large as possible.

3.6.1 Sinusoidal PWM

The most popular PWM approach is the sinusoidal PWM. In this method a triangular (carrier) wave is compared to a sinusoidal wave of the desired fundamental frequency and the relative

levels of the two signals are used to determine the pulse widths and control the switching of devices in each phase leg of the inverter. Therefore, the pulse width is a sinusoidal function of the angular position of the reference signal. However, due to the variation of the sine wave reference values during a PWM period, the relation between reference values and the carrier wave is not fixed. This results in existence of harmonics in the output voltage causing undesired low-frequency torque and speed pulsations. The switching frequency is not constant and very narrow pulses may occur depending on the intersection between the carrier wave and the sine reference[21].

3.6.2 Space vector PWM

The SVPWM generates less harmonic distortion in both output voltage and current applied to the phases of an Ac motor and provides a more efficient use of the supply voltage in comparison with sinusoidal modulation techniques. SVPWM provides a constant switching frequency and therefore the switching frequency can be adjusted easily [21].

SVPWM is accomplished by rotating a reference vector around the state diagram, which is composed of six basic non-zero vectors forming a hexagon. A circle can be inscribed inside the state map and corresponds to sinusoidal operation. The area inside the inscribed circle is called the linear modulation region or under-modulation region. As seen in Figure 3.6, the area between the inside circle and outside circle of the hexagon is called the nonlinear modulation region or over-modulation region. The concepts in the operation of linear and nonlinear modulation regions depend on the modulation index, which indirectly reflects on the inverter utilization capability.

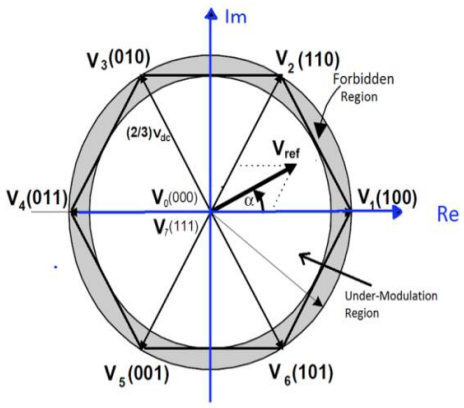


Figure 3.6: Under-modulation and Over-modulation Regions in Space Vector Representation [22].

The output voltage magnitude from a three-phase voltage source inverter can be enhanced by adding an offset to the sinusoidal modulating signal. The design of the offset is as follows:

$$Offset = \frac{-(V_{\max} + V_{\min})}{2}; V_{\max} = \{V_{an} \quad V_{bn} \quad V_{cn}\},$$
$$V_{\min} = \{V_{an} \quad V_{bn} \quad V_{cn}\}$$

The output voltage magnitude reaches the same value as that of the third-harmonic injection of PWM [22].

Implementation of a Two-Level Space Vector PWM

The SVPWM scheme is more complicated than that of the conventional SPWM. It requires the determination of a sector, calculation of vector segments, and it involves region identification based on the modulation index and calculation of switching time durations.

The procedure for implementing a two-level space vector PWM can be summarized as follows:

a. Calculate the angle θ and reference voltage vector $\vec{v}_{ref}$ based on the input voltage components.
b. Calculate the modulation index and determine if it is in the over-modulation region.
c. Find the sector in which $\vec{v}_{ref}$ lies and the adjacent space vectors of $\vec{V}$ 1 and $\vec{V}$ 2 based on the sector angle θ.
d. Find the time intervals T1 and T2 and TO based on Ts, and the angle θ
e. Determine the modulation times for the different switching states.

Angle and Reference Voltage Vector

Using the co-ordinate transformation to 2-Φ stationary reference frame in Figure 3.7, the Vα, Vβ, $\vec{v}_{ref}$ and angle (α) can be determined as follows:

$$V\alpha = V_{an} - V_{bn}\cos(60) - V_{cn}\cos(60)$$
$$V\alpha = V_{an} - \frac{1}{2}V_{bn} - \frac{1}{2}V_{cn} \qquad 3.16$$

$$V_{\beta} = 0 + V_{bn}\cos(30) - V_{cn}\cos(30)$$
$$V_{\beta} = \frac{\sqrt{3}}{2}V_{bn} - \frac{\sqrt{3}}{2}V_{cn} \qquad 3.17$$

Therefore, the above equations can be summarized in matrix form as follows.

$$\begin{bmatrix} V\alpha \\ V_\beta \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_{an} \\ V_{bn} \\ V_{cn} \end{bmatrix} \quad 3.18$$

The reference space vector voltage crossing every sector is derived as:

$$|V_{ref}| = \sqrt{v_\alpha^2 + v_\beta^2} \quad 3.19$$

The current sector in which the reference voltage vector found is determined by equation:-

$$\theta = \tan^{-1}(\frac{V_\beta}{V_\alpha}) \quad 3.20$$

, Where θ is an element of $[0,2\pi]$

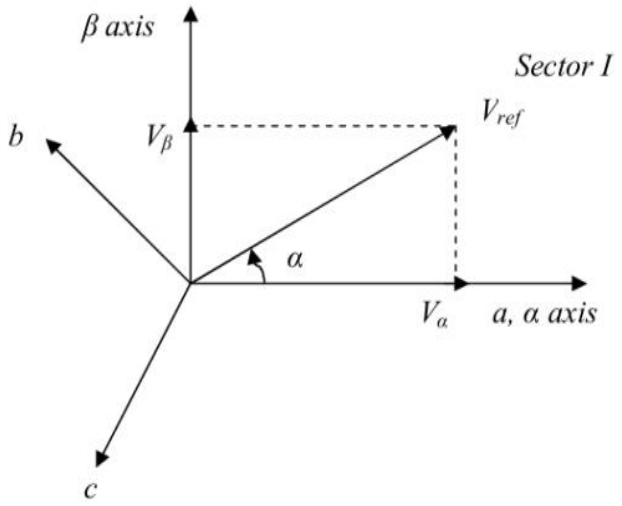


Figure 3.7: Voltage space vector and its components in (abc axis) [23].

Modulation Index of Linear Modulation

In the linear region, the rotating reference vector always remains within the hexagon. The largest output voltage magnitude is the radius of the largest circle that can be inscribed within the hexagon. This means that the linear region ends when the reference voltage is equal to the radius of the circle inscribed within the hexagon.

The ratio between the reference vector and the fundamental peak value of the square phase voltage wave $2\,V_{dc}/\pi$ is called the modulation index. In this linear region, the MI can be express as:

$$\text{MI} = \frac{\vec{v}_{ref}}{V_{\max\_six\ step}} \tag{3.21}$$

From the geometry of Figure 3.6 the maximum modulation index is obtained when $\vec{v}_{ref}$ the radius of the inscribed circle equals.

$$\vec{v}_{ref(max)} = \frac{2}{3} v_{dc} \cos\frac{\pi}{6}$$

Sector Determination

It is necessary to know in which sector the reference output lies in order to determine the switching time and sequence. The phase voltages correspond to eight switching states, six nonzero vectors and two zero vectors at the origin. Depending on the reference voltages, Vα and Vβ, the angle of the reference vector can be used to determine the sector as per Table 3.III.

Table 3.III: Sector identification

Sector	Degrees
1	$0 < \theta \leq 60°$
2	$60° < \theta \leq 120°$
3	$120° < \theta \leq 180°$
4	$180° < \theta \leq 240°$
5	$240° < \theta \leq 300°$
6	$300° < \theta \leq 360$

Time Duration

The duty cycle computation is done for each triangular sector formed by two state vectors. The magnitude of each switching state vector is $2v_{dc}/\sqrt{3}$ and the magnitude of a vector to the midpoint of the hexagon line from one vertex to another is $v_{dc}/\sqrt{3}$. The reference space vector ($\vec{v}_{ref}$) can be found with two active and one zero vector. The switching time duration at Sector 1 can be calculated as:

$$\int_0^{Tz} V_{ref}\, dt = \int_0^{T1} V_1 dt + \int_{T1}^{T1+T2} V_2 dt + \int_{T1+T2}^{Tz} v_0 dt \tag{3.22}$$

$$T_z.V_{ref} = T_1.V_1 + T_2.V_2 \quad (3.23)$$

$$T_z\,|.V_{ref}\,|.\begin{bmatrix}\cos(\alpha)\\ \sin(\alpha)\end{bmatrix} = T_1\frac{2}{3}V_{dc}\begin{bmatrix}1\\0\end{bmatrix} + T_2\frac{2}{3}V_{dc}\begin{bmatrix}\cos(\frac{\Pi}{3})\\ \sin(\frac{\Pi}{3})\end{bmatrix} \quad (3.24)$$

Where,

$$0 < \alpha \le 60^{\circ} \quad (3.25)$$

$$T_1 = T_z.a.\frac{\sin(\Pi/3 - \alpha)}{\sin(\Pi/3)} \quad (3.26)$$

$$T_2 = T_z.a.\frac{\sin(\alpha)}{\sin(\Pi/3)} \quad (3.27)$$

$$T_0 = T_z - (T_1 + T_2) \quad (3.28)$$

Where, $T_z = \frac{1}{f_z}$ and

$$a = \frac{|V_{ref}|}{\frac{2}{3}V_{dc}}$$

T1, T2 are the switching time durations of vectors V1 and V2 respectively.

T0 is the time duration of the zero vectors and

Tz is the time period for which one sector is applied.

Switching time duration at any sector is given by the following equations:-

$$T_n = \frac{\sqrt{3}T_z|v_{ref}|}{v_{dc}}\left(\sin(\frac{\Pi}{3} - \alpha + \frac{n-1}{3}\Pi)\right)$$

$$T_n = \frac{\sqrt{3}T_z|v_{ref}|}{v_{dc}}.\sin(\frac{n\Pi}{3} - \alpha) \quad (3.29)$$

$$T_{n+1} = \frac{\sqrt{3}T_z|v_{ref}|}{v_{dc}}.\sin(\alpha - \frac{n-1}{3}\Pi) \quad (3.30)$$

$$T_0 = T_z - (T_n + T_{n+1}) \quad (3.31)$$

Where , n = 1 through 6 (sector I to VI) and $0 \le \alpha \le 60$

The method used to approximate the desired stator reference voltage with only eight possible states

of switches is to combine adjacent vectors of the reference voltage and to determine the time of application of each adjacent vector as shown in Figure 3.8 for the first sector.

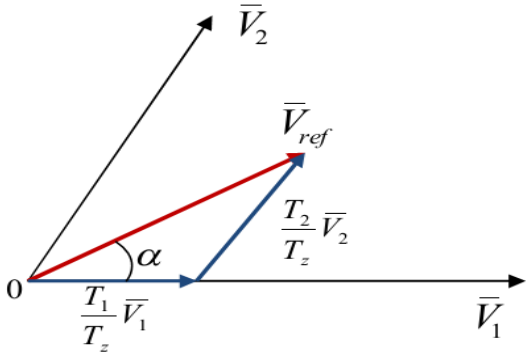


Figure 3.8: Reference voltage as a combination of adjacent vectors in sector 1[23].

Determine the switching time of each transistor (S1 to S6)

It is necessary to arrange the switching sequence so that the switching frequency of each inverter leg is minimized. There are many switching patterns that can be used to implement SVPWM. Based on the equations for T1, T2, T0, T7,and according to the principle of symmetrical PWM, the switching sequence in Table 3.IV is shown for the upper and lower switches.

Table 3.IV: Switching Time Calculation at Each Sector

Sector	*Upper switches(S1,S3,S5)*	*Lower switches(S4,S6, S2)*
1	$s1 = T1 + T2 + T0/2$ $s3 = T2 + T0/2$ $s5 = T0/2$	$s4 = T0/2$ $s6 = T1 + T0/2$ $s2 = T1 + T2 + T0/2$
2	$s1 = T1 + T0/2$ $s3 = T1 + T2 + T0/2$ $s5 = T0/2$	$s4 = T2 + T0/2$ $s6 = T0/2$ $s2 = T1 + T2 + T0/2$
3	$s1 = T0/2$ $s3 = T1 + T2 + T0/2$ $S5 = T2 + T0/2$	$s4 = T1 + T2 + T0/2$ $s6 = T0/2$ $s2 = T2 + T0/2$
4	$s1 = T0/2$ $s3 = T1 + T0/2$ $s5 = T1 + T2 + T0/2$	$s4 = T1 + T2 + T0/2$ $s6 = T1 + T0/2$ $s2 = T0/2$

5	$s1 = T2 + T0/2$ $s3 = T0/2$ $s5 = T1 + T2 + T0/2$	$s4 = T2 + T0/2$ $s6 = T1 + T2 + T0/2$ $s2 = T0/2$
6	$s1 = T1 + T2 + T0/2$ $s3 = T0/2$ $s5 = T1 + T0/2$	$s4 = T0/2$ $s6 = T1 + T2 + T0/2$ $s2 = T2 + T0/2$

3.7 Position sensor

Knowledge of the rotor flux angle is essential for accurately applying the Clarke and Park transforms. There are different types of rotor position measuring device like potentiometer, linear variable differential transformer, optical encoder, resolver, and tacho generator. The ones most commonly used for motors are encoders and resolver. Depending on the application and performance desired by the motor a position sensor with the required accuracy can be selected.

3.7.1 Optical encoder

Encoders transform rotary movement into a sequence of electrical pulses. An encoder consists of rotating disk, a light source, and a photo detector (light sensor). The disk, is mounted on the rotating shaft, has coded patterns of opaque and transparent sectors. As the disk rotates, these patterns the light emitted onto the photo detector, generating a digital pulse or output signal. It is the most popular type of encoder. There are two types of optical encoders, incremental encoder and absolute encoder.

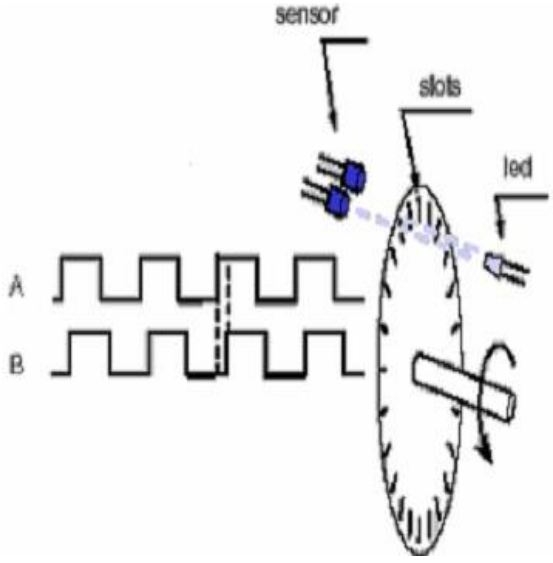


Figure 3.9 : Optical encoder [23]

Incremental encoders

Incremental encoders have good precision and are simple to implement. The disk of an incremental encoder is patterned with a single track of lines around its periphery. The disk count is defined as the number of dark/light line pairs that occur per revolution ("cycles / revolution"). The number of square wave cycles produced per one turn of the shaft is called the encoder resolution. An incremental encoder can measure the change in position, not the absolute position. Therefore, the incremental encoder cannot tell the position relative to a known reference.

