Dissertation Report

Master in Electrical Engineering

High Performance Position Control for Permanent Magnet Synchronous Drives

Aurelio Antonio Pesántez Palacios

Leiria, September 2017

This page was intentionally left blank

Dissertation Report

Master in Electrical Engineering

High Performance Position Control for Permanent Magnet Synchronous Drives

Aurelio Antonio Pesántez Palacios

Dissertation/Report developed under the supervision of Doctor Luís Neves, professor at the School of Technology and Management of the Polytechnic Institute of Leiria and co-supervision of Engineer Rodrigo Sempértegui, professor at the Faculty of Engineering of the University of Cuenca.

Leiria, September 2017

This page was intentionally left blank

Dedication

I would like to dedicate this dissertation

to my beloved parents

This page was intentionally left blank

Acknowledgements

I would like to thank SENESCYT, University of Cuenca and Polytechnic Institute of Leiria for the given opportunity to study and obtain my master's degree.

I also would like to thank Prof. Luís Neves and Ing. Rodrigo Sempertegui, for the guidance and support provided during the elaboration of this work.

This page was intentionally left blank

Resumo

Na conceção e teste de sistemas de controlo de acionamento elétrico, as simulações por computador fornecem uma maneira útil de verificar a correção e a eficiência de vários esquemas e algoritmos de controlo, antes de proceder à construção do sistema final, portanto, reduzindo assim o tempo de desenvolvimento e os custos associados. No entanto, a transição da fase de simulação para a implementação real deve ser tão direta quanto possível. Este documento apresenta o design e a implementação de um sistema de controlo de posição para maquinas síncronas de ímanes permanentes, incluindo uma revisão e comparação de vários trabalhos relacionados sobre sistemas de controlo não-lineares aplicados a este tipo de máquinas. O sistema geral de controlo de acionamento elétrico foi simulado e testado no software Proteus VSM que é capaz de simular a interação entre o firmware a implementar num microcontrolador e os circuitos analógicos a ele ligados. O dsPIC33FJ32MC204 foi usado como o processador de destino para implementar os algoritmos de controlo, e o modelo da máquina elétrica foi desenvolvido a partir de elementos genéricos existentes na biblioteca Proteus VSM. Como em qualquer sistema de acionamento elétrico de alto desempenho, aplicou-se um controlo orientado a fluxo magnético para alcançar uma regulação precisa de binário. O sistema de controlo completo é distribuído em três malhas de controlo, nomeadamente binário, velocidade e posição. Foram implementados e testados um sistema de controlo PID padrão e um sistema de controlo híbrido baseado em lógica difusa. Foram também simuladas a variação natural dos parâmetros do motor, como a resistência do enrolamento e o fluxo magnético. As comparações entre os dois esquemas de controlo foram realizadas para controlo de velocidade e posição, usando diferentes medidas de erro tais como o integral de erro quadrático, o integral de erro absoluto e o erro quadrático médio. Os resultados da comparação mostram um desempenho superior do controlador híbrido baseado em lógica fuzzy ao lidar com as variações dos parâmetros e reduzindo a ondulação de binário, mas os resultados são invertidos quando ocorrem distúrbios de binario periódicos. Finalmente, os controladores de velocidade foram implementados e avaliados fisicamente num banco de ensaio, embora baseado num motor DC sem escovas, com os algoritmos de controlo implementados num dsPIC30F2010, sendo os resultados consistentes com a simulação.

Palavras-chave: prototipagem de acionamentos elétricos, máquinas de íman permanente, controlo de velocidade e posição, Proteus VSM, dsPIC30F / 33F, sistema de controlo difuso.

This page was intentionally left blank

Abstract

In the design and test of electric drive control systems, computer simulations provide a useful way to verify the correctness and efficiency of various schemes and control algorithms before the final system is actually constructed, therefore, development time and associated costs are reduced. Nevertheless, the transition from the simulation stage to the actual implementation has to be as straightforward as possible. This document presents the design and implementation of a position control system for permanent magnet synchronous drives, including a review and comparison of various related works about non-linear control systems applied to this type of machine. The overall electric drive control system is simulated and tested in Proteus VSM software which is able to simulate the interaction between the firmware running on a microcontroller and analogue circuits connected to it. The dsPIC33FJ32MC204 is used as the target processor to implement the control algorithms. The electric drive model is developed using elements existing in the Proteus VSM library. As in any high performance electric drive system, field oriented control is applied to achieve accurate torque control. The complete control system is distributed in three control loops, namely torque, speed and position. A standard PID control system, and a hybrid control system based on fuzzy logic are implemented and tested. The natural variation of motor parameters, such as winding resistance and magnetic flux are also simulated. Comparisons between the two control schemes are carried out for speed and position using different error measurements, such as, integral square error, integral absolute error and root mean squared error. Comparison results show a superior performance of the hybrid fuzzy-logic-based controller when coping with parameter variations, and by reducing torque ripple, but the results are reversed when periodical torque disturbances are present. Finally, the speed controllers are implemented and evaluated physically in a testbed based on a brushless DC motor, with the control algorithms implemented on a dsPIC30F2010. The comparisons carried out for the speed controllers are consistent for both simulation and physical implementation.

Keywords: electric drives prototyping**,** permanent magnet machines, position control, Proteus VSM, dsPIC30F/33F, fuzzy control system.

This page was intentionally left blank

Table of Contents

List of Figures

List of Tables

This page was intentionally left blank

List of Acronyms

ADRC	Active disturbance rejection control
ANN	Artificial neural networks
BLDC	Brush-less direct current motor
CRPWM	Current regulated pulse width modulation
ESO	Extended state observer
PID	Proportional, integral and derivative
PMSM	Permanent magnet synchronous motor
PSO	Particle swarm optimization
SVPWM	Space vector pulse width modulation
TSM	Terminal sliding mode

This page was intentionally left blank

1. INTRODUCTION

Advances in microprocessor technologies and embedded systems have made possible implementations of complex control algorithms which require intensive math computations. Moreover, the recent development in power electronics and semiconductor devices have given a way for AC electric drives to be used instead of DC motors in high-performance applications [1].

The permanent magnet synchronous motor (PMSM) has gained an important place in applications where high-performance speed and position control are required. Characteristics such as high mass-power ratio, high torque-inertia ratio, high power density, high efficiency, reduced maintenance, etc., make the PMSM an interesting choice in applications such as industrial robots, machining tools, electric vehicles, wind turbines, etc. [2] [3] [4] [5].

A widely used control method in high-performance AC drives, is field oriented control, also known as vector control. This approach allows to control the three-phase AC machine currents through a coordinated change in the supply voltage amplitude, phase and frequency [6]. Field oriented control allows to regulate an AC electric drive in a way similar to that of the separately exited DC machine, but maintaining all the benefits of AC machines [7].

The overall performance of an electric drive will depend not only on the accuracy and speed of the control, but also on the robustness of the controller to operate correctly even if there are significant external disturbances, uncertainties in motor parameters, and lack of precise mathematical models.

Machine parameters change dynamically with temperature variations, magnetic saturations, skin effect, etc. These changes may affect the performance of an electric drive. To deal with these drawbacks, nonlinear control techniques such as fuzzy-logic controllers, sliding mode controllers, adaptive controllers, neural network controllers and hybrid controllers have been developed.

This research deals with the design and implementation of a PMSM drive control system, considering two types of controllers namely: a conventional proportional-integral-derivative (PID) controller and a hybrid controller based on fuzzy logic. The PMSM drive system is simulated and tested using the software Proteus VSM, including the implementation of controllers, coded for a dsPIC33FJ32MC204 processor.

This dissertation is organized as follows:

Chapter 2 reviews the state of the art considering some researches about nonlinear control methods for PMSM drives. A summary table of the reviewed control methods is also presented with information about the implementation platform, estimated processing power and complexity.

In Chapter 3 the PMSM control system is designed, starting with the mathematical model of the machine. Standard PID-based controllers are designed for three control loops namely current, speed and position. The design for a controller with only current and position loop (without explicit speed loop) is also presented. Hybrid controllers based on fuzzy-logic are designed for the speed and position loops. The chapter ends pointing out some practical issues about implementation of digital controllers.

Chapter 4 presents the controllers implementation, including the PMSM model developed in Proteus VSM and the required interfacing circuits for the dsPIC33FJ32MC204 processor. An algorithm for the space vector PWM implementation is also presented. Stator resistance and permanent magnet flux variations are simulated adding suitable circuits into the Proteus model. Simulation results and comparisons performed for various operating conditions are presented. The chapter finalizes giving details about the physical implementation of the speed controllers (PID and fuzzy) for a brushless DC motor (BLDC), with respective results and comparison.

Conclusions and discussions are presented in chapter 5.

2. LITERATURE REVIEW

A modern electric drive system is generally composed of several parts such as driven mechanical system, electric machine, power electronic converter, digital/analog controller, sensors/observers, and so on. With the current development in the field of power electronics and embedded systems technologies, there is a tendency of using AC machines instead of DC machines for electric drive systems [8].

Improvements in magnetic materials and motor fabrication technologies have made AC synchronous machines with permanent magnet excitation, interesting solutions for electric drive applications due to their special characteristics [9]; namely:

- There are no excitation losses which means substantial increase in efficiency.
- Higher power density than synchronous motors with electromagnetic excitation.
- High torque/inertia ratio.
- Higher magnetic flux density in the air gap.
- Better dynamic performance.
- Compact size.
- Simplification of construction and maintenance.

In high performance drive systems, precise control with fast dynamic response and good steady state response are mandatory. Furthermore, unmodeled dynamics, external disturbances, and parameter variations have to be taken into account in a high performance electric drive system.

High performance control of permanent magnet synchronous motors has been addressed by many researchers using different non-linear control techniques. Some of these non-linear control implementations and their characteristics are described below.