Absolute encoder

The absolute encoder captures the exact position of the rotor with a precision directly related to the number of bits of the encoder. It can rotate indefinitely and even if the motor stops the position can be measured or obtained. Absolute encoders can measure the position of an object relative to a reference position at any time. The output signal of the absolute encoder presents the absolute positions in a digital code format.

CHAPTER FOUR

SYSTEM SIMULATION AND RESULTS

4.1 System simulation

The simulation for speed control of PMSM was developed in Mat Lab/Simulink R2013a by using components from the Power System's Block set as shown in Figure 4.1. The overall system simulation block includes different sub -functional blocks: FLC, PI controller blocks, Coordinate Transformation blocks and SVPWM block are main sub-functional blocks.

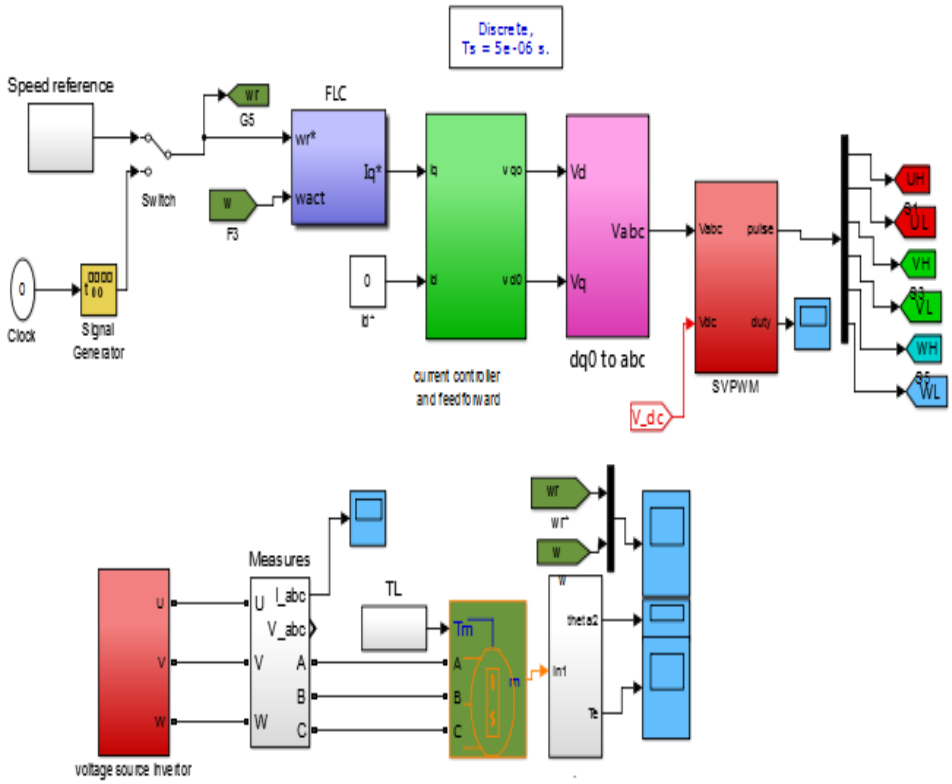


Figure 4.1: Speed control of PMSM using FL controller system simulation block diagram

4.2 Simulation result

The simulation of PMSM drive system with Fuzzy controller has been carried out using MATLAB. The speed, torque and current responses of PMSM are observed with Fuzzy controller. The performances are observed under various condition and the results are presented. The parameters of PMSM used in the simulation/experimental investigation are given in Table 4.I.

Table 4.I: motor parameter (specification)

Stator resistance: R= 1.456Ω	Stator inductance: L=0.008H
Rotor flux linkage Laf: = 0.175 V.s	Rotor inertia: J=0.06JKgm²
Friction coefficient: F= 0.001Nms	Number of pole pairs: P=3
Flux distribution type: Sinusoidal	Voltage constant: = 95.2445V/krpm
Torque constant: =7875Nm/A	

The overall Matlab/Simulink model of the PMSM drive allows simulating the behavior of the machine using proposed Fuzzy Logic controller for different operating modes. Figure 4.2 (a) – 4.2(d) are shows the speed tracking performance test using FL controller. As we see from the Figure 4.2 (a) the steady state speed is the same as that of the commanded reference speed. The system can follow the reference signal with a rise time of around 0.07 sec, settling time of 0.1 sec and has 0.41% maximum over shoot value, steady state error of 0.04rad/s and this indicates that the system has good transient and steady state response.

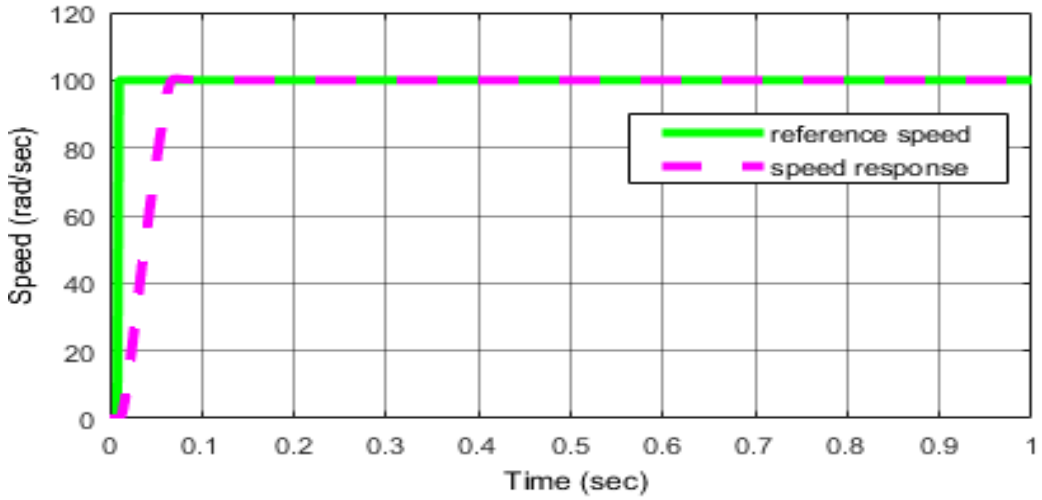


Figure 4.2 (a): Speed response of the PMSM at 100 rad/sec without load conditions

Figure 4.2 (b) and 4.2 (C) show that system can operate with different input (variable speed i.e sinusoidal followed by step input and variable step response) with good accuracy.

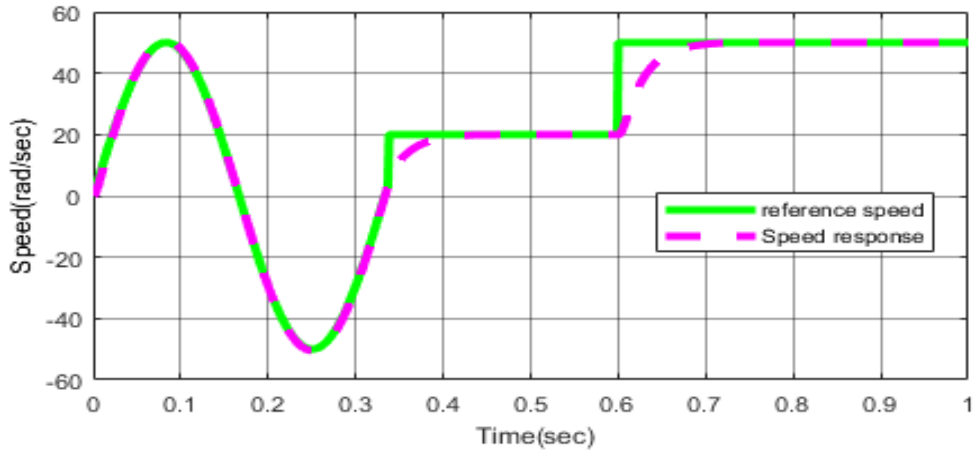


Figure 4.2(b): Variable Speed step response in rad/s with load torque of 2Nm

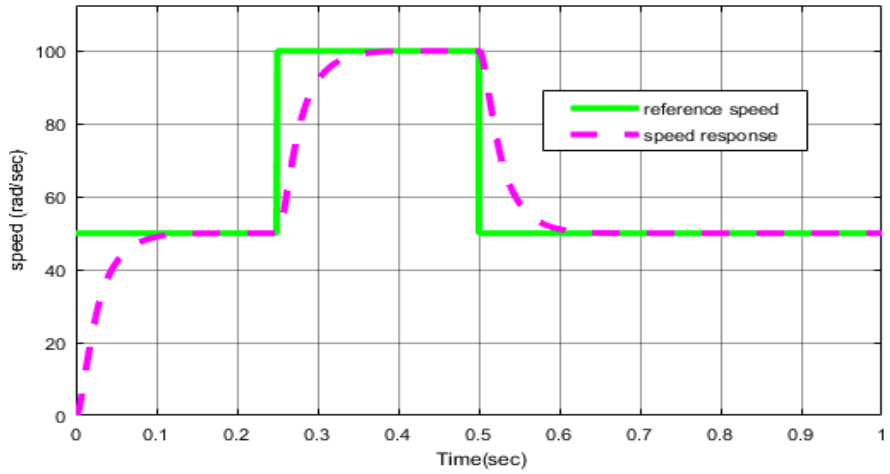


Figure 4.2(c): Variable Speed step response in rad/s without load torque

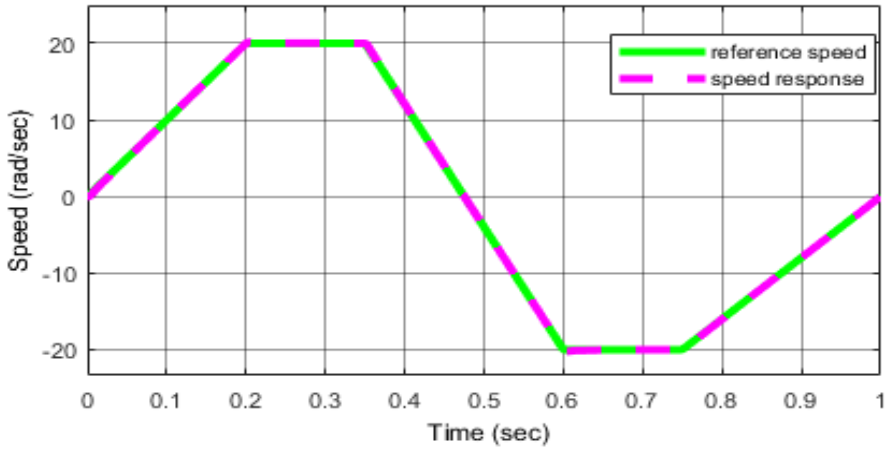


Figure 4.2(d): Variable Speed step response in rad/s without load torque using

Figure 4.2: different Speed response in rad/s under different speed input and load torque

The rotor reference speed is varied in a trapezoidal manner from zero at t = 0 to 20 rad/s then remain constant between 0.2 sec to 0.35 sec as shown in figure 4.2(d), The actual rotor speed follows the reference speed.

Figure 4.3 shows The wave form of space vector modulator outputs voltage at 100 rad/s. These continuous output space vector voltage wave form pattern obtained.

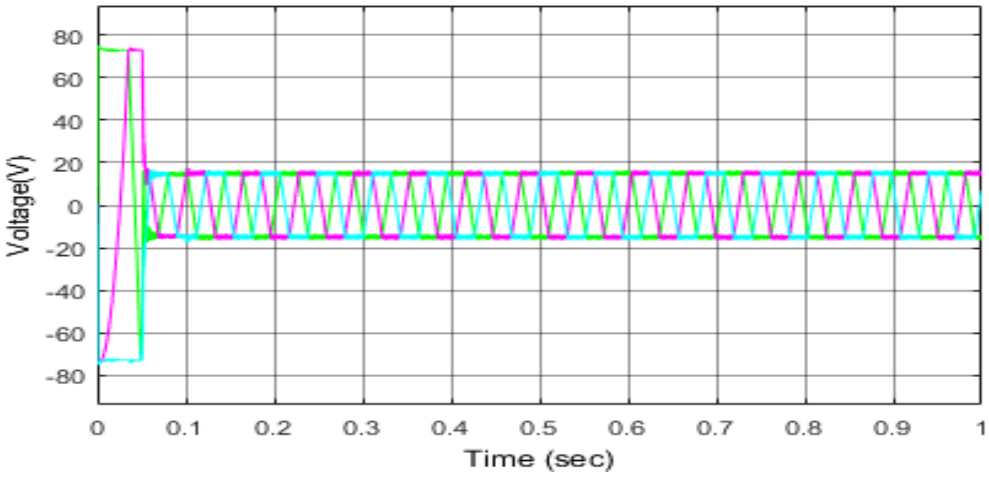


Figure 4.3: The wave form of space vector modulator outputs voltage at 100 rad/s

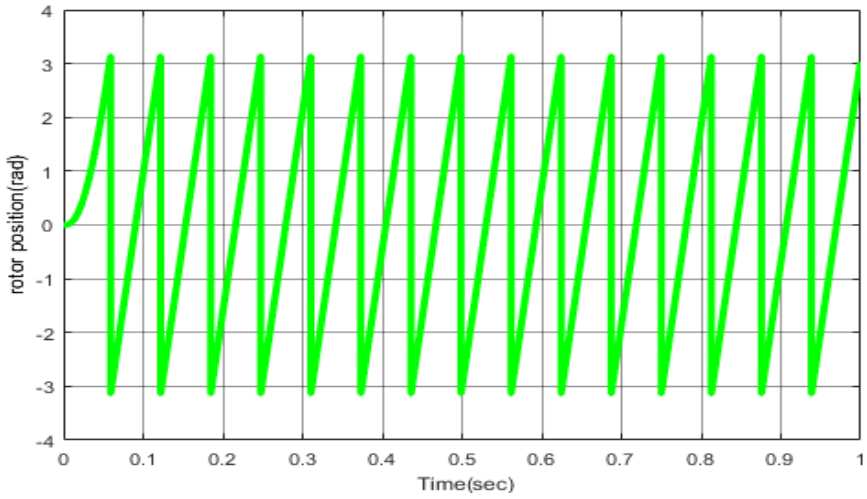


Figure 4.4: rotor position

Figure 4.4 shows the simulation of the rotor position from the beginning of the transient state when the speed is 100 rad/sec under no-load. When the steady-state speed is achieved at 0.1 sec the saw tooth periods become identical which means that the speed of the rotor is constant.

Figure 4.5: shows the developed electromagnetic torque. The starting torque until 0.07 sec is around 45 Nm which is due to primarily the acceleration of the rotor to reach to the steady speed of 100 rad/sec. The starting torque is higher when compare to the steady state value. So this generated torque is meant to support acceleration of the rotor and the friction retard without the load torque (TL=0). However, after 0.07 sec the generated torque is reduced almost to 0Nm to support only the approximately zero retard friction.

Figure 4.6 to 4.7 shows that stator current of motor and electromagnetic develop torque respectively with load torque of 2Nm at speed of 100 rad/s using FL control. It is clear that the current is non-sinusoidal at the starting and becomes sinusoidal when the motor reaches the controller command speed at steady state.

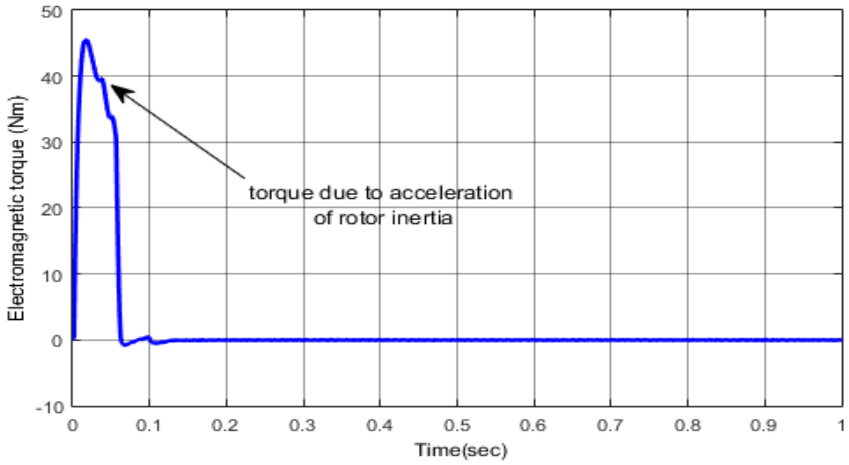


Figure 4.5: Electromagnetic torque develop while the motor is running at no -load.

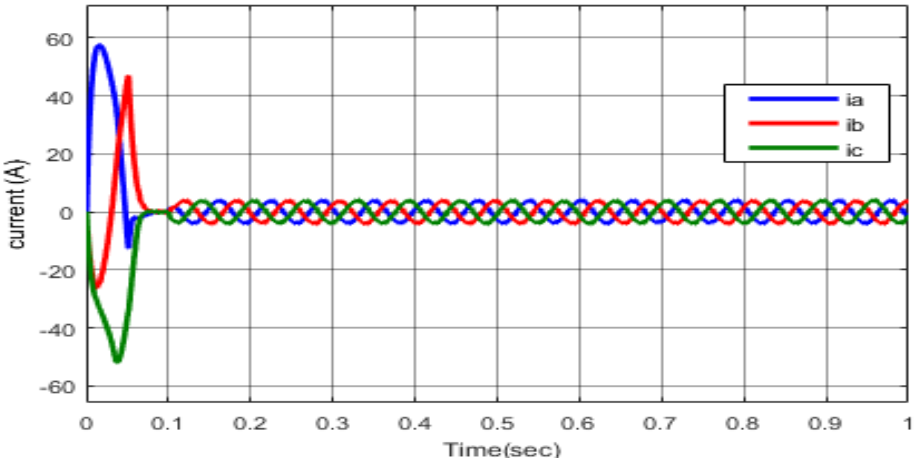


Figure 4.6: Three phase stator current at 100 rad/s With TL=2N

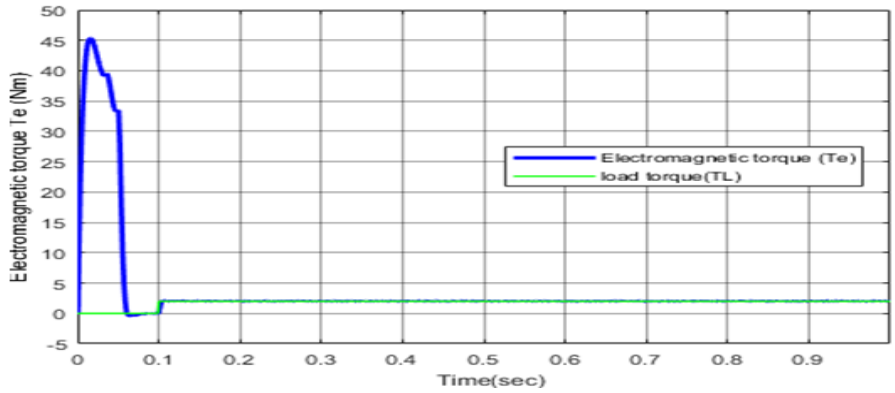


Figure 4.7: electromagnetic torques develop while the motor is running at TL=2 Nm

The motor is run in the FL speed control mode without a load at starting point. Then 10 Nm load torque is applied and removed step-wise; this is a highly hypothetical situation to test the noise removal capability of the controller. The load is suddenly applied after starting 0.2 sec and also suddenly removed at the moment of 0.4 sec from starting point. Speed, torque and phase current responses are shown in figure 4.8 (a), (b) and (c) respectively.

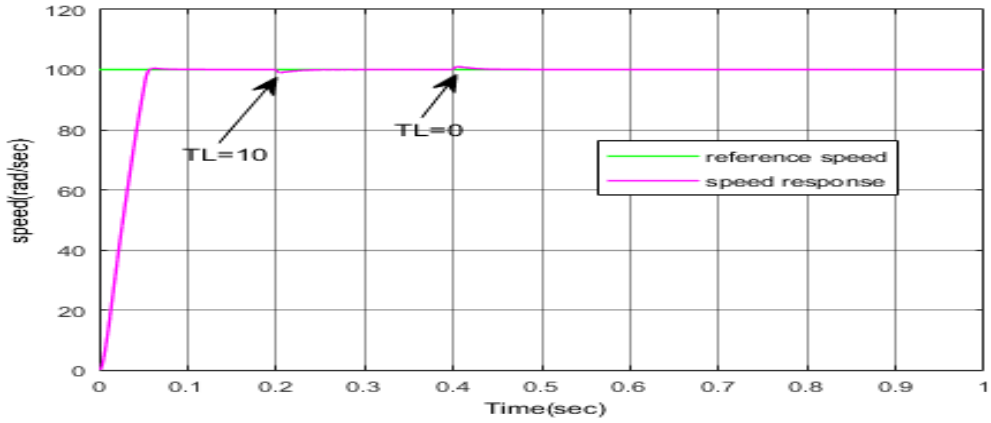


Figure 4.8 (a)

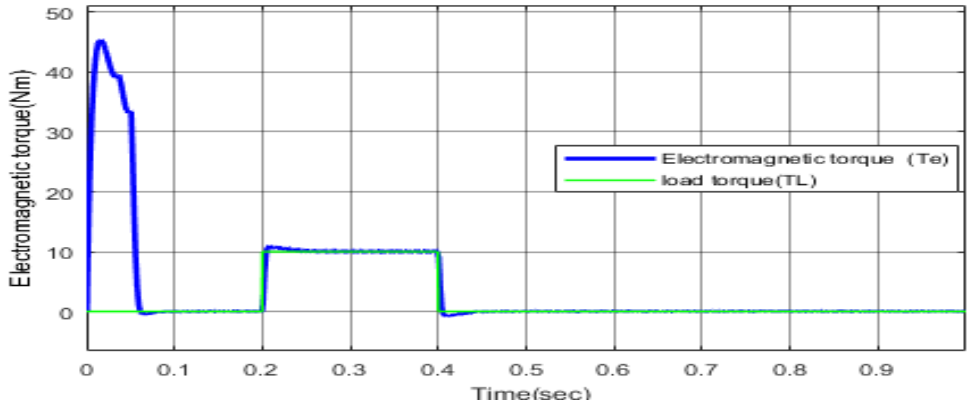


Figure 4.8 (b)

When the load torque is applied, the speed drops, then quickly returns to the command value without steady state error. Similarly, the speed rises when the load torque is removed, then returns to speed command after a short period of time. The torque response shown in Figure 4.8(b) has demonstrated the dynamic of electromagnetic torque. The motor's torque cannot change abruptly to match the load torque. This is the reason for a dip in speed shown in Figure 4.8(a). However, the motor torque increases rapidly to bring the speed back to the required value and stabilize the speed response. Stator phase current is increased during the loading period of the motor, as shown in Figure 4.8(c).

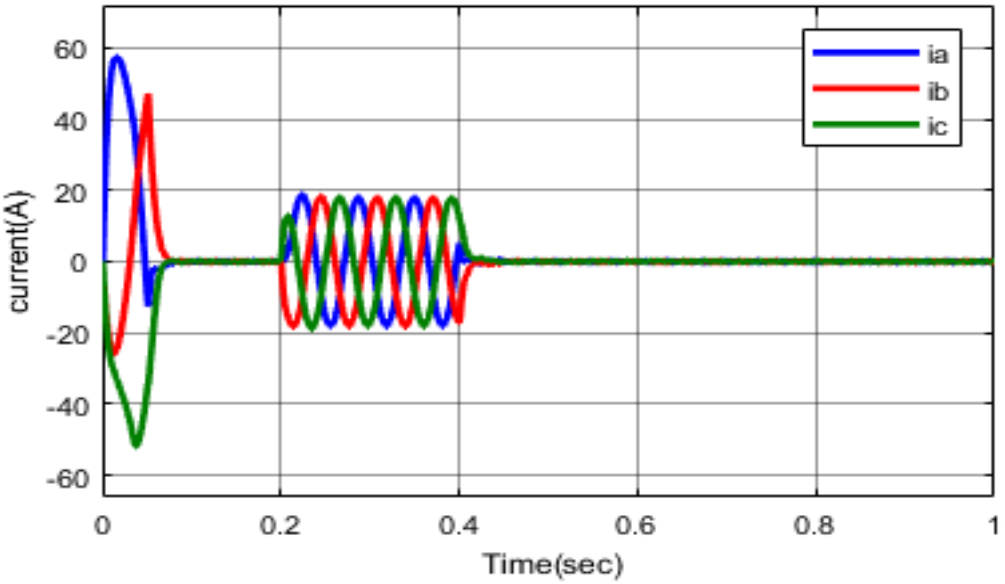
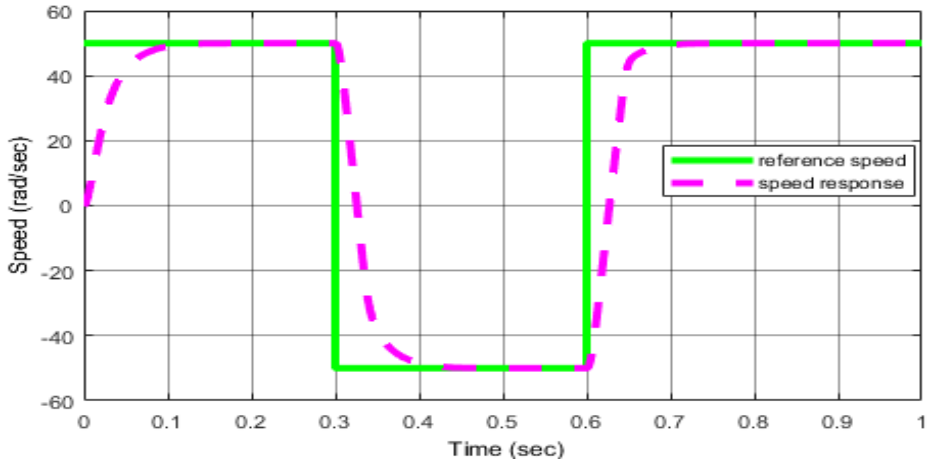


Figure 4.8 (c)

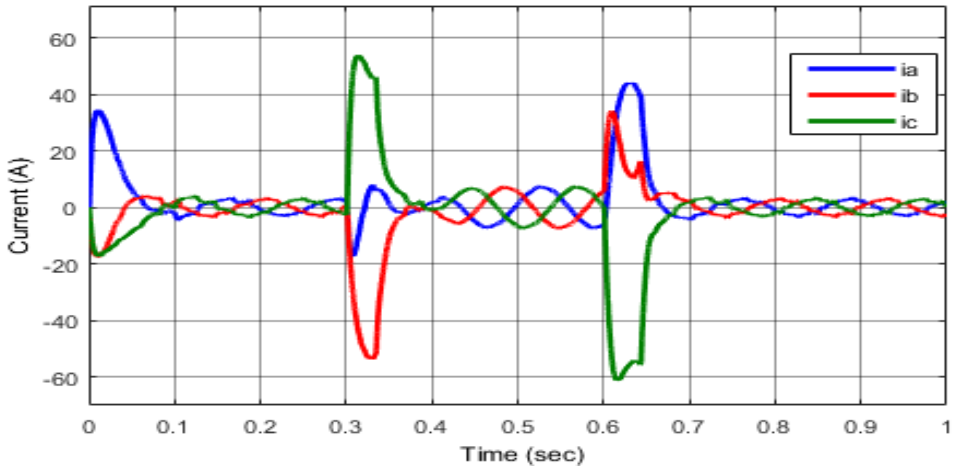
Figure 4.8: (a) Speed response in load torque variation (b) Dynamic of motor's electromagnetic torque during the application and removal of load torque and (c) 3-ϕ current

Figure 4.9 (a) and (b) shows that simulation result for speed reversal in step input and crossponding current using FL controller. The motor reference speed is changed from 50 rad/s to -50 rad/s at 0.3 second and again speed is set to 50 rad/s at 0.6 sec. The result shows that the actual speed takes arrounnd 0.1sec to follow the reference speed with good accuracy of transient response.

Figure 4.10 show the response of system speed for zero speed input. This shows that when the applied input speeds is zero the motor will immediately stop.