2.1. Active Disturbance Rejection Control

The Active Disturbance Rejection Control (ADRC) uses an estimation/cancellation strategy to cope with disturbances both internal and external. The strategy is to use the measured information of the output of the system to estimate the total disturbance (internal unmodeled dynamics and external perturbations). An extended state observer (ESO), which takes into account not only the states but also the total disturbance, is used to estimate the required states. Once the disturbance estimation is complete, it is used in the feed-back loop, cancelling the total disturbance of the system. This cancellation

leads to a time invariant linear system which can be treated with conventional control theory [10].

A position control of PMSM using an active disturbance rejection controller had been proposed by Xing-Hua Yang et al., (2010), which is a disturbance rejection technique designed without an explicit mathematical model of the plant. In this reference work, field oriented control is applied to maximize torque. PI controllers are used for the current loops and the ADRC controller is applied in the position loop. A comparison study between a standard PID controller and the proposed ADRC controller is carried out by means of computer simulation with MATLAB/Simulink software, showing that both controllers have good performance but the ADRC controller leads to a smaller error and a faster response. An experimental verification of the proposed controller is applied using a TMS320F2812 DSP chip to implement the control algorithm. Satisfactory performance is obtained when the parameters of the controller are selected according to the maximum allowable overshoot and the required speed response of the system [11].

2.2. Backstepping Control

The backstepping is a systematic and recursive design methodology for nonlinear feedback control. The main idea behind this technique is to recursively select appropriate functions of state variables as pseudo-control inputs for lower dimension subsystems. In other words, starting with a known-stable subsystem, outer subsystems can be designed expressed in terms of the inner ones. When the procedure terminates, a feedback design for the whole system is obtained. The system is designed with the desired characteristics and stability using a recursive Lyapunov-based scheme [12].

Kendouci Khadija et al., (2010), had presented a speed tracking control of PMSM using a backstepping control technique based on feedback laws and Lyapunov stability theory. In this reference work, an extended Kalman filter observer is applied to estimate the rotor speed which is feedback controlled by the backstepping control strategy. Field oriented control is applied, the d-axis current command is set to zero to maximize the torque production. The performance of the proposed backstepping sensorless speed control is evaluated by computer simulation using MATLAB/Simulink software. An experimental validation of the control algorithm is also carried out in a test-bed using the dSPACE 1103 control board. Results show that the system can track speed step references with acceptable performance, although, a large ripple is present even for considerable speeds (1000 rpm) [13].

2.3. Backstepping Control with Particle Swarm Optimization

Particle swarm optimization PSO refers to a metaheuristic that imitates the nature process of group communication to share individual experiences. PSO allows to optimize a problem starting with a possible population of solutions named “particles”. These particles are moved across the entire search space based on mathematical rules that consider the position and speed of the particles. The movement of every particle is influenced by its better local position found so far, as well as by the better global positions found by other particles as they travel through the search space [14].

Ming Yang et al., (2010), present an improved proposal for controlling the speed of a PMSM based on the backstepping technique with the addition of an adaptive weighted particle swarm optimization (PSO). The PSO is used to optimize the controller parameters, adding robustness to the control system. The proposed control strategy is tested by means of computer simulation. A comparative study between the normal backstepping-based controller and the PSO-based backstepping controller is performed. Results show that the proposed adaptive weighted PSO has better dynamic and steady state performance than the normal backstepping-based controller [15].

2.4. Model Reference Adaptive Control

The idea behind the model-reference adaptive control technique is to develop a closed loop controller with parameters that can be modified to change the response of the system. The desired response of the process to a signal input is specified as a reference model. The output of the process is compared with the output of the reference model to generate an error signal. An adaptation mechanism looks at this error and calculates the adequate parameters for the main controller in order to minimize the error. Lyapunov’s stability and Popov’s hyperstability theories are standard design methods for the control law in adaptive control systems [16].

Liu Mingji et al., (2004) had proposed a position control for PMSM using a model reference adaptive control scheme. Popov’s hyperstability theory is applied for designing the adaptive control law in the position loop. A current regulated pulse with modulation (CRPWM) technique is used for controlling the voltage source inverter that feeds the motor. A velocity observer is used to estimate the velocity of the motor shaft. The controller is implemented on an industrial computer and the results show that the output of the system follows the output of the reference model with acceptable performance despite uncertainties and parameter variations [17].

2.5. Dynamic Inversion Control

Dynamic inversion technique uses a virtual control input that allows to control a nonlinear system in a simple linear way. The strategy is to rewrite the state space system in its companion form in such way that all the nonlinear terms only affect the last state-space variable. The virtual control input is then defined in terms of the last state space elements. To clarify, consider the following nonlinear dynamic system [18]

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})u$$

$f(x)$ and $g(x)$ can be nonlinear functions. The companion form of this model will be

$$\begin{bmatrix} \dot{x_1} \\ \vdots \\ \dot{x_{n-1}} \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_n \\ b(\boldsymbol{x}) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a(\boldsymbol{x}) \end{bmatrix} u$$

As can be seen, all the nonlinear terms only affect x_n.

The virtual control input v is defined as

$$v = b(\boldsymbol{x}) + a(\boldsymbol{x})u$$

The input of the system in terms of the virtual control input is

$$u = a^{-1}(\boldsymbol{x})\big(v - b(\boldsymbol{x})\big)$$

The virtual control input v can now be used to control the entire system in a linear way.

Zhang Yaou, et al., (2010) propose a velocity control of PMSM based on the dynamic inversion approach. The controller is designed with a structure similar to a conventional PI cascade control system. The dynamic inversion is applied separately to the low frequency (velocity loop) and to the high frequency (current loop) dynamics of the system. The proposed controller is tested in terms of computer simulations using MATLAB/Simulink software. A step speed command is applied and tested for three different load torques. Acceptable performance is obtained with a good steady state response for all cases [19].

2.6. Fuzzy Logic Model Reference Adaptive Control

Basically, Fuzzy Logic is a multilevel logic that allows to define intermediate values when evaluating a statement. It is an attempt to catch and represent the human knowledge. In fuzzy logic, an affirmation can be truth for many degrees of truth, from completely true to completely false [20].

Nowadays, fuzzy logic is widely applied in control systems. A fuzzy logic controller will use fuzzy membership functions and inference mechanisms to determine the appropriate control signal. Fuzzy-logic-based controllers are usually applied together with other types of controllers/systems to achieve better performances [21].

Mohamed Kadjoudj et al., (2007), propose a model reference adaptive scheme to control the speed of a PMSM in which the adaptation mechanism uses the error and the variation of the error between the output of the reference model and the output of the system as inputs for a fuzzy-based adaptation mechanism. The main controller is also a fuzzy-logic controller whose rule base and inference mechanism are modified according to the adaptation mechanism. A comparison among the proposed fuzzy-logic adaptive controller, stand-alone fuzzy-logic controller and a fixed gain PI controller is performed using computer simulations with the MATLAB/Simulink software. Results show that the proposed fuzzy-logic adaptive controller has better performance when a repetitive step change in load torque is applied [22].

Ying-Shieh Kung and Pin-Ging Huang (2004), had presented a high-performance position controller for PMSM using a fuzzy-logic controller in the position control loop with and adaptation mechanism based on the gradient method. Vector control is applied setting the d-axis current reference to zero. PI controllers are used for the current control loop. Space vector pulse width modulation (SVPWM) is applied as a modulation technique to control the inverter. The overall system, including the adaptive controller and the SVPWM scheme are implemented in a TMS320F2812 DSP chip taking advantage of its processing power and peripheral availability. Experimental results demonstrate that in step command response and frequency command response, the rotor position rapidly tracks the prescribed dynamic response, thus, obtaining a high-performance position controller for PMSM drives [23].

2.7. Control using Artificial Neural Networks

Artificial neural networks (ANN) are non-linear processing information devices that are constituted by elementary processing devices interconnected to each other, the so-called neurons. The basic building blocks that constitute an artificial neural network are: network architecture, weights determination, and activation functions. The way in which neurons are arranged in layers and interconnection patterns, within and inside of those layers, is called the network architecture. There are many types of neural network architectures namely, feed forward, feedback, fully interconnected net, competitive net, etc. Neural networks use hidden units to enhance the internal representation of input patterns [24].

Mahmoud M. Saafan et al., (2012) present a neural network controller for PMSM. Two methods are proposed, the first one is the application of a neural-network-based controller for the speed loop and the second one is a neural-network-based torque

constant and stator resistance estimator. In both cases, the neural network is used to minimize torque ripple. The neural network weights are initially chosen randomly with small values, then, a model reference control algorithm is applied to adjust those weights to optimal values. A feed-forward neural-network architecture is applied for the parameter estimation strategy in the second method. The proposed control schemes are tested by means of computer simulation using MATLAB/Simulink software. Results show good performance with no speed overshoot. Furthermore, the obtained torque ripple values are compared with the torque ripple percentages given in other publications, observing an improved torque ripple reduction with the proposed methods [25].

2.8. Sliding Mode Control

Sliding mode control is a nonlinear control method whose purpose is to alter the dynamic of a nonlinear system applying a discontinuous control signal that force the system to "slide" along a defined state-space trajectory. The intrinsic discontinuous characteristic of the sliding mode allows a simple control that can be designed to switch between only two states (on/off) without a precise definition, therefore, adding robustness against parameter variations. A drawback of sliding mode is the introduction of high frequency oscillations around the sliding surface that strongly reduces the control performance. The aforementioned drawback is the so-called chattering effect which has to be taken into account in high performance control system implementations [26].

Fadil Hicham et al., (2015) present a velocity control of PMSM based on the sliding-mode along with a fuzzy-logic system for chattering minimization. The sliding mode controller is applied to the velocity control loop. PI controllers with decoupling compensations are applied to the current control loop. To deal with the chattering effect, a fuzzy logic controller is implemented based on the calculation of a mitigating term which will be multiplied by the discontinuous component of the sliding-mode controller. The proposed system is tested by means of computer simulations using the software tool PLECS integrated with MATLAB/Simulink. The controller is also implemented in a eZdspF28335 board using MATLAB/Simulink rapid prototyping to control an 80W PMSM. Both the sliding mode controller and the fuzzy-logic sliding mode controller were tested obtaining similar dynamic responses but verifying the effectiveness of the fuzzy-logic sliding mode controller to reduce the chattering effect [27].