(a) The actual and reference speed in speed reversal with TL=2Nm



(b) The 3-ϕ stator

Figure 4.9: (a)The actual and reference speed in speed reversal with TL=2Nm (b) The 3-ϕ stator current response with speed reversal at TL=2Nm

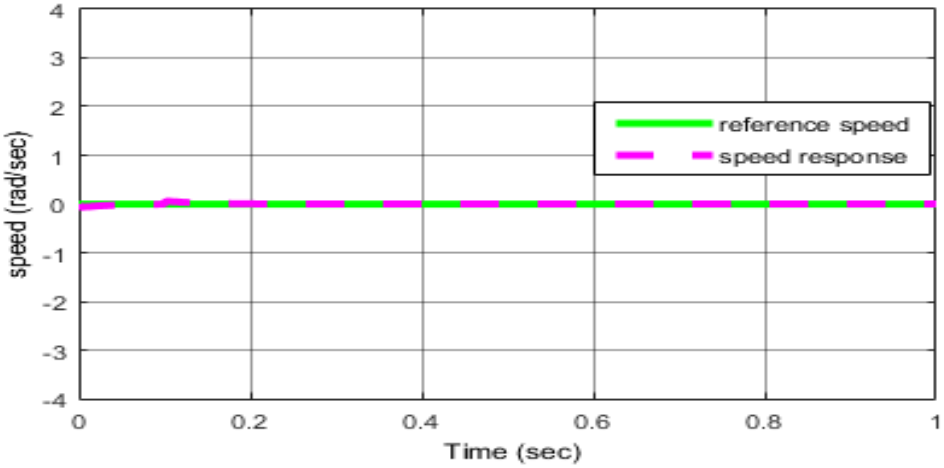
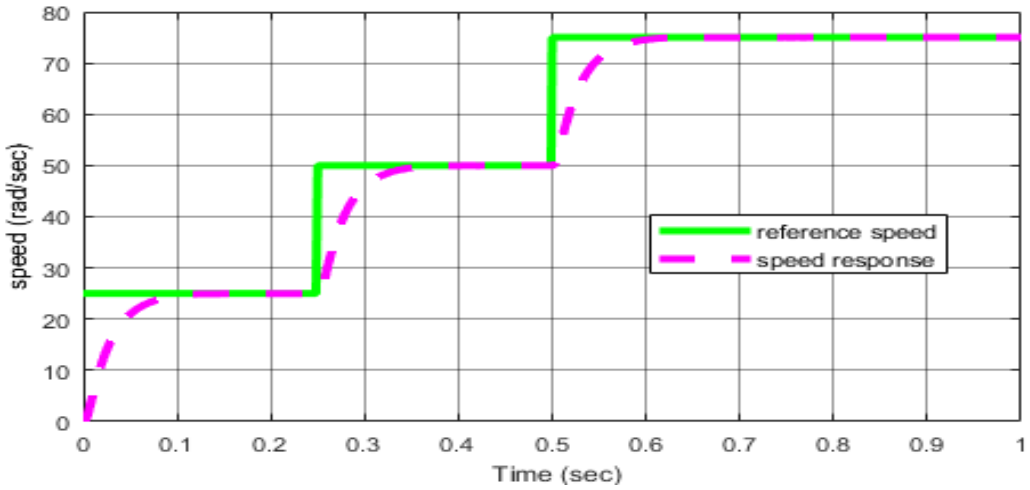
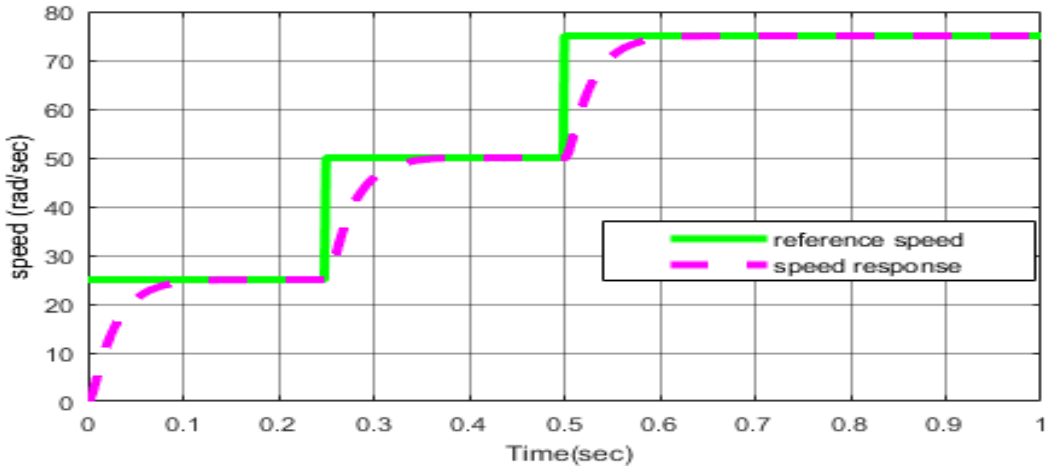


Figure 4.10: Speed response for zero speed input with no load torque using FL

figure 4.11 (a) and (b) show that simulation result for parameters variation of PMSM due to temrature variation.Specifically, the 50% inecrease and decrease in parameter of Rs and Ls at different speed referance by keeping other parameter constant. The speed input is varies in 25,50 and 75 rad/sec at time 0 , 0.25,0.5 sec respectively in both case. The result show that for all step inputs the actual speed acheve staedy state at time less than 0.1sec. Therefore ,the systetem is less sentive to parameter variations.



(a) Rs and Ls are increase by 50% its nominal value



b) When Rs and Ls are decrease by 50% its nominal value

Figure 4.11: sensitivity due to parameters variation using FLC

Comparison of the performance of the proposed fuzzy and PI controller

Figures 4.12- 4.16 shows a sample of simulation results for the operation with P1 and FL speed controllers. Comparison of the drive behavior under P1 and FL speed control is performed by overlapping speed responses.

Figure 4.12 show Speed response of the PMSM at 100 rad/s under no-load (TL =0Nm). Using PI controller Speed response has maximum percent over shoot of 5% and steady state error of 1.5 rad/s. Its also clear that FLC is fast response time compares to PI when speed is set to 100 rad/s. Using FLC the maximum overshoot is improved from 5% to 0.4% and staedy ststee error from 1.5 rad/s to 0.04 rad/s.

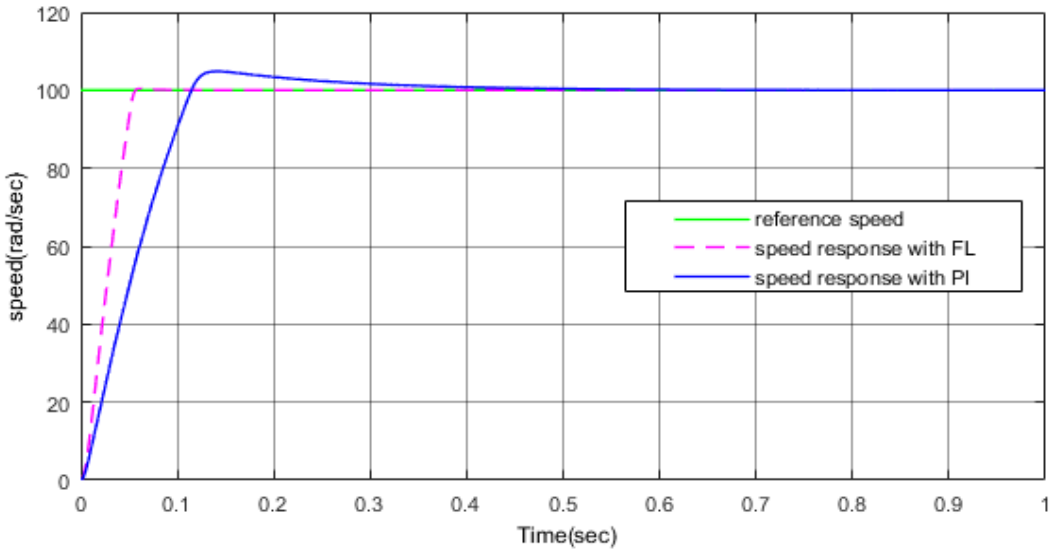


Figure 4.12: Speed response of the PMSM at 100 rad/sec using PI and FL control with TL=0Nm

Figure 4.13 show load disturbance rejection capability of each controller when load torque(TL=10 Nm) is suddenly applied at 0.4 sec and also suddenly removed at the moment of 0.7 sec from starting point under constant speed of 100 rad/s. The PMSM speed with FLC drops to 99 rad/s and rises 101 rad/s when load applied and removed respectively. But, speed with PI control drops to 96.6 rad/s and rises to 103.5 rad/s under the same conditions. The FL at that moment returns quickly to command speed, whereas the PI controller yields longer restoration time for load torque application. Therefore, maximum peak overshoot due to sudden load changes is improved from 3.5% to 1% using FLC. Figure 4.14- 4.15 shows 3-ϕ stator current and generated electromagnetic torque response of PMSM.

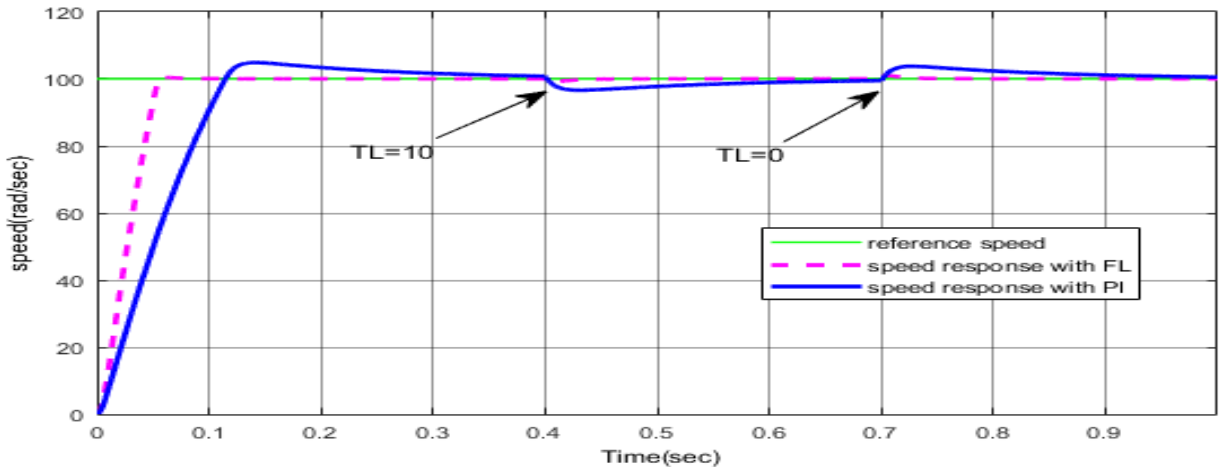


Figure 4.13: Speed response of FL and PI controller at 100rad/s under sudden load change

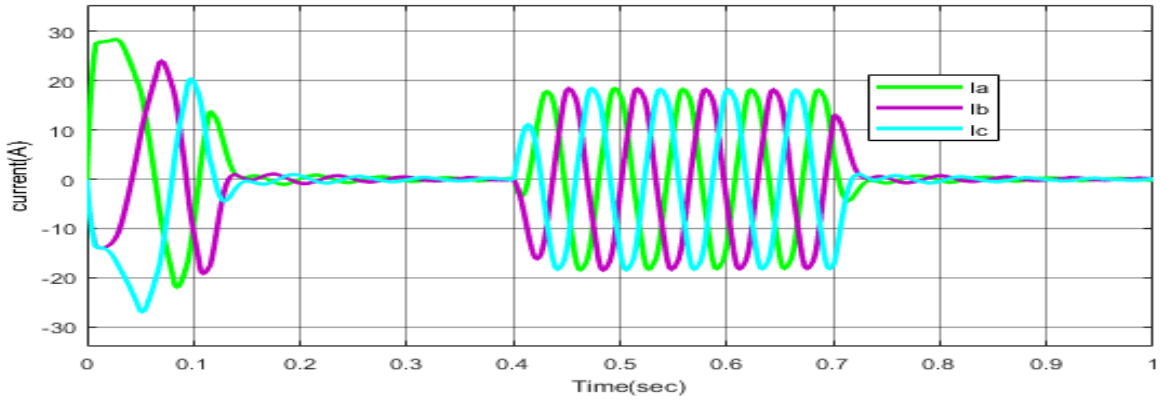


Figure 4.14:3-ϕ stator current with PI under sudden load change

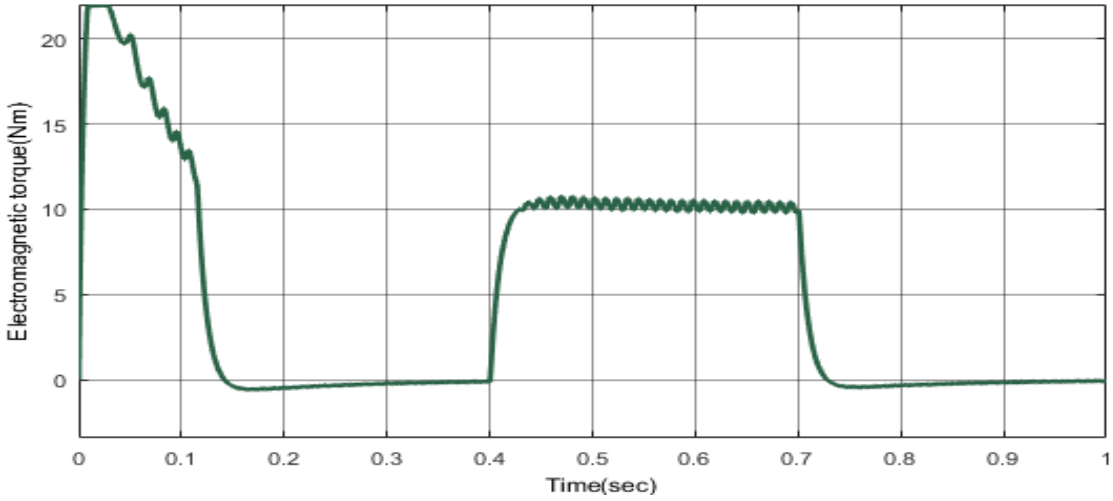


Figure 4.15: Electromagnetic torque with PI under sudden load change

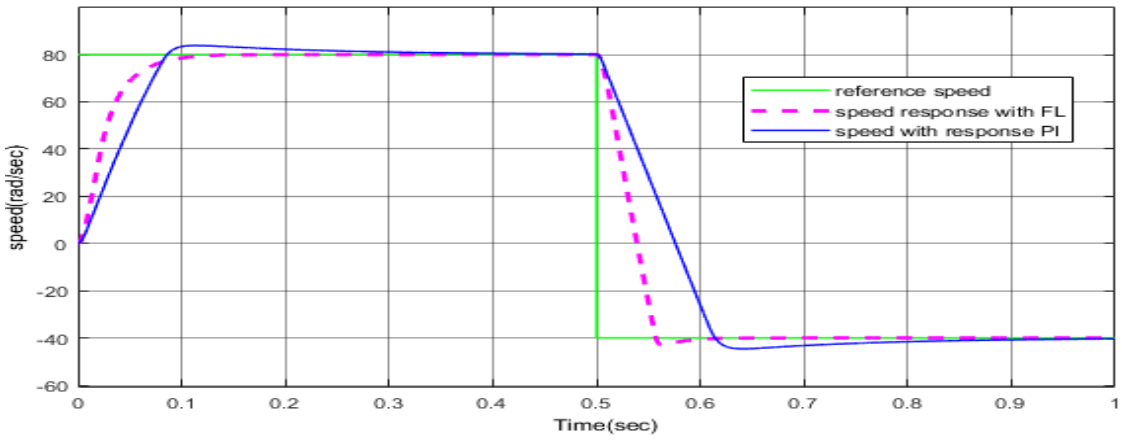


Figure 4.16: The PMSM speed response with change in speed reference With TL=0Nm

As shown in figure 4.16, the motor speed reference is 80 rad/s till 0.5 sec and then it changed to -40 rad/s. The Overshoot of PI control is 5%, while it is increased to 3% for FL control. The time needed to achieve the new steady-state is around 0.1 sec for FL and 0.2 sec using PI controllers.