Fouad Giri (2013) presents a high order terminal sliding mode control (TSM) with mechanical resonance suppressing for PMSM servo systems. TMS manifolds are designed for stator currents and load speed, respectively, to ensure convergence in finite time and obtain better tracking precision. A full-order state observer is applied to estimate

the load speed and the shaft torsion angle which cannot be measured directly. To evaluate the proposed sliding-mode based mechanical resonance suppressing method, some computer simulations with MATLAB are carried out. The step response of the motor speed is compared for three different mechanical resonance suppressing methods, namely, notch filter, acceleration feedback, and TSM control. Results show that the response of the notch filter is faster compared to other two methods. The effect of suppressing mechanical resonance using the acceleration feedback is better than the notch filter. The effect of suppressing mechanical resonance using the TMS control is the better compared to the other two methods. The speed response time of the TSM control is similar to the notch filter [28].

2.9. Hybrid Model Reference Adaptive Control

Various control techniques can be applied together in order to obtain an enhanced control performance. A hybrid position controller for PMSM conformed by three main controllers namely, an adaptive fuzzy-logic-neural-network controller, a robust controller and an auxiliary controller based on the sliding mode had been proposed by Fayez F.M. El-Sousy (2014). This complex controller is designed in order to guarantee stability and high-performance operation of the PMSM and to eliminate the need of having a prior knowledge of the constrain conditions of the system, thus, increasing the portability of the controller to other nonlinear dynamic systems. In this proposal, a decoupled current control loop is implemented with PI controllers for the d-axis and q-axis currents. To maximize torque, d-axis current reference is forced to be zero. The adaptive hybrid controller is applied to the position loop, skipping the velocity loop and thus, giving the torque reference directly from the position controller to the torque controller. The experimental validation of the proposed tracking control scheme is carried out using the MATLAB/Simulink package and a DSP control board dSPACE DS1102 based on TMS320C31 and TMS320P14 DSP chips installed in the control desktop computer. To investigate the robustness of the proposed controllers, four cases including parameter uncertainties and external load disturbances are considered. The experimental results successfully confirm that the proposed adaptive hybrid control system grants robust performance and precise dynamic response to the reference model regardless of the PMSM parameter variations and load disturbances [29].

2.10. Summary

As can be seen, there are various nonlinear control techniques which can be applied to cope with uncertainties and parameter variations in PMSM drive systems. Most of the reviewed PMSM control systems are implemented and tested with the help of MATLAB/Simulink software. Some experimental validations are also carried out in PMSM testbeds. The real-world implementations are performed using MALTAB/Simulink code generation capability in some cases, and direct coding in some others. For all the reviewed real-world implementations, high-end powerful hardware is used to execute the control algorithms.

Although there is not a direct way to determine the relationship between a specific control algorithm and the amount of processing power required to execute it, having a way to experiment and estimate that relationship will be helpful for selecting the hardware and the control strategy which better fit to a specific application.

Furthermore, it is not always practical/possible to test the controllers within a real-world testbed. For instance, if different hardware platforms need to be considered/compared, or if a change in hardware is required, a computer simulation of these scenarios will reduce costs and implementation time. Nevertheless, a straightforward transition from the computer simulation to the real-world implementation is required.

Another point to be noted in the reviewed literature is the PMSM model used for the simulations. In all cases, the PMSM model considers balanced stator windings with sinusoidal distributed magnetomotive force, sinusoidal inductance vs position, and neglects saturation and parameter changes.

To validate a control algorithm in terms of computer simulation, the model used to represent the plant/process must be as accurate as possible to obtain consistent results. The PMSM model used in the reviewed literature has sufficient characteristics for most initial designs, but the controller will require further adjustment/calibration to be performed in the real-world implementation. Nevertheless, a more realistic PMSM model will be required to catch the effects and performance of controllers implemented by means of computer simulation.

A summary of the reviewed non-linear control methods for PMSM drive systems is presented in table 2.1, including an estimation of the relative complexity and power processing capability required to implement the controller in each case. The processing power estimations are based on the number of calculations that need to be performed, paying special attention to divisions. The estimation of the implementation complexity

considers the hardware and software used to implement each specific controller. For instance, an implementation using a control board with MATLAB/Simulink support for code generation, will be easier than an implementation in a stand-alone controller via hand-written firmware.

Author(s)	*Control strategy*	*Simulation platform*	*Hardware*	*Implementation complexity*	*Processing power required*
Xing-Hua Yang et al., (2010)	Active disturbance rejection	MATLAB/Simulink	TMS320F2812 DSP (150MIPS, 32-bit CPU, fixed point arithmetic, motor control peripherals, 12-bit ADC @ 12.5 MSPS)	Medium	Medium
Kendouci Khadija et al., (2010)	Backstepping control	MATLAB/Simulink	dSPACE DS1103 control board based on TM320F240 DSP (40MIPS, 16-bit fixed point arithmetic)	Low	High
Ming Yang (2010)	Adaptive Weighted PSO	MATLAB/Simulink	-	-	High
Liu Mingji et al., (2004)	Model reference adaptive control	-	Industry computer (without specifications)	Medium	Medium
Zhang Yaou et al., (2010)	Model Reference Dynamic Inversion	MATLAB/Simulink	-	-	High
Mohamed Kadjoudj et al., (2007)	Model reference fuzzy-logic adaptive control with fuzzy logic controller	MATLAB/Simulink	-	-	High
Ying-Shieh Kung and Pin-Ging Huang (2004)	Model reference adaptive control with fuzzy logic controller	-	TMS320F2812 DSP (150MIPS, 32-bit CPU, fixed point arithmetic, motor control peripherals, 12-bit ADC @ 12.5 MSPS)	High	High
Mahmoud M. Saafan et al., (2012)	Artificial Neural Network Control	MATLAB/Simulink	-	-	Medium
Fadil Hicham et al., (2015)	Sliding-Mode Speed Control with Fuzzy-Logic Chattering Minimization	MATLAB/Simulink	eZdsp F28335 Board based on TMS320F28335 DSC (150MIPS, 32-bit CPU, IEEE-754 single-precision floating point unit, 12-bit ADC @ 12.5 MSPS)	Low	Medium
Fouad Giri (2013)	High order terminal sliding mode control with mechanical resonance suppressing	MATLAB/Simulink	-	-	Medium
Fayez F.M. El-Sousy (2014)	Model reference adaptive hybrid control (fuzzy-neural-network controller, robust controller, auxiliary sliding mode controller)	MATLAB/Simulink	dSPACE DS1102 control board based on TMS320C31 (floating point arithmetic, 40MIPS, 32-bit CPU) and TMS320P14 (fixed point arithmetic, 8.77MIPS, 32-bit ALU, 16x16 hardware multiplier) DSP chips	High	Very High

Table 2.1 Summary of the reviewed non-linear control methods for PMSM drives

3. CONTROL SYSTEM DESIGN FOR PMSM DRIVES

The control architecture of a high-performance electric drive system, designed to track a position reference, is composed for at least two control loops disposed in a cascade fashion. The current/torque controller will be in the inner-most loop, which is required to add robustness against stator resistance sensitivity. In addition, the torque loop will facilitate velocity control due to the intrinsic relationship between torque and acceleration. The intermediate control loop can be a speed controller, which helps to minimize the effects due to temperature sensitivity of the permanent magnets. The final control objective is accomplished by a position controller, which conforms the outermost loop of the overall control system. If only two-loops are considered, the explicit intermediate speed-loop is replaced by a more complex position controller.

This chapter includes the mathematical model of the PMSM machine, and presents the design of all the controllers required in a position tracking drive system. Conventional PID controllers, as well as fuzzy-logic based controllers are designed.

3.1. Mathematical model of Permanent Magnet Synchronous Machines

Obtaining a suitable dynamic model is the starting point to design and analyze any control system. The PMSM dynamic model is obtained considering the fundamental relationship between stator voltages and currents, expressed in the space phasor form. The procedure followed to obtain the PMSM mathematical model is based on reference [30]. Considering a three-phase machine with balanced three-phase currents given by

$$i_a(t) = I_s \cos(\omega t + \phi)$$

$$i_b(t) = I_s \cos\left(\omega t + \phi - \frac{2\pi}{3}\right)$$

$$i_c(t) = I_s \cos\left(\omega t + \phi - \frac{4\pi}{3}\right)$$

Where ω is the phase current frequency, ϕ is the initial angle, and I_s is the amplitude. The space vector representation of the three-phase stator current can be written as

$$\overrightarrow{i_s} = \frac{2}{3}\left[i_a(t) + i_b(t)e^{j\frac{2\pi}{3}} + i_c(t)e^{j\frac{4\pi}{3}}\right]$$

$$\overrightarrow{i_s} = I_s e^{j(\omega t+\phi)}$$

And the space vector representation of the three-phase stator voltage

$$\overrightarrow{v_s} = \frac{2}{3}\left[v_a(t) + v_b(t)e^{j\frac{2\pi}{3}} + v_c(t)e^{j\frac{4\pi}{3}}\right]$$

Assuming that $\overrightarrow{\varphi_s}$ is the space vector representation of the stator flux linkage, the stator voltage equation of the machine is

$$\overrightarrow{v_s} = R_s\overrightarrow{\imath_s} + \frac{d\overrightarrow{\varphi_s}}{dt} \tag{3.1}$$

Where $R_s\overrightarrow{\imath_s}$ is the voltage drop across the equivalent stator resistance, and $\frac{d\overrightarrow{\varphi_s}}{dt}$ is the induced voltage due to magnetic flux variations.

3.1.1. Representation in Stationary Reference Frame $(\boldsymbol{\alpha - \beta})$

Projecting the three phase space vectors of the voltage and currents onto the real (α) and imaginary (β) axes, these vectors can be represented by complex notations as follows

$$\overrightarrow{v_s} = v_\alpha + jv_\beta$$
$$\overrightarrow{\imath_s} = i_\alpha + ji_\beta$$

The relationship between the three-phase variables and the $\alpha - \beta$ variables is given by the Clarke transformation as follows

$$\begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix}$$

The coefficient $\frac{2}{3}$ is used to guarantee the energy conservation. The x_0 term represents the zero-sequence component of the three-phase system. For a balanced three-phase system, the x_0 term is zero.