CHAPTER FIVE

SYSTEM HARDWARE AND SOFTWARE ORGANIZATION

5.1 System hardware

Introduction

The performances of an permanent magnet synchronous motor are strongly dependent on its control. DSP controllers enable enhanced real time algorithms as well as sensor less control. The combination of both allows a reduction in the number of components and optimizes the design of silicon to achieve a system cost reduction [24].

A powerful processor such as a DSP controller does the following [25]:

- favors system cost reduction by an efficient control in all speed ranges implying right dimensioning of power device circuits
- performs high-level algorithms due to reduced torque ripple, resulting in lower vibration and longer life time
- enables a reduction of harmonics using enhanced algorithms, to meet easier requirements and to reduce filters cost
- Can removes speed or position sensors by the implementation of sensor less algorithms
- decreases the number of look-up tables which reduces the amount of memory required
- real-time generation of smooth near-optimal reference profiles and move trajectories, resulting in better-performing
- controls power switching inverters and generates high-resolution PWM outputs
- provides single chip control system

The overall experimental block diagram includes software and hardware. The hardware mainly contains the following modules: High voltage motor control kit, piccolo TMS320F28035 control card and PMSM as shown in Figure 5.1.

The High voltage motor control kit module include: gate driver which control three phase voltage source inverter, analog signal conditioning circuit used to sense stator current and DC bus voltage, and three phase invertor to drive the motor. The second module is piccolo TMS320F28035 control card consist: control algorithms and hardware module. Control algorithms include like ramp generator, ramp control, phase current reconstruction, coordinate transformation and SVPWM algorithm. The hardware modules also indicated in Figure 5.1.

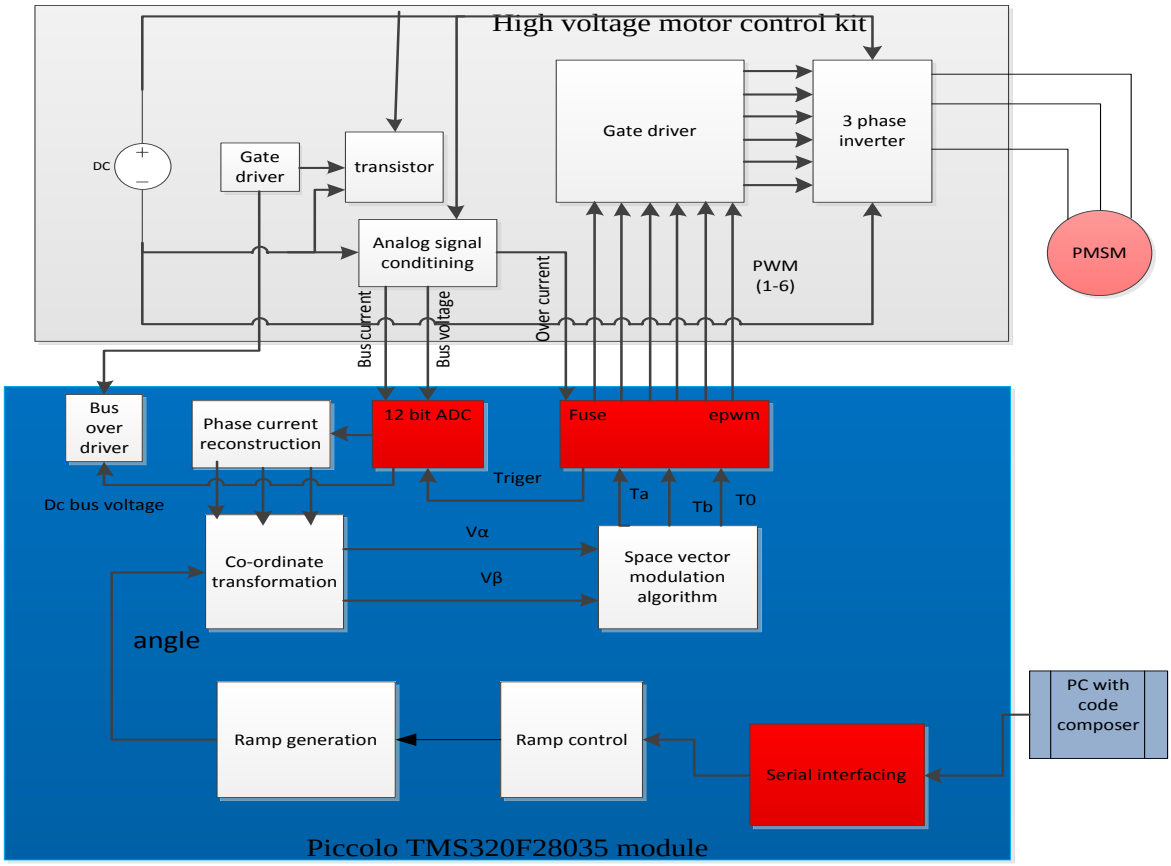


Figure 5.1: Experimental block diagram of system

Hard ware feature of HVDMCMTRPFCKIT

The high voltage motor control and PFC developer's kit is based around the C2000 32-bit microcontroller family supporting C20000 piccolo TMS320F28035 MCU. The C2000 piccolo TMS320F28035 MCU features a 60 MIPs C28xTM core with additional 60 MIPS CLA real-time control co-processors,128Kb flash memory,14 PWM channel with high resolution capability,12-bit 4.6 MSPS ADC ,capture interface, QEP interface ,serial interface and more.

The boards include an AC rectifier stage, two –phase interleaved power factor corrections stage,3-phase motor inverter stage, isolated CAN and SCI connectivity, PWM DAC for oscilloscope monitoring of system variable, and a control card DIMMY 100 slot with included based on the c2000 piccolo TMS320F28035 MCU. Figure 5.2 shown high voltage motor control and PFC (HVDMCMTRPFC) kit.

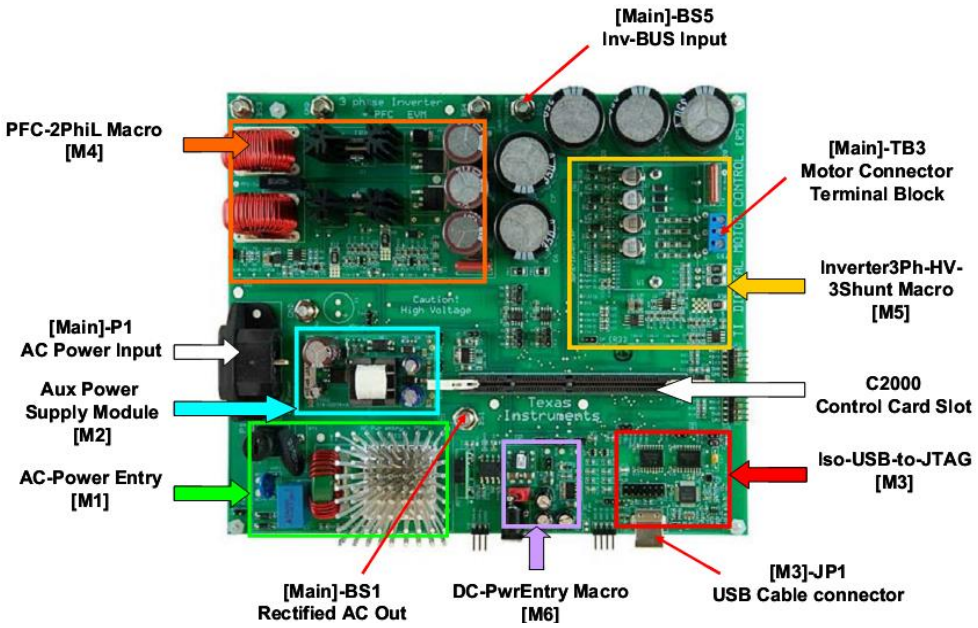


Figure 5.2: HVDMCMTRPFCKit Board Macros[27]

Key features of the TMS320F28035 Piccolo™ Microcontrollers

The Piccolo F2803x ISO control CARD can be used as a quick evaluation board for the F2803x C2000 controller as well as a noise-resistant plug-in card with Isolate JTAG emulation build in, to enable quick debug and bring up of boards requiring isolated emulator. All the key peripherals are brought out on the control CARD pins including the ADC, PWM, GPIO etc. Along with this an isolated emulator is on the control card itself, and can be connected to the host computer using a mini-USB cable to connect to the debugger. The functional block diagram of TMS320F28035 Piccolo™ Microcontrollers is shown in figure A2 appendix A.

To summarize the control CARD features:

- Small size – 90mm x 25mm (3.5" x 1")
- All GPIO, ADC and other key signals routed to gold connector fingers
- 5V input supply to the control CARD is needed for the controller. Extensive supply pin decoupling with L+C connected close to the device are provided on the control card
- Clamping diode protection at ADC input pins
- Anti-aliasing filter (noise filter) at ADC input pins
- Isolated JTAG Emulator though the mini-USB connector
- Isolated Serial connection using USB-Serial of the FTDI chip, though the mini-USB connector

- The emulator drives it's power from the mini-USB connector and connection to the controller are isolated using digital isolators.

The overall experimental system hard ware components are shown in figure 5.6.

5.2 Software organization

The overall system software is based on two modules: The initialization module and the interrupt subroutine module

Initialization module: After a DSP processor reset the initialization module performs the following task [24] as shown in figure 5.3.

- DSP setup: core, clocks, SCI,ADC, GPIO, Event manager, etc.
- Variable initialization.
- Interrupt source selection and enable: it describes enabling the DSP modules to accept interrupt and from where the interrupt is originated.
- Waiting loop: It is a loop which waits for indefinite time length until the interrupt sub module sends the interrupt starting signal

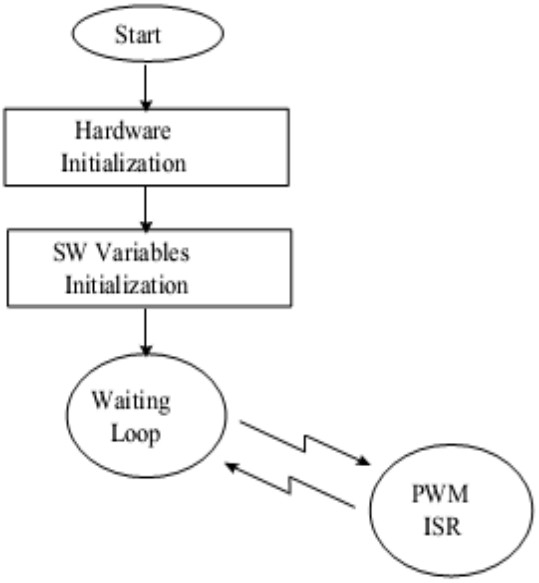


Figure 5.3: Software flowcharts

Interrupt sub routine module description: The interrupt module handles the whole field oriented control (FOC) algorithm. The complete FOC algorithm is computed within the PWM ISR, which periodically computed according to a fixed PWM period value. In this thesis, a PWM frequency of 10 KHz has been chosen. The goal of interrupt module is to update the stator voltage reference

value and to ensure the regulation of stator currents and rotor mechanical speed to their reference value.

- Reading ADC output current value and scaling up it: The stator current sensor output is connected to the ADC input of the digital signal processor. The DSP processor reads this current value from the ADC result register.
- Coordinate transformation: Different coordinate transformations for FOC are computed
- Speed setting
- SVPWM algorithm: it is an algorithm used to generate a nearly sinusoidal stator current by the inverter using q-axis and d-axis stator voltages.

The system algorithm used is shown in the figure 5.4.

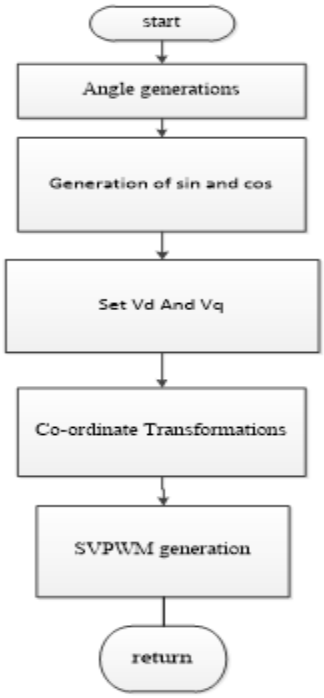


Figure 5.4: Experimental control algorithm

Code Composer Studio (CCS)

The code development is carried out on an application suite called code composer studio (CCS v6.0) made by Texas instruments. It supports all their families and variations of DSP. The key feature of the CCS is to serve the whole development chain and supports C and Assembly programming language. In Figure 5.5 the outline of the programming interface is stated. CCS consists of four main parts: project window, program window, watch window, and output window.

When choosing the compile command, potential errors will show up in the Output window. The Watch window enables tracking of variable values while running the program. The CCS communicates with DSP through the computer standard parallel port.