The inverse Clarke transformation is defined as

$$\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix}\begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix}$$

The voltage and current variables in the α-β reference frame are still sinusoidal because of the direct relationship established by the Clarke transformation.

3.1.2. Representation in Rotating Reference Frame $(\boldsymbol{d - q})$

Rotating the space vector in $\alpha - \beta$ reference frame clockwise by θ_e, the $d - q$ reference frame is obtained. In this reference frame, the direct axis d is always aligned with the rotating flux produced by the permanent magnets of the rotor, and the q axis is in

quadrature. Because the rotor runs at the same speed as the supplying frequency at steady-state, this reference frame is also called the synchronous reference frame.

Mathematically, the rotation of the space vectors is translated into multiplication by the factor $e^{-j\theta_e}$, which leads to a set of new space vectors $\overrightarrow{v_s}', \overrightarrow{\imath_s}'$ denoting the space vectors referred to synchronous $d-q$ reference frame. Projecting the transformed space vectors into the real and imaginary axes, the current and voltage variables in the $d-q$ reference frame are

$$\overrightarrow{v_s}' = \overrightarrow{v_s} e^{-j\theta_e} = v_d + jv_q \tag{3.2}$$

$$\overrightarrow{\imath_s}' = \overrightarrow{\imath_s} e^{-j\theta_e} = i_d + ji_q \tag{3.3}$$

Similar, the stator flux can also be represented in the $d-q$ frame by rotating the flux vector clockwise by θ_e, leading to

$$\overrightarrow{\varphi_s}' = \overrightarrow{\varphi_s} e^{-j\theta_e} = \varphi_d + j\varphi_q \tag{3.4}$$

The real and imaginary parts of the flux vector in the $d-q$ frame are

$$\varphi_d = L_d i_d + \phi_{mg} \tag{3.5}$$

$$\varphi_q = L_q i_q \tag{3.6}$$

Where ϕ_{mg} is the amplitude of the flux introduced by the permanent magnets, and is assumed to be constant.

Multiplying the original voltage equation (3.1) by $e^{-j\theta}$ gives

$$\overrightarrow{v_s} e^{-j\theta_e} = R_s \overrightarrow{\imath_s} e^{-j\theta_e} + \frac{d\overrightarrow{\varphi_s}}{dt} e^{-j\theta_e} \tag{3.7}$$

Now, taking derivative on both sides of equation (3.4)

$$\overrightarrow{\varphi_s}' = \overrightarrow{\varphi_s} e^{-j\theta_e}$$

$$\frac{d\overrightarrow{\varphi_s}'}{dt} = \frac{d\overrightarrow{\varphi_s}}{dt} e^{-j\theta_e} - j\omega_e e^{-j\theta_e} \overrightarrow{\varphi_s}$$

$$\frac{d\overrightarrow{\varphi_s}'}{dt} = \frac{d\overrightarrow{\varphi_s}}{dt} e^{-j\theta_e} - j\omega_e \overrightarrow{\varphi_s}'$$

The following expression is obtained

$$\frac{d\overrightarrow{\varphi_s}}{dt} e^{-j\theta_e} = \frac{d\overrightarrow{\varphi_s}'}{dt} + j\omega_e \overrightarrow{\varphi_s}' \tag{3.8}$$

Substituting (3.2), (3.3), and (3.8) into (3.7), the voltage equation in terms of the space vectors $\overrightarrow{v_s}', \overrightarrow{\imath_s}'$ has the following form

$$\overrightarrow{v_s}' = R_s \overrightarrow{\imath_s}' + \frac{d\overrightarrow{\varphi_s}'}{dt} + j\omega_e \overrightarrow{\varphi_s}' \tag{3.9}$$

This equation governs the relationship between the voltage and current variables in space vector form that leads to the dynamic model in the $d-q$ reference frame.
Rewriting the equation (3.9) in its complex form

$$v_d + jv_q = R_s i_d + jR_s i_q + \frac{d\varphi_d}{dt} + j\frac{d\varphi_q}{dt} + j\omega_e\varphi_d - \omega_e\varphi_q$$

The real and imaginary components of the left-hand side are equal to the corresponding components of the right-and side, therefore

$$v_d = R_s i_d + \frac{d\varphi_d}{dt} - \omega_e\varphi_q$$

$$v_q = R_s i_q + \frac{d\varphi_q}{dt} + \omega_e\varphi_d$$

Finally, substituting (3.5) and (3.6) in the above equations, the $d-q$ model equations of the PMSM are

$$v_d = R_s i_d + L_d\frac{di_d}{dt} - \omega_e L_q i_q$$

$$v_q = R_s i_q + L_q\frac{di_q}{dt} + \omega_e L_d i_d + \omega_e\phi_{mg}$$

The relationship between the variables in the $\alpha-\beta$ and $d-q$ reference frame is given by the Park's transformation

$$\begin{bmatrix} x_d \\ x_q \end{bmatrix} = \begin{bmatrix} \cos(\theta_e) & \sin(\theta_e) \\ -\sin(\theta_e) & \cos(\theta_e) \end{bmatrix}\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix}$$

Conversely, the inverse Park's transformation is defined as

$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \begin{bmatrix} \cos(\theta_e) & -\sin(\theta_e) \\ \sin(\theta_e) & \cos(\theta_e) \end{bmatrix}\begin{bmatrix} x_d \\ x_q \end{bmatrix}$$

3.1.3. Electromagnetic Torque

The electromagnetic torque is computed as the cross product of the space vector of the stator flux with the stator current. In the $d-q$ reference frame the electromagnetic torque is given by

$$T_e = \frac{3}{2}Z_p\overrightarrow{\varphi_s}' \otimes \overrightarrow{\imath_s}'$$

$$T_e = \frac{3}{2}Z_p\left(\varphi_d i_q - \varphi_q i_d\right)$$

Replacing the equations (3.5) and (3.6) in the above equation leads to

$$T_e = \frac{3}{2}Z_p\left[\phi_{mg} i_q + \left(L_d - L_q\right)i_d i_q\right]$$

Where Z_p is the number of pole pairs.

3.1.4. Complete Model of PMSM in $(d-q)$ reference frame

For a PMSM with multiple pair of poles, the electrical speed relates to the mechanical speed by

$$\omega_e = Z_p \omega_m$$

The dynamic equation that describes the rotation of the motor is given by

$$J_m \frac{d\omega_m}{dt} = T_e - B_v \omega_m - T_L$$

Where J_m is the total inertia, B_v is the viscous friction coefficient and T_L is the load torque. Replacing the mechanical speed with the electrical speed gives

$$\frac{d\omega_e}{dt} = \frac{Z_p}{J_m}\left(T_e - \frac{B_v}{Z_p}\omega_e - T_L\right)$$

Now, considering a control with $i_d = 0$, the electromagnetic torque equation is

$$T_e = \frac{3}{2} Z_p \phi_{mg} i_q$$

With this torque equation, the differential equation for the electrical speed becomes

$$\frac{d\omega_e}{dt} = \frac{Z_p}{J_m}\left(\frac{3}{2} Z_p \phi_{mg} i_q - \frac{B_v}{Z_p}\omega_e - T_L\right)$$

Using the above results, the complete dynamic model of a PMSM in the $d-q$ rotating reference frame is governed by the following differential equations

$$\frac{di_d}{dt} = \frac{1}{L_d}\left(v_d - R_s i_d + \omega_e L_q i_q\right) \quad (3.10)$$

$$\frac{di_q}{dt} = \frac{1}{L_q}\left(v_q - R_s i_q - \omega_e L_d i_d - \omega_e \phi_{mg}\right) \quad (3.11)$$

$$\frac{d\omega_e}{dt} = \frac{Z_p}{J_m}\left(\frac{3}{2} Z_p \phi_{mg} i_q - \frac{B_v}{Z_p}\omega_e - T_L\right) \quad (3.12)$$

3.2. Standard PID Control System Design

A position control system with permanent magnet synchronous drives based on PID controllers has to contain at least two cascade control loops, one for current/torque regulation and other for position control. An intermediate speed control loop can be inserted between the current/torque loop and the position loop, leading to a three-loop position control system. The intermediate speed control loop adds robustness against parameter variations due to temperature sensitivity of the magnets [7].

The inner-most control loop is the current/torque loop. PI controllers are used to regulate the d-axis ($i_d = 0$) and the q-axis currents of this loop. The outer-most and primary control objective is in the position control loop. A cascade control system structure is

applied to control the position of the permanent magnet synchronous drive. Two approaches are considered in terms of the control loops applied. The first approach is to use a PID controller for the position control loop which directly feeds the current reference signal to the current/torque control loop. The second approach is to use an intermediate speed controller which receives the speed command signal from the position controller and feeds the current reference signal to the current/torque controller.

Each control loop has different bandwidths. The innermost current control loop will have the biggest bandwidth and the outermost position control loop will have the smaller bandwidth of the overall system.

The pole-placement design technique is applied for tuning the PID controllers of the PMSM. The main idea behind the pole-placement approach is to select the appropriate closed loop performance based on the desired damping ratio ξ and the desired undamped natural frequency ω_n. The denominator of the closed loop transfer function is made equal to a desired closed loop polynomial. To apply the pole-placement technique, a first-order or a second-order model of the plant is required [30].

3.2.1. PI Controller Design

Assuming a plant represented by a first order model with the following transfer function

$$G(s) = \frac{b}{s+a}$$

and a PI controller whose transfer function is

$$C(s) = K_c\left(1 + \frac{1}{\tau_I s}\right)$$

Where K_c is the proportional gain and τ_I is the integral time constant. Figure 3.1 shows the block diagram of the PI control system

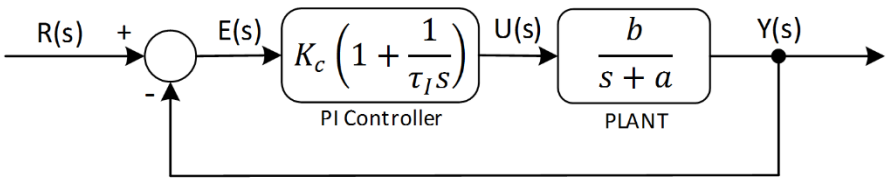

Figure 3.1 Block diagram of PI control system

Rewriting the PI controller transfer function as

$$C(s) = \frac{c_1 s + c_0}{s}$$

where $K_c = c_1$

$$\tau_I = \frac{c_1}{c_0}$$

The closed loop transfer function from the reference signal to the output signal is expressed as

$$\frac{Y(s)}{R(s)} = \frac{G(s)C(s)}{1 + G(s)C(s)}$$

$$\frac{Y(s)}{R(s)} = \frac{\frac{b}{s+a}\frac{c_1 s + c_0}{s}}{1 + \frac{b}{s+a}\frac{c_1 s + c_0}{s}}$$

$$\frac{Y(s)}{R(s)} = \frac{b(c_1 s + c_0)}{s(s+a) + b(c_1 s + c_0)}$$

The closed-loop poles can be found solving

$$s(s+a) + b(c_1 s + c_0) = 0$$

The locations of the closed-loop poles determine the closed-loop stability, speed response and disturbance rejection of the system.

Using the pole-placement technique and selecting the damping coefficient and the natural frequency of a second order polynomial as the design parameters, the following polynomial equation is set

$$s(s+a) + b(c_1 s + c_0) = s^2 + 2\xi\omega_n s + \omega_n^2$$

Where ξ is the damping coefficient and ω_n is the natural frequency or bandwidth of the closed-loop system.

Rearranging the elements in the left-hand side

$$s^2 + (a + bc_1)s + bc_0 = s^2 + 2\xi\omega_n s + \omega_n^2$$

Equating the elements in the left-hand side to the elements in the right-hand side

$$a + bc_1 = 2\xi\omega_n$$

$$bc_0 = \omega_n$$

Solving for c_0 and c_1

$$c_0 = \frac{\omega_n}{b}$$

$$c_1 = \frac{2\xi\omega_n - a}{b}$$

Finally, the proportional gain and the integral time constant of the PI controller are found as

$$K_c = c_1 = \frac{2\xi\omega_n - a}{b}$$

$$\tau_I = \frac{c_1}{c_0} = \frac{2\xi\omega_n - a}{\omega_n^2}$$

The selection of ξ and ω_n is made according to the desired closed-loop performance of the system.

3.2.2. PID Controller Design

A position control system without an explicit speed control loop will require a PID controller because the transfer function from the reference angular position to the reference current/torque will be of second order [30]. The second order transfer function of the plant will have the following form

$$\frac{Y(s)}{U(s)} = \frac{b}{s(s+a)}$$

Considering an ideal PID controller with the transfer function

$$C(s) = K_c\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)$$

where K_c is the proportional gain, τ_I is the integral time constant and τ_D is the derivative gain. Rewriting the PID controller as

$$C(s) = \frac{c_2 s^2 + c_1 s + c_0}{s}$$

where

$$K_c = c_1$$

$$\tau_I = \frac{c_1}{c_0}$$

$$\tau_D = \frac{c_2}{c_1}$$

The closed loop transfer function with the PID controller, from the reference signal to the output signal is expressed as

$$\frac{Y(s)}{R(s)} = \frac{G(s)C(s)}{1 + G(s)C(s)}$$

$$\frac{Y(s)}{R(s)} = \frac{\frac{b(c_2 s^2 + c_1 s + c_0)}{s^2(s+a)}}{1 + \frac{b(c_2 s^2 + c_1 s + c_0)}{s^2(s+a)}}$$

$$= \frac{b(c_2 s^2 + c_1 s + c_0)}{s^2(s+a) + b(c_2 s^2 + c_1 s + c_0)}$$

As can be seen, the closed loop polynomial is of third order, thus, it is required to select three desired closed-loop poles for the closed-loop performance specification. The pair of dominant poles are selected as

$$s_{1,2} = -\xi\omega_n \pm j\omega_n\sqrt{1-\xi^2}$$

The third pole is chosen to be

$$s_3 = -n\omega_n$$

with $n \gg 1$ so that ω_n can be considered the bandwidth of the desired closed-loop system. With these specifications, the closed-loop polynomial is

$$(s^2 + 2\xi\omega_n s + \omega_n^2)(s + n\omega_n) = s^3 + t_2 s^2 + t_1 s + t_0$$

where

$$t_2 = (2\xi + n)\omega_n$$

$$t_1 = (2\xi n + 1)\omega_n^2$$

$$t_0 = n\omega_n^3$$

Now, the desired closed-loop polynomial is equated with the actual closed-loop polynomial

$$s^2(s+a) + b(c_2 s^2 + c_1 s + c_0) = s^3 + t_2 s^2 + t_1 s + t_0$$

Comparing the coefficients form both sides, the controller parameters are found as

$$c_2 = \frac{t_2 - a}{b} = \frac{(2\xi + n)\omega_n - a}{b}$$

$$c_1 = \frac{t_1}{b} = \frac{(2\xi n + 1)\omega_n^2}{b}$$

$$c_0 = \frac{t_0}{b} = \frac{n\omega_n^3}{b}$$

Finally, the PID controller parameters are found as

$$K_c = c_1 = \frac{(2\xi n + 1)\omega_n^2}{b}$$

$$\tau_I = \frac{c_1}{c_0} = \frac{(2\xi n + 1)\omega_n^2}{n\omega_n^3} = \frac{(2\xi n + 1)}{n\omega_n}$$

$$\tau_D = \frac{c_2}{c_1} = \frac{(2\xi + n)\omega_n - a}{(2\xi n + 1)\omega_n^2}$$

The derivative term should be implemented directly on the output signal to avoid a derivative “kick” due to a step reference signal change.

3.2.3. Current Controller

The first control loop required for any high-performance drive control system is the current/torque loop. In this loop, the d-axis and the q-axis currents of the PMSM are regulated using PI controllers. A schematic diagram for the current control of a PMSM drive is presented in the figure 3.2

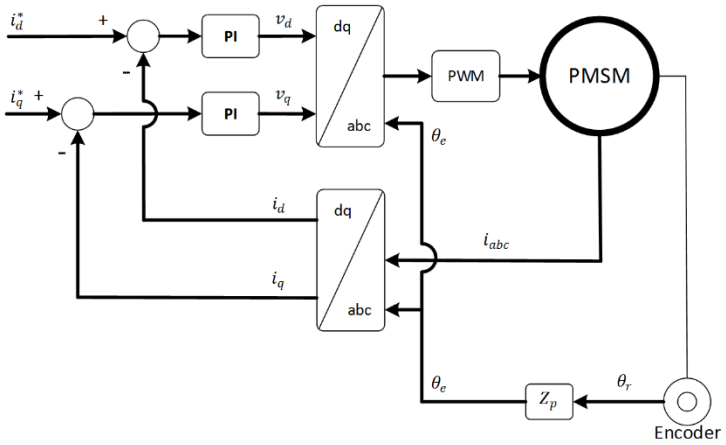

Figure 3.2 Schematic diagram for current control of PMSM drives

The feedback signals of the controllers are the d-axis current i_d and the q-axis current i_q. These feedback signals are obtained measuring the three-phase currents and applying the Clark’s and Park’s transformations as follows

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}$$

The electrical angular position of the rotor θ_e is required to apply the above equations, and is obtained through a position sensor such as an encoder or a resolver.

As can be seen in the PMSM mathematical model presented in 3.1.4, there are nonlinear cross-coupling terms in the differential equations for the d-q currents. These cross-coupling terms can be eliminated with an input-and-output linearization and feedforward manipulation, as outlined in [30].

Using the auxiliary variables $\widehat{v_d}$, $\widehat{v_q}$ defined such that

$$\frac{1}{L_d}\widehat{v_d} = \frac{1}{L_d}\left(v_d + \omega_e L_q i_q\right)$$

$$\frac{1}{L_q}\widehat{v_q} = \frac{1}{L_q}\left(v_q - \omega_e L_d i_d - \omega_e \phi_{mg}\right)$$

By replacing the above equations into the PMSM model equations (3.10) and (3.11), the following first order differential equations are obtained

$$\frac{di_d}{dt} = -\frac{R_s}{L_d} i_d + \frac{1}{L_d}\widehat{v_d}$$

$$\frac{di_q}{dt} = -\frac{R_s}{L_q} i_q + \frac{1}{L_d}\widehat{v_q}$$

The Laplace transfer functions of the above equations are

$$\frac{I_d(s)}{\widehat{V_d}(s)} = \frac{\frac{1}{L_d}}{s + \frac{R_s}{L_d}}$$

$$\frac{I_q(s)}{\widehat{V_q}(s)} = \frac{\frac{1}{L_q}}{s + \frac{R_s}{L_q}}$$

With these first-order plant models, the PI current controllers are parametrized using the pole-placement technique explained in 3.2.1. The PI controller parameters for the d-axis current are

$$K_c^d = \frac{2\xi\omega_n - \frac{R_s}{L_d}}{\frac{1}{L_d}} \tag{3.13}$$

$$\tau_I^d = \frac{2\xi\omega_n - \frac{R_s}{L_d}}{\omega_n^2} \tag{3.14}$$

and for the q-axis current

$$K_c^q = \frac{2\xi\omega_n - \frac{R_s}{L_q}}{\frac{1}{L_q}} \tag{3.15}$$

$$\tau_I^q = \frac{2\xi\omega_n - \frac{R_s}{L_q}}{\omega_n^2} \tag{3.16}$$

The damping coefficient ξ is selected to be 0.707 or 1. The natural frequency ω_n determine the desired closed-loop settling time, which also correspond to the desired bandwidth of the closed-loop system. Therefore, the larger ω_n is, the shorter the desired closed-loop settling time is.