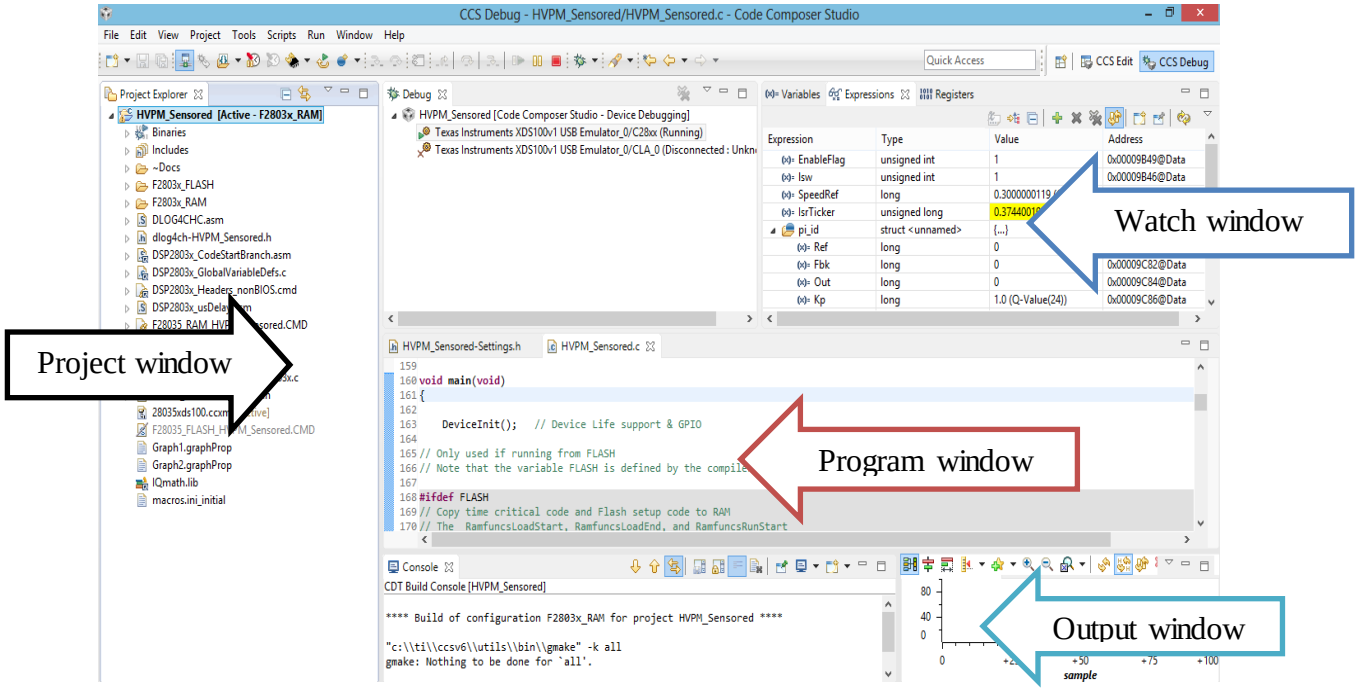


Figure 5.5: Snapshot of CCS programming interface

5.3 Experimental set up

To achieve open loop control system the main components that needed are HVMTRPFCKIT (high voltage motor control kit) with TMS320F28035 DSP control card, personal computer with Code Composer Studio (CCSv6) installed, a high voltage DC power supply, surface mounted PMSM and to see the PWM pulses, there should be digital oscilloscope. Figure 5.6 is shown the complete picture of the experimental set up that is used in this thesis.

The personal computer is used to program the system algorithm through code composer studio v6.1. The HVDMCMTRPFC kit is connected with the PC by USB cable through the FTDI driver. The source code on the PC is debugged and loaded to the memory of the piccolo TMS320F28035 control card through this cable.

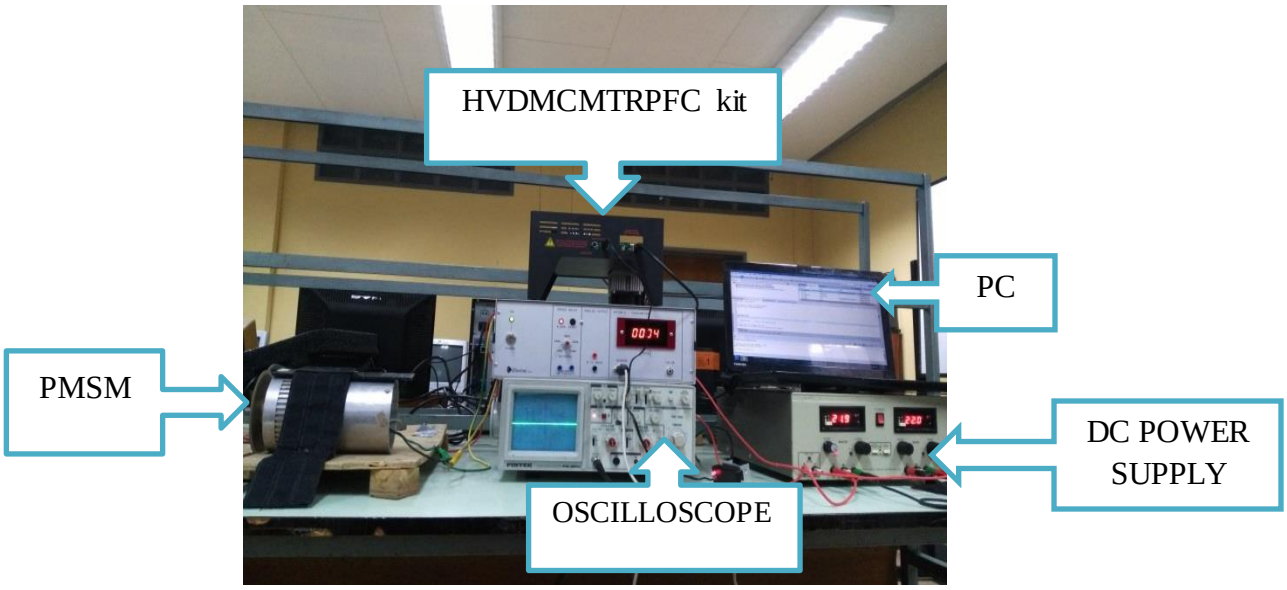


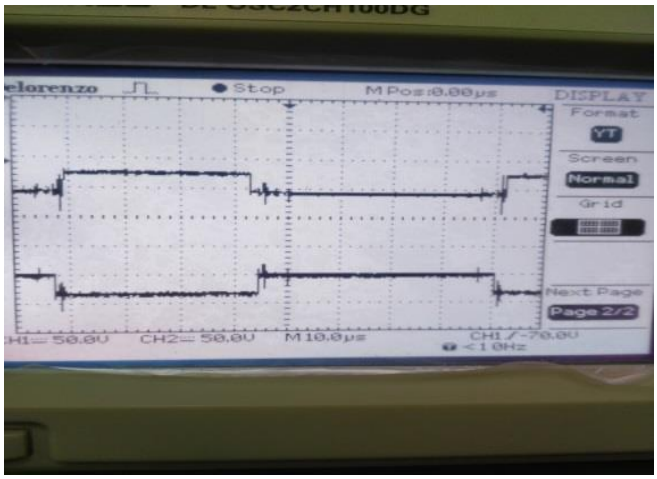
Figure5.6 :Experimental setup while the motor was running in open loop condition

5.4 Experimental result

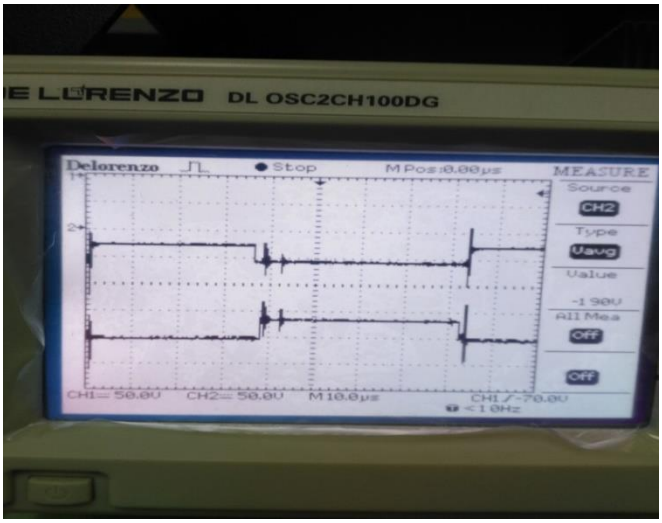
The main aim of experimental investigation is to generate the appropriate PWM signal which generates the appropriate sinusoidal voltage on the stator of the motor through the HVDMCMTRPFC kit. This makes the motor to rotate at the certain speed.

The open loop investigation is done by using a ramp angle generation technique and applying this generated angle signal to the system algorithm as a rotor position as shown in Figure 5.8.

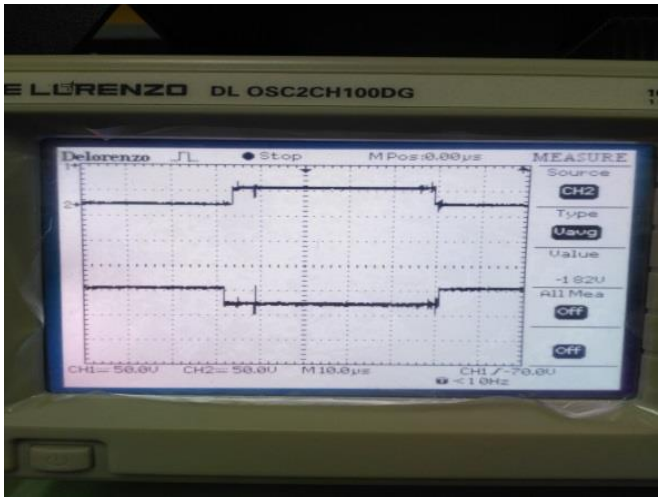
Figure 5.7 shows the wave form of PWM-H1 and PWM-1L to PWM-H3 and PWM-3L respectively on oscilloscope. From this it is clear that the space vector algorithm works correctly.



a) PWM-1H and PWM-1L



b) PWM-2H and PWM-2L



c) PWM-3H and PWM-3L

Figure 5.7: a,b and c shows the PWM output signal from the DSP while the motor is running open loop on oscilloscope

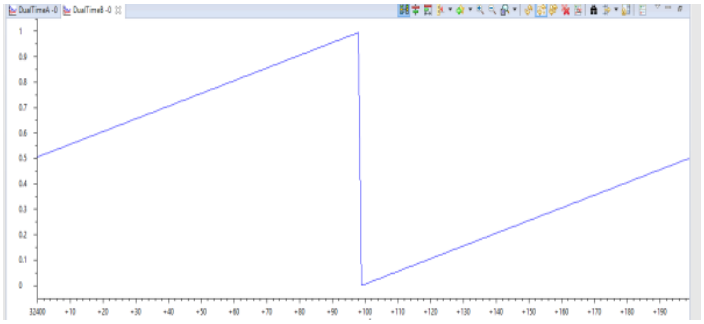


Figure 5.8: Rotor position while the motor is running in open loop

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

A fuzzy logic controller (FLC) was developed according to the PMSM drive system characteristic. The FL controller is used as the speed control in PMSM drive. The controller inputs were speed error and the speed error change rate.

The complete system has been simulated using the MATLAB/Simulink software. The Simulation result for system has been presented. The performance of proposed fuzzy logic speed controller was analyzed in terms of speed set point tracking capability, disturbance rejection, zero speed behavior, step response of drive with speed reversal and sensitivity to motor parameter variation. The simulation result at desire speed gives the maximum steady state error of 0.04 rad/s and good transient response has been achieved.

A performance comparison between the proposed fuzzy logic and conventional PI controllers has been carried out by sample simulations at no-load, with sudden load change and step change in reference speed. The simulation results show, for no-load torque overshoot is improved from 5% to 0.4% using proposed FL controller. When 10 Nm load torque is suddenly applied and removed at constant speed of 100 rad/s, overshoot is improved with FL control from 3.5% to 1%. Further, the proposed controller improves overshoot from 5% to 3% for step change in reference speed from 80 rad/s to -40 rad/s. The time needed to achieve the new steady-state for sudden load change and change in reference speed also reduced from 0.2 sec to 0.1 sec using FL controller. These simulation results confirm that FL Controller is superior to convectional PI controller.

The open loop experimental investigation is implemented using Texas Instrument, Piccolo TMS320F28035 control card.

6.2 Recommendations

The main recommendations for this thesis are summarized as:

- If all equipment is available, the developed fuzzy logic speed controller can be implemented for closed system.
- Sensor less control can also be (with no rotor position sensor) implemented.
- The thesis can also extend to operate the motor in the speed region beyond the rated speed; in field weakening region. A flux-weakening technique is applied when extension of the motor speed beyond base speed is desired.

REFERENCES

[1] Remitha K Madhu and Anna Mathew, "Matlab/Simulink Model Of Field Oriented Control of PMSM Drive Using Space Vectors," IJAET, Vol. 6, pp. 1355-1364, July 2013.

[2] Ahmad FikriHilmie, and Farrukh Hafiz Nagi, " MATLAB Simulation For Development of PWM Fuzzy Speed DSP Controller," JCET, Vol. 3, PP. 221-228, October 2013.

[3] Mohammad Abdul Mannan, Asif Islam, Mohammad Nasir Uddin, Mohammad Kamrul Hassan, Toshiaki Murata, and Junji Tamura," Fuzzy-Logic Based Speed Control of Induction Motor Considering Core Loss into Account," Intelligent Control and Automation, PP. 229-235,Auguest 2012.

[4] A.D. Ghorapade ," Comparative Study of Fuzzy Logic Based Speed Control System ," International Journal of Engineering Science and Technology ,Vol. 4, PP. 2307-2311,May 2012.

[5] Hamdihun Abdie, "Observer based Speed Control of PMSM using TMS320F2812 DSP," thesis for the degree of Master of Science in Electrical and Computer Engineering, Addis Ababa Institute of Technology (AAiT), October, 2012.

[6] Ms. Madhavi Mallam, M. Bhanu Sri , Dr.A. Guruva Reddy, "Implementation of DSP Based Fuzzy Logic Controller for Speed Control of DC Motor," International Journal of Research in Electronics & Communication Technology, Vol. 1, Issue 1, pp. 65-71,July-September 2013.

[7] Mahlet Legesse Gebresilassie, "Speed Control of Vector Controlled PMSM Drive using Fuzzy Logic-PI Controller," thesis for the degree of Master of Science in Electrical and Computer Engineering, Addis Ababa Institute of Technology (AAiT) ,August 2011.

[8] Binita Nanda and A.N Thakur, "Fuzzy Logic Based Field Oriented Control of Permanent Magnet Synchronous Motor," International Journal of Electrical, Electronics and Data Communication, Vol.3, PP.27-33, Aug.2015.

[9] G.Sakthivel, T.S. Anandhi and S.P. Natarjan,"Real Time Implementation of DSP based Fuzzy Logic Controller for Speed Control of BLDC Motor," International Journal of Computer Applications,Vol.10,PP.22-28,November 2010.

[10] R. Krishnan, "Permanent Magnet Synchronous and Brushless DC Motor Drives," Electrical and Computer Engineering Department, Virginia Tech Blacksburg, Virginia,U.S.A., CRC Press Taylor & Francis Group, 2010.

[11] Mohd Khairul Adzhar Bin Umar, " Design Of Fuzzy Controller Of IPMSM For Electric Vehicle Application," Thesis for Degree of Master of Electrical Engineering Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia ,January 2014.

[12] Maha Sabra, Bashar Khasawneh, Mohamed A. Zohdy, "Nonlinear Control of Interior PMSM Using Control Lyapunov Functions," Journal of Power and Energy Engineering, Vol.2, PP. 17-26, January 2014.