Selecting ω_n relative to the bandwidth of the open-loop system $\left(\frac{R_s}{L_d}\ or \frac{R_s}{L_q}\right)$ and using a normalized parameter $0 < \gamma < 1$, the parameter ω_n is calculated as

$$\omega_n = \frac{1}{1-\gamma}\frac{R_s}{L_d}$$

for the d-axis current control, and for the q-axis current control

$$\omega_n = \frac{1}{1-\gamma}\frac{R_s}{L_q}$$

As the normalized parameter γ gets closer to 1, ω_n tends to ∞. The parameter γ is selected around 0.8 or 0.9 in order to obtain a fast response.

With the controller parameters calculated, the voltage control signals will be

$$v_d = K_c^d\ e_d + \frac{K_c^d}{\tau_I^d}\int_0^t e_d(\tau)d\tau + f_d \quad (3.17)$$

$$v_q = K_c^q\ e_q + \frac{K_c^q}{\tau_I^q}\int_0^t e_q(\tau)d\tau + f_q \quad (3.18)$$

Where

$$e_d = i_d^* - i_d$$

$$e_q = i_q^* - i_q$$

$$f_d = -\omega_e L_q i_q$$

$$f_q = \omega_e L_d i_d + \omega_e \phi_{mg}$$

3.2.4. Position Controller without Explicit Speed Control Loop

In this approach, there are only two control loops in the overall control system, namely, the inner current control loop and the outer position control loop. A PID controller is applied in the position loop. The derivative action in the position controller works as an equivalent proportional gain of a speed controller. The design starts with the relationship between angular speed and angular position

$$\theta_e(t) = \int_0^t \omega_e(\tau)d\tau$$

The Laplace transfer function between the velocity $\Omega_e(s)$ and the angular position $\Theta_e(s)$, is given by

$$\frac{\Theta_e(s)}{\Omega_e(s)} = \frac{1}{s}$$

The relationship between the q-axis current and the angular velocity is obtained from equation (3.12) and is given by

$$\left(s + \frac{B_v}{J_m}\right)\Omega_e(s) = \frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m} I_q(s)$$

$$\frac{\Omega_e(s)}{I_q(s)} = \frac{\frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}}{s + \frac{B_v}{J_m}}$$

Therefore, the relationship between the angular position and the q-axis current will be

$$\frac{\Theta_e(s)}{\Omega_e(s)}\frac{\Omega_e(s)}{I_q(s)} = \frac{\Theta_e(s)}{I_q(s)} = \frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}\frac{1}{s\left(s + \frac{B_v}{J_m}\right)}$$

Setting the bandwidth of the current loop much bigger than the bandwidth of the position loop, the inner-loop dynamics of the current regulator can be neglected, thus, the approximation $I_q(s) = I_q^*(s)$ is taken. As a result, a second order model is obtained as follows

$$\frac{\Theta_e(s)}{I_q^*(s)} = \frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}\frac{1}{s\left(s + \frac{B_v}{J_m}\right)} = \frac{b}{s(s+a)}$$

With this model, a PID position controller can be designed using the pole-placement approach described in 3.2.2. The controller parameters are calculated according to the following equations

$$K_c = \frac{(2\xi n + 1)\omega_n^2}{\frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}} \tag{3.19}$$

$$\tau_I = \frac{(2\xi n + 1)}{n\omega_n} \tag{3.20}$$

$$\tau_D = \frac{(2\xi + n)\omega_n - \frac{B_v}{J_m}}{(2\xi n + 1)\omega_n^2} \tag{3.21}$$

The damping coefficient ξ is selected to be 0.707 or 1, and the natural frequency ω_n is forced to be at least 1/10 of the natural frequency of the current loop.

The control signal is calculated using a combination of the proportional, integral and derivative terms. To avoid overshoots, the proportional and derivative actions are implemented on the output only. The reference q-axis current is calculated as follows

$$i_q^* = -K_c\theta_e + \frac{K_c}{\tau_I}\int_0^t (\theta_e^*(\tau) - \theta_e(\tau)d\tau \; - K_c\tau_D\omega_e \tag{3.22}$$

The block diagram for the described angular position control is illustrated in figure 3.3

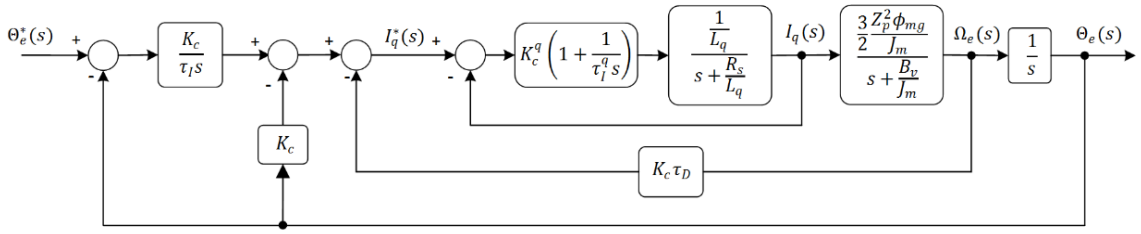

Figure 3.3 Block diagram for angular position control without explicit speed control loop

3.2.5. Position Controller with Intermediate Speed Control Loop

This approach implements an intermediate speed control loop with the reference signal supplied by the position controller. Therefore, the position controller design is simplified and can be implemented as a simple proportional controller plus a feedforward speed signal calculated as the derivative of the reference angular position. Nevertheless, an additional PI controller is required to regulate the angular speed.

3.2.5.1. Speed Controller

Rewriting the speed differential equation (3.12) as

$$\frac{d\omega_e}{dt} = -\frac{B}{J_m}\omega_e + \frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}i_q - \frac{Z_p}{J_m}T_L$$

and applying the Laplace transformation to get the relationship between the angular velocity and the q-axis current

$$\left(s+\frac{B_v}{J_m}\right)\Omega_e(s) = \frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}I_q(s)$$

$$\frac{\Omega_e(s)}{I_q(s)} = \frac{\frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}}{s+\frac{B_v}{J_m}}$$

Now, replacing v_q in the q-axis current differential equation (3.11) with the value given by the PI current controller, and assuming cancelation of the nonlinear terms, the following differential equation is obtained

$$\frac{di_q}{dt} = -\frac{R_s}{L_q} i_q + \frac{1}{L_q} K_c^q \left(i_q^* - i_q\right) + \frac{K_c^q}{\tau_I^q L_q} \int_0^t \left(i_q^*(\tau) - i_q(\tau)\right) d\tau$$

Taking the Laplace transformation of the above equation leads to

$$sI_q(s) = -\frac{R_s}{L_q} I_q(s) + \frac{K_c^q}{L_q}\left(I_q^*(s) - I_q(s)\right) + \frac{K_c^q}{\tau_I^q L_q s}\left(I_q^*(s) - I_q(s)\right)$$

$$\frac{I_q(s)}{I_q^*(s)} = \frac{\frac{K_c^q}{L_q} + \frac{K_c^q}{\tau_I^q L_q s}}{s + \frac{R_s}{L_q} + \frac{K_c^q}{L_q} + \frac{K_c^q}{\tau_I^q L_q s}}$$

The following identities are obtained from equations (3.15) and (3.16) of the current controller design

$$\frac{K_c^q}{\tau_I^q} = L_q \omega_n^2$$

$$\frac{K_c^q}{L_q} = 2\xi\omega_n - \frac{R_s}{L_q}$$

Applying the above identities, the transfer function from the q-axis reference current to the actual q-axis current is given by

$$\frac{I_q(s)}{I_q^*(s)} = \frac{\left(2\xi\omega_n - \frac{R_s}{L_q}\right)s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

Using the above transfer function together with equation (3.12), the relationship between the reference q-axis current $I_q^*(s)$, and the electrical speed $\Omega_e(s)$ is given by

$$\frac{\Omega_e(s)}{I_q^*(s)} = \left(\frac{\frac{3}{2}\frac{Z_p^2 \phi_{mg}}{J_m}}{s + \frac{B_v}{J_m}}\right)\left(\frac{\left(2\xi\omega_n - \frac{R_s}{L_q}\right)s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}\right)$$

In order to design a PI controller using the pole-placement approach, a first-order plant model is required. Therefore, the above transfer function needs to be approximated by a first order model.

If the natural frequency ω_n is chosen to be much greater than the mechanical relationship $\frac{B_v}{J_m}$, the inner current-loop dynamics can be neglected, and the following first-order model approximation can be taken

$$\frac{\Omega_e(s)}{I_q^*(s)} \approx \frac{\frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}}{s+\frac{B_v}{J_m}}$$

Applying the pole-placement design technique explained in 3.2.1, the PI controller parameters for the speed loop are calculated as follows

$$K_c = \frac{2\xi\omega_n - \frac{B_v}{J_m}}{\frac{3}{2}\frac{Z_p^2\phi_{mg}}{J_m}} \tag{3.23}$$

$$\tau_I = \frac{2\xi\omega_n - \frac{B_v}{J_m}}{\omega_n^2} \tag{3.24}$$

3.2.5.2. Position Controller

The position controller consists of a simple proportional controller plus a feedforward speed signal calculated as the derivative of the reference angular position. The control action of the proportional controller, which is the speed reference signal, is calculated with equation (3.25)

$$\omega_e^* = K_p(\theta_e^* - \theta_e) + \hat{\omega}_e^* \tag{3.25}$$

where $\hat{\omega}_e^* = \frac{d\theta_e^*}{dt}$

The block diagram for angular position control with the inner speed control loop is presented in figure 3.4

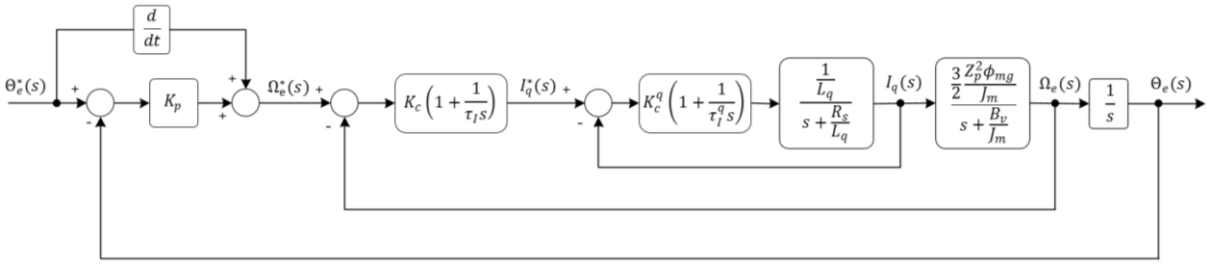

Figure 3.4 Block diagram for angular position control with intermediate speed control loop

3.3. Hybrid Control System Design based on Fuzzy-Logic

Nonlinearities, unmodeled dynamics, and parameter variations can affect the control performance of a PMSM drive system. A hybrid control system based on fuzzy-logic is proposed as a way to cope with these drawbacks.

While conventional control systems are based on the mathematical model of the plant, fuzzy control is based on the intuition and experience of the human operator. And thus, for plants with vaguely known models, fuzzy control is clearly opportune and adequate. In essence, implicitly, fuzzy motion control is self-adaptive and thus its robustness becomes apparent [6].

3.3.1. Direct Fuzzy-Logic Position Controller

The first step in designing fuzzy-logic controllers is to define inputs, outputs, and its corresponding membership functions. Considering a PMSM position controller with intermediate speed controller, the inputs are selected to be the error and the variation of the error, and the output will be the reference angular speed. A scaling factor is applied for each signal. Figure 3.5 presents a schematic diagram of the proposed direct fuzzy-logic controller.

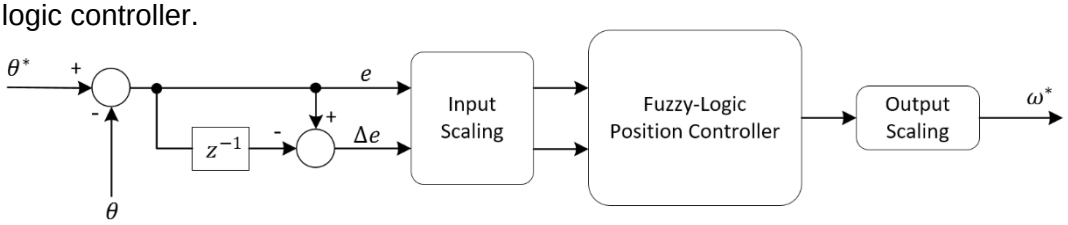

Figure 3.5 Direct fuzzy-logic position controller

Seven triangular-shaped membership functions with 50% overlap, are applied for each input and output of the fuzzy controller. The membership functions are symmetrically distributed along the corresponding universe of discourse, which is stablished according to the process operating ranges. The same universe of discourse is applied for the error and the variation of the error, and a normalized universe of discourse is taken for the output. The names for the membership functions are defined as follows

NB = negative big
NM = negative medium
NS = negative small
Z = zero
PS = positive small
PM = positive medium
PB = positive big

The membership functions for the inputs are shown in figure 3.6

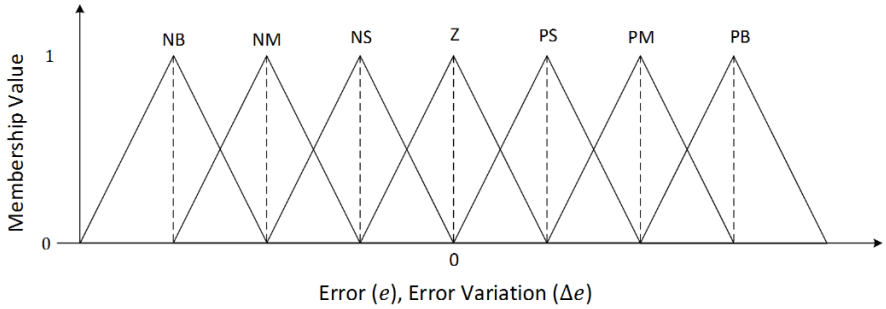

Figure 3.6 Input membership functions of the fuzzy-logic position controller

The membership functions for the output, with a normalized universe of discourse, are presented in figure 3.7

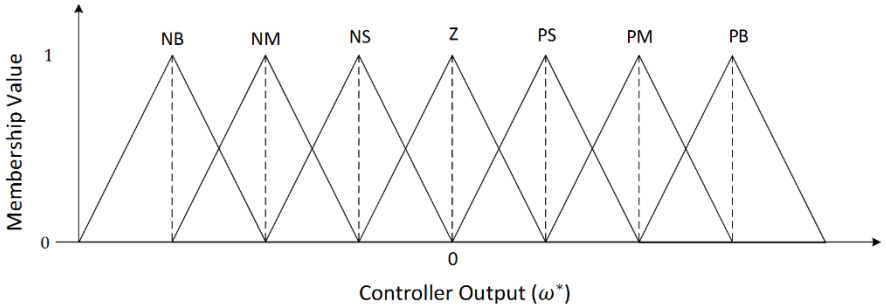

Figure 3.7 Output membership functions of the fuzzy-logic position controller

A Mamdani-type fuzzy inference system is applied. The minimum operation is used as the 'AND' method for fuzzy implication, and the maximum operation is applied for the union of all outputs.

The rule-base of the fuzzy controller relates the error and the error variation to obtain a consequent output. The linguistic fuzzy rules are based on the Macvicar-Whelan matrix described in Table 3.1

	$\Delta\ ERROR$							
$ERROR$		NB	NM	NS	Z	PS	PM	PB
	NB	NB	NB	NB	NB	NM	NS	Z
	NM	NB	NB	NB	NM	NS	Z	PS
	NS	NB	NB	NM	NS	Z	PS	PM
	Z	NB	NM	NS	Z	PS	PM	PB
	PS	NM	NS	Z	PS	PM	PB	PB
	PM	NS	Z	PS	PM	PB	PB	PB
	PB	Z	PS	PM	PB	PB	PB	PB

Table 3.1 Rule-base for the fuzzy-logic position controller

The weighted average defuzzification method is applied to find the crisp output value. Mathematically, this method is defined as

$$u^{crisp} = \frac{\sum_{k=1}^{m} c[k] f[k]}{\sum_{k=1}^{m} f[k]}$$

where $c[k]$ is the center value of the individual k-output membership function, and $f[k]$ is the corresponding membership value.

3.3.2. Fuzzy-Logic Position Controller with Proportional Action

In order to improve the position controller response at steady-state, an error proportional factor is applied to the output of the controller. With this scheme, the control action strength is reduced as the position gets closer to its reference value and thus, the oscillations at steady-state are reduced. The block diagram of the proposed controller is presented in figure 3.6

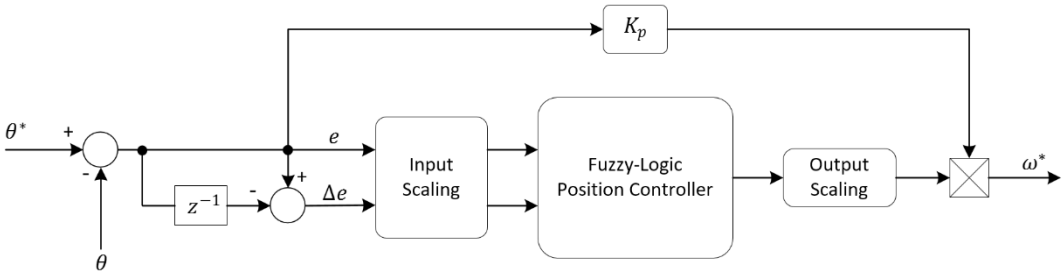

Figure 3.8 Fuzzy-logic position controller with proportional action

The output of this controller will be

$$\omega^* = u^{crisp} K_p e$$

The output scaling factor and the proportional constant have to be properly parametrized in order to maintain the effect of the fuzzy-logic controller output in the final control action. In general terms, the output scaling factor of the fuzzy controller has to be selected much bigger that the proportional constant.

3.3.3. Fuzzy Tuned PI Speed Controller

The use of a fuzzy inference system can be adopted to determine the values of the PI speed controller parameters during the transient response in order to decrease the rise time and reduce the overshoot. This approach is based on the fuzzy set-point weighting methodology presented in [31].

A schematic diagram of the proposed fuzzy-tuned PI speed controller is presented in figure 3.9

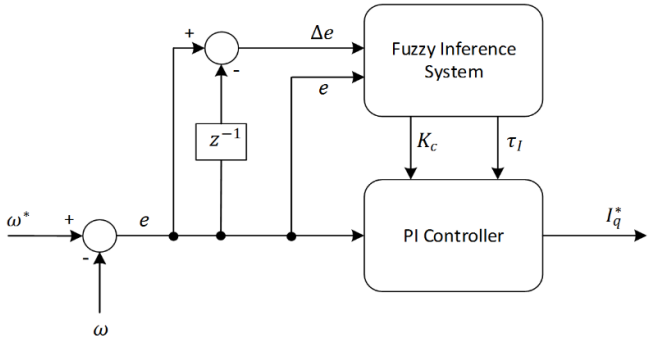

Figure 3.9 Block diagram for the fuzzy-tuned PI speed controller

As can be seen in the block diagram, the fuzzy inference system has two inputs which are the error and the error variation, and has two outputs corresponding to the proportional and integral parameters of the PI speed controller.