[13] Varsu Srinivas , P.V.B Kumar, " Sensor less Speed Estimation Technique For PMSM," International Journal of Advanced Technology in Engineering and Science, Vol.2, PP.690-720, December 2014.

[14] Bhagyashree Shikkewal & Vaishali Nandanwar ,"Fuzzy Logic Controller for PMSM," International Journal of Electrical and Electronics Engineering (IJEEE) ISSN (PRINT): 2231 – 5284, Vol.1,PP.73-78, Iss-3, 2012.

[15] X. T. Garc ia , B. Zigmund , A. Te rlizz I ,R. Pavlanin , L. Salvatore ,"Comparison between FOC and DTC strategies for permanent magnet synchronous motors," Advances in Electrical and Electronic Engineering, vol. 5, PP.76-81, March - June 2006.

[16] www.microsemi.com/soc/support/search/default.aspx. "Park, Inverse Park and Clarke, Inverse Clarke Transformations MSS Software Implementations User Guide," Microsemi.

[17] M.V.Ramesh, J.Amarnath, S.Kamakshaiah and M.Balakrishna ,"Field Oriented Control for Space Vector Modulation based Brushless DC Motor drive,'' IJAREEIE, Vol. 2, pp. 4231-4238, September 2013.

[18] Rathinavel Jeyabalan, "Evaluation of microcontroller architectures for PMSM control," Master of Science, Thesis in Department of Energy and Environment, Chalmers University of Technology Goteborg, 2015.

[19] Freescale semiconductor, "Design of a PMSM servo system using the 56F8357 device," document number: AN 3301.

[20] Yodyium Tipsuwan and Mo-Yuen Chow, "Fuzzy logic micro controller implementation for dc motor speed control ," department of electrical and computer engineering, North Carolina state university, USA.

[21] Haitham Abu-Rub,Atif Iqbal, Jaroslaw Guzinski "High Performance Control of Ac Drives With Matlab/Simulink Models ,published 2012.

[22] J. Holtz, W. Lotzkat, and A.M. Khambadkone, "On Continuous Control of PWM Inverters in the Over-modulation Range Including the Six-StepMode," IEEE Trans-actions on Power Electronics, Vol. 8, No. 4, pp. 546-553, October 1993.

[23] Erwan Simon, "Implementation of a Speed Field Oriented Control of three-phase PMSM Motor using TMS320F240" Texas Instruments Application Report SPRA588.

[24] Texas Instruments Incorporated, "Digital Signal Processing Solution for AC Induction Motor" Application Note: BPRA043, 1996.

[25] Texas Instruments, "Digital Signal Processing Solution for Permanent Magnet Synchronous Motor", Application Note Literature Number: BPRA044, 1997

[26] TMS320C28x DSP. CPU and Instruction Set Reference Guide,Texas Instruments, Inc. Dallas, TX, [Online]. Available: http://www.ti.com.

[27] A. Kiruthika R., Agasthiya and T. Ramesh ," Speed control of a sensored Brushless DC Motor using FLC," IJERT, Vol. 3 ,Issue 4, PP.2159- 2162, April – 2014.

[28] P.Pillay and R.krrishnan. 'Modelling , "simulation and analysis of a Permanent magnet brushless Dc motor drive". IEEE trans.Ind Applicant, Vol.26 ,pp.124-129,2002.

[29] Minarech Peter, Makys Pavol and Vittek Ján, "PI-Controllers Determination for Vector Control Motion," Department of Power Electrical Systems, Faculty of Electrical Engineering, University of Zilina.

[30] B. Zigmund, A.Te rlizzi , X. T. Garc Ia , R. Pavlanin , L. Salvatoren , "experimental Evaluation of Pi Tuning Techniques For Field Oriented Control Of Permanent Magnet Synchronous Motors," Advances in Electrical and Electronic Engineering, pp.114-119,

[31] Zhenyu Yu, DSP for Digital Motor Control, Application Report, SPRA550, June 1999

Appendix A

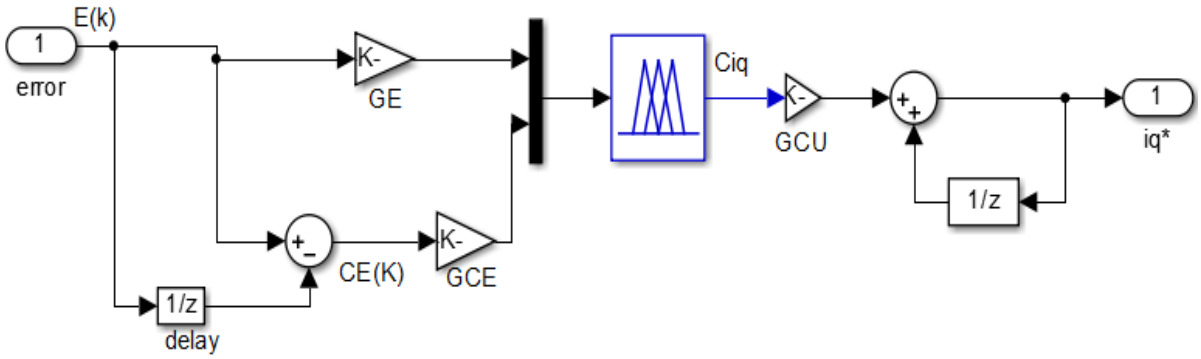


Figure A.1: The Structure Of Develop Fuzzy Speed Controller

- The selected scaling factors for FLC are: - GE=1.3, GCE=0.95 and GCU=4.

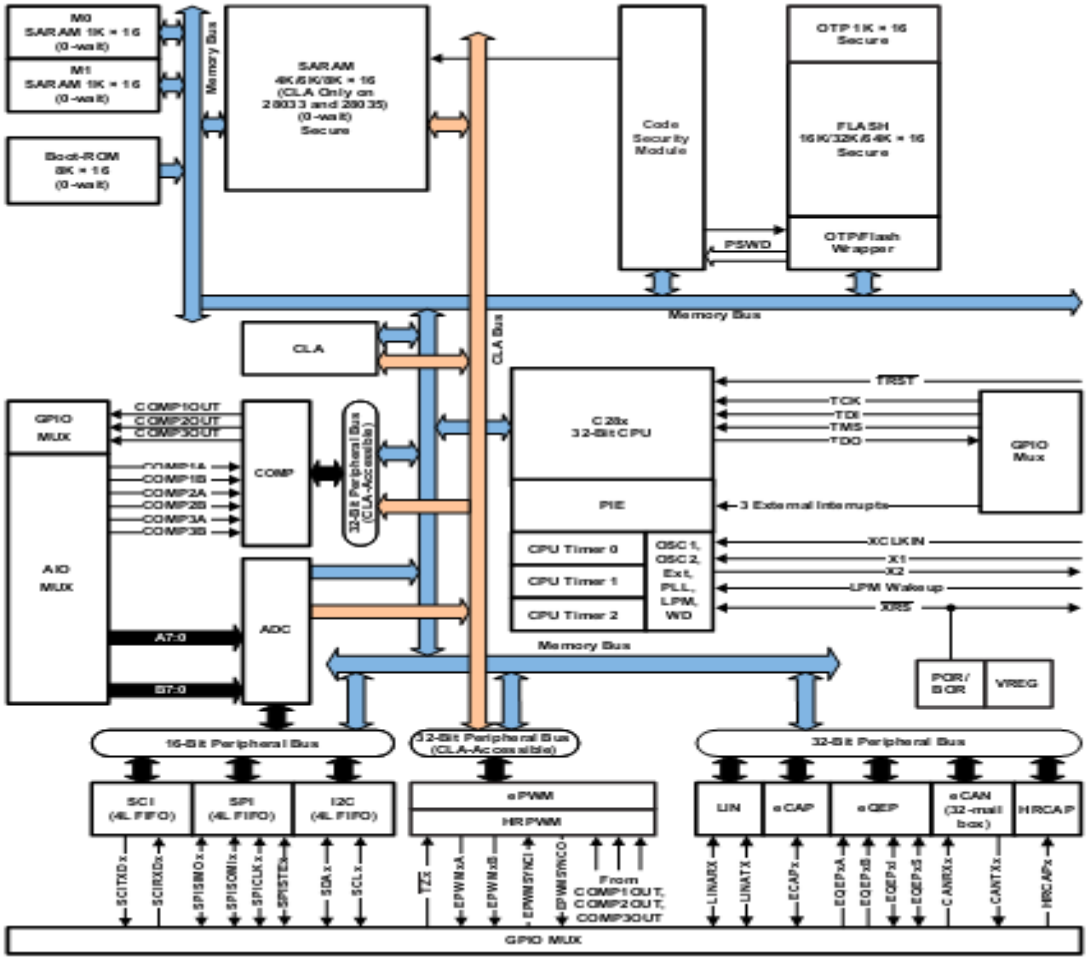


Figure A.2:Functional block diagram of TMS320 F28035 control card

Appendix B

Open loop code for motor control

```
//=================================================================================
//File name: PMSM_Sensored-Settings.H

//Description: motor parameter control setting file.
//=================================================================================
//=========== */
#ifndef PROJ_SETTINGS_H

#ifndef TRUE
#define FALSE 0
#define TRUE  1
#endif
#define PI 3.14159265358979

// Define the system frequency (MHz)
#if (DSP2803x_DEVICE_H==1)
#define SYSTEM_FREQUENCY 60
#elif (DSP280x_DEVICE_H==1)
#define SYSTEM_FREQUENCY 100
#endif

// Define the ISR frequency (kHz)
#define ISR_FREQUENCY 10
#define BASE_FREQ      50          // Base electrical frequency (Hz)
#endif
/*
//=================================================================================
//System Name:          PMSM_Sensored

//File Name:            PMSM _Sensored.C

//Description:    Primary system file for the Real Implementation of Sensored
//          Field Orientation Control for Three Phase PMSM
//=================================================================================
//=========== */
// Include header files used in the main function

#include "PeripheralHeaderIncludes.h"
#define  MATH_TYPE      IQ_MATH
#include "IQmathLib.h"
#include "HVPM_Sensored.h"
```

```
#include "HVPM_Sensored-Settings.h"
#include <math.h>

#ifdef FLASH
#pragma CODE_SECTION(MainISR,"ramfuncs");
#pragma CODE_SECTION(OffsetISR,"ramfuncs");
#endif

// Prototype statements for functions found within this file.
interrupt void MainISR(void);
interrupt void OffsetISR(void);
void DeviceInit();
void MemCopy();
void InitFlash();
void HVDMC_Protection(void);

// State Machine function prototypes
//-----------------------------------
// Alpha states
void A0(void);          //state A0
void B0(void);          //state B0
void C0(void);          //state C0

// A branch states
void A1(void);          //state A1
void A2(void);          //state A2
void A3(void);          //state A3

// B branch states
void B1(void);          //state B1
void B2(void);          //state B2
void B3(void);          //state B3

// C branch states
void C1(void);          //state C1
void C2(void);          //state C2
void C3(void);          //state C3

// Variable declarations
void (*Alpha_State_Ptr)(void);      // Base States pointer
void (*A_Task_Ptr)(void);           // State pointer A branch
void (*B_Task_Ptr)(void);           // State pointer B branch
void (*C_Task_Ptr)(void);           // State pointer C branch

// Used for running BackGround in flash, and ISR in RAM
extern Uint16 *RamfuncsLoadStart, *RamfuncsLoadEnd, *RamfuncsRunStart;
```

```
int16    VTimer0[4];          // Virtual Timers slaved off CPU Timer 0 (A events)
int16    VTimer1[4];          // Virtual Timers slaved off CPU Timer 1 (B events)
int16    VTimer2[4];          // Virtual Timers slaved off CPU Timer 2 (C events)
int16    SerialCommsTimer;

// Global variables used in this system

Uint16 OffsetFlag=0;
_iq offsetA=0;
_iq offsetB=0;
_iq offsetC=0;
_iq K1=_IQ(0.998);           //Offset filter coefficient K1: 0.05/(T+0.05);
_iq K2=_IQ(0.001999);        //Offset filter coefficient K2: T/(T+0.05);
extern _iq IQsinTable[];
extern _iq IQcosTable[];

_iq  SpeedRef = _IQ(0.03);          // speed reference

float32  T = 0.001/ISR_FREQUENCY;    // Samping period (sec), see parameter.h

Uint32 IsrTicker  = 0;
Uint16 BackTicker  = 0;
Uint16 lsw=0;
Uint16 TripFlagDMC=0;                          //Trip status
Uint16 Init_IFlag=0;

// Default ADC initialization
int ChSel[16]   = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int TrigSel[16]  = {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5};
int ACQPS[16]   = {8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8};

int16 DlogCh1  = 0;
int16 DlogCh2  = 0;
int16 DlogCh3  = 0;
int16 DlogCh4  = 0;

volatile Uint16 EnableFlag  = FALSE;
Uint16 LockRotorFlag  = FALSE;

Uint16 SpeedLoopPrescaler  = 10;     // Speed loop prescaler
Uint16 SpeedLoopCount  = 1;         // Speed loop counter

// Instance a few transform objects
CLARKE clarke1 = CLARKE_DEFAULTS;
PARK park1 = PARK_DEFAULTS;
IPARK ipark1 = IPARK_DEFAULTS;
```

```
// Instance a PWM driver instance
PWMGEN pwm1 = PWMGEN_DEFAULTS;
// Instance a PWM DAC driver instance
PWMDAC pwmdac1 = PWMDAC_DEFAULTS;
// Instance a Space Vector PWM modulator. This modulator generates a, b and c
// phases based on the d and q stationery reference frame inputs
SVGEN svgen1 = SVGEN_DEFAULTS;
// Instance a ramp controller to smoothly ramp the frequency
RMPCNTL rc1 = RMPCNTL_DEFAULTS;
//      Instance a ramp generator to simulate an Anglele
RAMPGEN rg1 = RAMPGEN_DEFAULTS;
// Create an instance of DATALOG Module
DLOG_4CH dlog = DLOG_4CH_DEFAULTS;

void main(void)
{
        DeviceInit();   // Device Life support & GPIO

#ifdef FLASH

// symbols are created by the linker. Refer to the linker files.
        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);

 // This function must reside in RAM
        InitFlash();              // Call the flash wrapper init function
#endif                          //(FLASH)

  // Waiting for enable flag set
  while (EnableFlag==FALSE)
   {
     BackTicker++;
   }

// Timing sync for background loops
// Timer period definitions found in device specific PeripheralHeaderIncludes.h
        CpuTimer0Regs.PRD.all = mSec1;          // A tasks
        CpuTimer1Regs.PRD.all = mSec5;          // B tasks
        CpuTimer2Regs.PRD.all = mSec50;         // C tasks

// Tasks State-machine init
        Alpha_State_Ptr = &A0;
        A_Task_Ptr = &A1;
        B_Task_Ptr = &B1;
        C_Task_Ptr = &C1;
```

```
// Initialize PWM module
pwm1.PeriodMax = SYSTEM_FREQUENCY*1000000*T/2;// Prescaler X1 (T1), ISR period = T
x 1
pwm1.HalfPerMax=pwm1.PeriodMax/2;
pwm1.Deadband  = 2.0*SYSTEM_FREQUENCY;  // 120 counts -> 2.0 usec for TBCLK =
SYSCLK/1
PWM_INIT_MACRO(1,2,3,pwm1)

// Initialize PWMDAC module
        pwmdac1.PeriodMax=500;    // @60Mhz, 1500->20kHz, 1000-> 30kHz, 500->60kHz
        pwmdac1.HalfPerMax=pwmdac1.PeriodMax/2;
        PWMDAC_INIT_MACRO(6,pwmdac1)     // PWM 6A,6B
        PWMDAC_INIT_MACRO(7,pwmdac1)     // PWM 7A,7B

// Initialize DATALOG module
   dlog.iptr1  = &DlogCh1;
   dlog.iptr2  = &DlogCh2;
   dlog.iptr3  = &DlogCh3;
   dlog.iptr4  = &DlogCh4;
   dlog.trig_value  = 0x1;
   dlog.size  = 0x0c8;
   dlog.prescalar  = 5;
   dlog.init(&dlog);

// Initialize ADC for DMC Kit Rev 1.1
        ChSel[0]=1;             // Dummy meas. avoid 1st sample issue Rev0 Picollo*/
        ChSel[1]=1;             // ChSelect: ADC A1-> Phase A Current
        ChSel[2]=9;             // ChSelect: ADC B1-> Phase B Current
        ChSel[3]=3;             // ChSelect: ADC A3-> Phase C Current
        ChSel[4]=15;       // ChSelect: ADC B7-> Phase A Voltage
        ChSel[5]=14;       // ChSelect: ADC B6-> Phase B Voltage
        ChSel[6]=12;       // ChSelect: ADC B4-> Phase C Voltage
        ChSel[7]=7;             // ChSelect: ADC A7-> DC Bus  Voltage

        ADC_MACRO_INIT(ChSel,TrigSel,ACQPS)

  BaseRpm = 120*(BASE_FREQ/POLES);

// Initialize the RAMPGEN module
   rg1.StepAngleMax  = _IQ(BASE_FREQ*T);


//Call HVDMC Protection function
        HVDMC_Protection();

// Reassign ISRs.
```

```
        EALLOW;      // This is needed to write to EALLOW protected registers

        PieVectTable.ADCINT1  = &OffsetISR;

// Enable PIE group 1 interrupt 1 for ADC1_INT
        PieCtrlRegs.PIEIER1.bit.INTx1  = 1;

// Enable EOC interrupt(after the 4th conversion)

        AdcRegs.ADCINTOVFCLR.bit.ADCINT1=1;
        AdcRegs.ADCINTFLGCLR.bit.ADCINT1=1;
        AdcRegs.INTSEL1N2.bit.INT1CONT=1;    //
        AdcRegs.INTSEL1N2.bit.INT1SEL=4;
        AdcRegs.INTSEL1N2.bit.INT1E=1;

// Enable CPU INT1 for ADC1_INT:
        IER |= M_INT1;

// Enable global Interrupts and higher priority real-time debug events:
        EINT;    // Enable Global interrupt INTM
        ERTM;            // Enable Global realtime interrupt DBGM

        EDIS;


// IDLE loop. Just sit and loop forever:
        for(;;)  //infinite loop
        {
                // State machine entry & exit point

        //===========================================================
                (*Alpha_State_Ptr)();  // jump to an Alpha state (A0,B0,...)

        //===========================================================
        }
} //END MAIN CODE

//===========================================================================
============
//        STATE-MACHINE SEQUENCING AND SYNCRONIZATION FOR SLOW
BACKGROUND TASKS
//===========================================================================
//--------------------------------- FRAMEWORK ---------------------------------
void A0(void)
{
```

```
        // loop rate synchronizer for A-tasks
        if(CpuTimer0Regs.TCR.bit.TIF == 1)
        {
                CpuTimer0Regs.TCR.bit.TIF = 1;    // clear flag

                //-----------------------------------------------------
                (*A_Task_Ptr)();                  // jump to an A Task (A1,A2,A3,...)
                //-----------------------------------------------------

                VTimer0[0]++;                     // virtual timer 0, instance 0 (spare)
                SerialCommsTimer++;
        }

        Alpha_State_Ptr = &B0;                    // Comment out to allow only A tasks
}

void B0(void)
{
        // loop rate synchronizer for B-tasks
        if(CpuTimer1Regs.TCR.bit.TIF == 1)
        {
                CpuTimer1Regs.TCR.bit.TIF = 1;                    // clear flag

                //-----------------------------------------------------
                (*B_Task_Ptr)();                  // jump to a B Task (B1,B2,B3,...)
                //-----------------------------------------------------
                VTimer1[0]++;                     // virtual timer 1, instance 0 (spare)
        }

        Alpha_State_Ptr = &C0;                    // Allow C state tasks
}

void C0(void)
{
        // loop rate synchronizer for C-tasks
        if(CpuTimer2Regs.TCR.bit.TIF == 1)
        {
                CpuTimer2Regs.TCR.bit.TIF = 1;                    // clear flag

                //-----------------------------------------------------
                (*C_Task_Ptr)();                  // jump to a C Task (C1,C2,C3,...)
                //-----------------------------------------------------
                VTimer2[0]++;                     //virtual timer 2, instance 0 (spare)
        }

        Alpha_State_Ptr = &A0;      // Back to State A0
```

```
}

//===========================================================================
============
//         A - TASKS (executed in every 1 msec)
//===========================================================================
============
//---------------------------------------------
void A1(void) // SPARE (not used)
//---------------------------------------------
{
            if(EPwm1Regs.TZFLG.bit.OST==0x1)
             TripFlagDMC=1;       // Trip on DMC (halt and IPM fault trip )

          A_Task_Ptr = &A2;
          //-----------------
}

//---------------------------------------------------------
void A2(void) // SPARE (not used)
//---------------------------------------------------------
{
          A_Task_Ptr = &A3;
          //-----------------
}
//-----------------------------------
void A3(void) // SPARE (not used)
//-----------------------------------
{
          A_Task_Ptr = &A1;
          //----------------
}

//===========================================================================
============
//         B - TASKS (executed in every 5 msec)
//===========================================================================
void B1(void) // Toggle  GPIO-00
//-----------------------------------
{
          B_Task_Ptr = &B2;
}
//-----------------------------------
void B2(void) //  SPARE
//-----------------------------------
{
```

```
        B_Task_Ptr = &B3;
        //----------------
}

//-----------------------------------
void B3(void) //  SPARE
//-----------------------------------
{
        //----------------
        //the next time CpuTimer1 'counter' reaches Period value go to B1
        B_Task_Ptr = &B1;
        //----------------
}
//===================================================================
============
//        C - TASKS (executed in every 50 msec)
//===================================================================
============

//--------------------------------- USER -----------------------------------

//-----------------------------------
void C1(void)           // Toggle GPIO-34
//-----------------------------------
{

        if(EPwm1Regs.TZFLG.bit.OST==0x1)                    // TripZ for PWMs is low (fault
trip)
          { TripFlagDMC=1;
          GpioDataRegs.GPBTOGGLE.bit.GPIO42 = 1;
          }

        if(GpioDataRegs.GPADAT.bit.GPIO15 == 1)             // Over Current Prot. for
Integrated Power Module is high (fault trip)
          { TripFlagDMC=1;
          GpioDataRegs.GPBTOGGLE.bit.GPIO44 = 1;
          }

        GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;          // Turn on/off LD3 on the
controlCARD
        //----------------
        //the next time CpuTimer2 'counter' reaches Period value go to C2
        C_Task_Ptr = &C2;
        //----------------
}
```

```
//-----------------------------------
void C2(void) //  SPARE
//-----------------------------------
{

        //---------------
        //the next time CpuTimer2 'counter' reaches Period value go to C3
        C_Task_Ptr = &C3;
        //---------------
}

//-----------------------------------
void C3(void) //  SPARE
//-----------------------------------
{
        //---------------
        //the next time CpuTimer2 'counter' reaches Period value go to C1
        C_Task_Ptr = &C1;
        //---------------
}

// MainISR
interrupt  void MainISR(void)
{

// Verifying the ISR
   IsrTicker++;

// ---------------------------------------------------------------------
//  Connect inputs of the RMP module and call the ramp control macro
// ---------------------------------------------------------------------
   rc1.TargetValue  = SpeedRef;
   RC_MACRO(rc1)

// --- -------------------------------------------------------------------
//  Connect inputs of the RAMP GEN module and call the ramp generator macro
// ---------------------------------------------------------------------
    rg1.Freq = rc1.SetpointValue;
    RG_MACRO(rg1)

// ---------------------------------------------------------------------
//  Measure phase currents, subtract the offset and normalize from (-0.5,+0.5) to (-1,+1).
//        Connect inputs of the CLARKE module and call the clarke transformation macro
// ---------------------------------------------------------------------
clarke1.As = _IQmpy2(_IQ12toIQ(AdcResult.ADCRESULT1)-offsetA);  // Phase A curr.
clarke1.Bs = _IQmpy2(_IQ12toIQ(AdcResult.ADCRESULT2)-offsetB);  // Phase B curr.
```

```
CLARKE_MACRO(clarke1)

// ---------------------------------------------------------------------
//  Connect inputs of the PARK module and call the park trans. macro
// ---------------------------------------------------------------------
        park1.Alpha  = clarke1.Alpha;
        park1.Beta = clarke1.Beta;
        park1.Angle  = rg1.Out;
        park1.Sine  = _IQsinPU(park1.Angle);
        park1.Cosine  = _IQcosPU(park1.Angle);
        PARK_MACRO(park1)

// ---------------------------------------------------------------------
//        Connect inputs of the INV_PARK module and call the inverse  park trans. macro
// ---------------------------------------------------------------------
   ipark1.Ds  = VdTesting;
   ipark1.Qs  = VqTesting;
   ipark1.Sine=park1.Sine;
   ipark1.Cosine=park1.Cosine;
   IPARK_MACRO(ipark1)

// ---------------------------------------------------------------------
//  Connect inputs of the SVGEN_DQ module and call the space-vector gen. macro
// ---------------------------------------------------------------------
        svgen1.Ualpha  = ipark1.Alpha;
        svgen1.Ubeta   = ipark1.Beta;
        SVGENDQ_MACRO(svgen1)

// ---------------------------------------------------------------------
//  Connect inputs of the PWM_DRV module and call the PWM signal generation macro
// ---------------------------------------------------------------------
   pwm1.MfuncC1  = svgen1.Ta;
   pwm1.MfuncC2  = svgen1.Tb;
   pwm1.MfuncC3  = svgen1.Tc;
   PWM_MACRO(1,2,3,pwm1)                                    // Calculate
the new PWM compare values

// ---------------------------------------------------------------------
//  Connect inputs of the PWMDAC module
// ---------------------------------------------------------------------
   pwmdac1.MfuncC1  = clarke1.Alpha;
   pwmdac1.MfuncC2  = clarke1.Bs;
   PWMDAC_MACRO(6,pwmdac1)  // PWMDAC 6A, 6B

   pwmdac1.MfuncC1  = svgen1.Tc;
   pwmdac1.MfuncC2  = svgen1.Tb-svgen1.Tc;
```

```
    PWMDAC_MACRO(7,pwmdac1)

// ---------------------------------------------------------------------
//  Connect inputs of the DATALOG module
// ---------------------------------------------------------------------
    DlogCh1  = (int16)_IQtoIQ15(svgen1.Ta);
    DlogCh2  = (int16)_IQtoIQ15(rg1.Out);
    DlogCh3  = (int16)_IQtoIQ15(clarke1.As);
    DlogCh4  = (int16)_IQtoIQ15(clarke1.Bs);

// ---------------------------------------------------------------------
//    Call the DATALOG update function.
// ---------------------------------------------------------------------
    dlog.update(&dlog);

// Enable more interrupts from this timer
        AdcRegs.ADCINTFLG.bit.ADCINT1=1;

// Acknowledge interrupt to recieve more interrupts from PIE group 3
        PieCtrlRegs.PIEACK.all  = PIEACK_GROUP1;

}// MainISR Ends Here

/*****************************************************************/
/*******************Offset Compensation*******************/
/*****************************************************************/
interrupt  void OffsetISR(void)
{
// Verifying the ISR
    IsrTicker++;

// DC offset measurement for ADC

    if (IsrTicker>=5000)
        {
offsetA=  _IQmpy(K1,offsetA)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT1));
        //Phase A offset
  offsetB=  _IQmpy(K1,offsetB)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT2));
        //Phase B offset
   offsetC=  _IQmpy(K1,offsetC)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT3));
        //Phase C offset
        }

        if (IsrTicker  > 20000)
        {
                EALLOW;
```

```
            PieVectTable.ADCINT1=&MainISR;
            EDIS;
        }

// Enable more interrupts from this timer
        AdcRegs.ADCINTFLG.bit.ADCINT1=1;

// Acknowledge interrupt to recieve more interrupts from PIE group 1
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

}
//*************** End of Offset Comp. ********************//
/*************************************************************/
/***************Protection Configuration*******************/
/*************************************************************/
void HVDMC_Protection(void)
{

//===================================================================
======
//Motor Control Trip Config, EPwm1,2,3
//===================================================================
======
    EALLOW;
// CPU Halt Trip
    EPwm1Regs.TZSEL.bit.CBC6=0x1;
    EPwm2Regs.TZSEL.bit.CBC6=0x1;
    EPwm3Regs.TZSEL.bit.CBC6=0x1;

    EPwm1Regs.TZSEL.bit.OSHT1   = 1; //enable TZ1 for OSHT
    EPwm2Regs.TZSEL.bit.OSHT1   = 1; //enable TZ1 for OSHT
    EPwm3Regs.TZSEL.bit.OSHT1   = 1; //enable TZ1 for OSHT

// TZA events can force EPWMxA
// TZB events can force EPWMxB

    EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
    EPwm1Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low

    EPwm2Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
    EPwm2Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low

    EPwm3Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
    EPwm3Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low

     EDIS;
```

```
    // Clear any spurious OV trip
     EPwm1Regs.TZCLR.bit.OST  = 1;
     EPwm2Regs.TZCLR.bit.OST  = 1;
     EPwm3Regs.TZCLR.bit.OST  = 1;

//*************************** End of Prot. Conf. ***************************//
}
//===========================================================================
======
// No more.
//===========================================================================
```