Five triangular-shaped membership functions with 50% overlap, are applied for each input and output of the fuzzy inference system. The membership functions are symmetrically distributed along the corresponding universe of discourse, which is stablished according to the process operating ranges. The same universe of discourse is applied for the error and the error variation, and a normalized universe of discourse is taken for the outputs. The names for the input membership functions are defined as follows

NB = Negative Big

N = Negative

Z = Zero

P = Positive

PB = Positive Big

The membership functions for the inputs are shown in figure 3.10

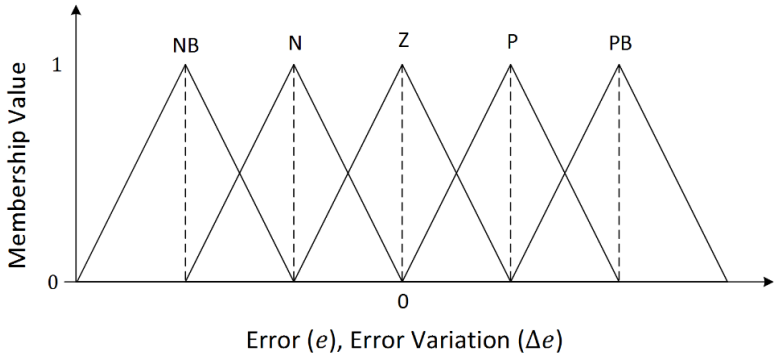

Figure 3.10 Input membership functions for the fuzzy-tuned PI speed controller

The names for the output membership functions are defined as follows

VS = Very Small

S = Small

M = Medium

L = Large

VL = Very Large

The output values will be the parameters of the PI speed controller, designated as K_p and $K_i = \frac{K_p}{\tau_I}$. Figure 3.11 shows the output membership functions

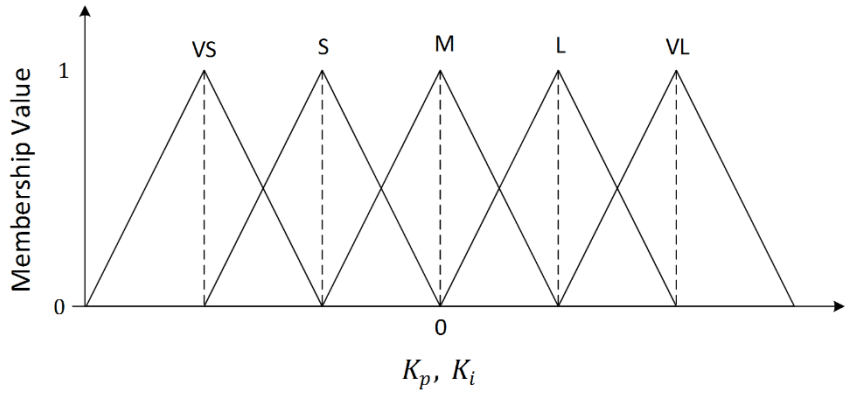

Figure 3.11 Output membership functions for the fuzzy-tuned PI speed controller

A Mamdani-type fuzzy inference system is applied. The minimum operation is used as the 'AND' method for fuzzy implication, and the maximum operation is applied for the union of all outputs.

The rule base is stablished based on the knowledge acquired from the performed computer simulations for the standard PI speed controller, where the following facts were identified

An increment in K_p:

- Increase the rise time
- Reduce overshoot
- Increase ripple in steady state
- Reduce the amplitude of torque disturbances

An increment in K_i:

- Reduce the rise time
- Increase overshoot
- Increase ripple at steady state
- Reduce the area of torque disturbances

The behavior of the speed response according to the signs of the error and the error variation is described as follows

- When $error$ is positive and $\Delta error$ is positive, then the speed gets closer to the reference signal
- When $error$ is positive and $\Delta error$ is negative, then the speed moves away from the reference signal
- When $error$ is negative and $\Delta error$ is positive, then the speed moves away from the reference signal
- When $error$ is negative and $\Delta error$ is negative, then the speed gets closer to the reference signal

In short, when error and error variation have the same sign, the speed gets closer to the reference signal and vice versa.

Based on the above information, the rule base for the fuzzy-tuned PI speed controller is defined according to table 3.2

$error$ \ $\Delta error$	NB	N	Z	P	PB
NB	$K_p \rightarrow$ S $K_i \rightarrow$ M	$K_p \rightarrow$ S $K_i \rightarrow$ M	$K_p \rightarrow$ VS $K_i \rightarrow$ VL	$K_p \rightarrow$ VL $K_i \rightarrow$ VS	$K_p \rightarrow$ VL $K_i \rightarrow$ VS
N	$K_p \rightarrow$ S $K_i \rightarrow$ M	$K_p \rightarrow$ VS $K_i \rightarrow$ S	$K_p \rightarrow$ M $K_i \rightarrow$ M	$K_p \rightarrow$ L $K_i \rightarrow$ S	$K_p \rightarrow$ VL $K_i \rightarrow$ VS
Z	$K_p \rightarrow$ L $K_i \rightarrow$ S	$K_p \rightarrow$ M $K_i \rightarrow$ M	$K_p \rightarrow$ VS $K_i \rightarrow$ VS	$K_p \rightarrow$ M $K_i \rightarrow$ M	$K_p \rightarrow$ L $K_i \rightarrow$ S
P	$K_p \rightarrow$ VL $K_i \rightarrow$ VS	$K_p \rightarrow$ L $K_i \rightarrow$ S	$K_p \rightarrow$ M $K_i \rightarrow$ M	$K_p \rightarrow$ VS $K_i \rightarrow$ S	$K_p \rightarrow$ S $K_i \rightarrow$ M
PB	$K_p \rightarrow$ VL $K_i \rightarrow$ VS	$K_p \rightarrow$ VL $K_i \rightarrow$ VS	$K_p \rightarrow$ VS $K_i \rightarrow$ VL	$K_p \rightarrow$ S $K_i \rightarrow$ M	$K_p \rightarrow$ S $K_i \rightarrow$ M

Table 3.2 Rule-base for the fuzzy-tuned PI speed controller

The weighted average defuzzification method is applied to find the crisp output value. Mathematically, this method is defined as

$$u^{crisp} = \frac{\sum_{k=1}^{m} c[k] f[k]}{\sum_{k=1}^{m} f[k]}$$

where $c[k]$ is the center value of the individual k-output membership function, and $f[k]$ is the corresponding membership value.

3.4. Practical Issues About Digital Control Implementation

In the practical implementation of a digital controller there are unwanted effects which can deteriorate the controller performance. Some of these effects are summarized in this section.

3.4.1. Analog to Digital Acquisition and Filtering

The phase currents of the motor are acquired by the ADC module. The ADC acquisition-conversion has to be fast enough for negligible conversion time relative to the sampling period. With the dsPIC33FJ32MC204 processor, capable of perform conversions up to 1.1Msps, the ADC conversion time is not an issue.

An important point to take into account when performing ADC conversion is the aliasing phenomena. This effect occurs in digital control systems when the sampled signal has frequency components above one-half of the sampling frequency. In this scenario, the sampling process creates new frequency components [32].

When acquiring the phase currents in a motor control system, the current signals can have many harmonic components and noise that can produce the aliasing effect. Thus, a filtering stage is required before the current signals go into the ADC module of the microcontroller.

An easy and practical way to avoid/reduce the aliasing effect is applying a synchronization process between the ADC module and the PWM module. Since the PWM module controls the inverter that feeds the machine, an ADC acquisition performed at the middle point of the PWM period can strongly reduce the effects of aliasing [33], besides other benefits such as

- The measurement is not influenced by disturbances and interferences from the switching of the power semiconductors.
- The average value of the currents can be obtained without any additional calculation.
- All the process can be done by hardware without requiring computer power.

The above process can be accomplished with a center-aligned PWM module and provided that the motor electrical time constant is many times higher than the switching period, so that an almost linear current waveform is obtained during the PWM pulse.

A simple RC low-pass filter can be also applied to provide further filtering. This filter has to be designed with a cut-off frequency higher than the bandwidth of the closed-loop

current control so as not to degrade the transient response of the system. The sampling frequency of the control system has to be higher than the filter cut-off frequency so there is sufficient attenuation above the Nyquist frequency [32].

The following equations can be considered for the RC low-pass filter design

$$F_{c_{filter}} = k * F_{closed-loop}$$

$$F_{sampling} = k * F_{c_{filter}}$$

$$k > 2$$

3.4.2. Phase Delay and ZOH

The digital to analog conversion performed by the space vector PWM algorithm can be modeled using a zero-order hold (ZOH). The ZOH introduces an additional delay in the control loop, approximately equal to half of the sampling period [32]. This delay can affect the stability of the system.

$$G_{ZOH} = \frac{1 - e^{-j\omega T}}{j\omega} \approx e^{-j\omega\frac{T}{2}}$$

3.4.3. Output Voltage Distortion due to Dead Time

The inverter has to be controlled paying special attention to the switches on-states related to the same phase. An opposite state has to be present in those switches all the time in order to avoid a short circuit. To guarantee opposite states in the switches, a dead-time is introduced by the PWM module, hence, for a certain time period, the gating signals of both upper and lower switches are maintained in off state. This dead time generates voltage and current distortions that may result in torque ripples and acoustic noises in the drive system [8].

3.4.4. Digital Signal Processing Delay

Due to the nature of the serial execution of the software in a digital controller, a time delay is inevitable. Because of this delay, the output voltage of the regulator has errors in magnitude and angle. These errors can be neglected when the synchronous speed ω_e is low enough compared to the sampling frequency, for instance $\omega_e \leq \frac{1}{40}\frac{2\pi}{T}$. Otherwise, the errors may result in stability problems of the current-loop regulator [8].

4. PMSM CONTROL SYSTEM IMPLEMENTATION AND TESTING

The work developed in this chapter was already submitted to the “7th International Electric Drives Production Conference and Exhibition 2017”.

4.1. Proteus VSM

Proteus Design Suite is an electronic design automation software tool which includes schematic capture, simulation and PCB layout modules. The most interesting feature of this software is the virtual system modelling (VSM) module. Proteus VSM allows to perform simulations of firmware applied to a microcontroller and digital or analog circuits connected to it, all within a mixed-mode SPICE circuit simulation. Therefore, the design of hardware and software can be performed within the same simulation environment.

Proteus has a good library of analog and digital electronic components, microprocessors, and many useful elements which can be used to construct and represent the mathematical model of a system.

4.2. PMSM Drive Model in Proteus VSM

The permanent magnet synchronous machine model is developed starting with the mathematical model and the equivalent circuits.

4.2.1. Dynamic Stator Equivalent Circuits

Figure 4.1 shows the equivalent circuits of the PMSM in the d-q reference frame.

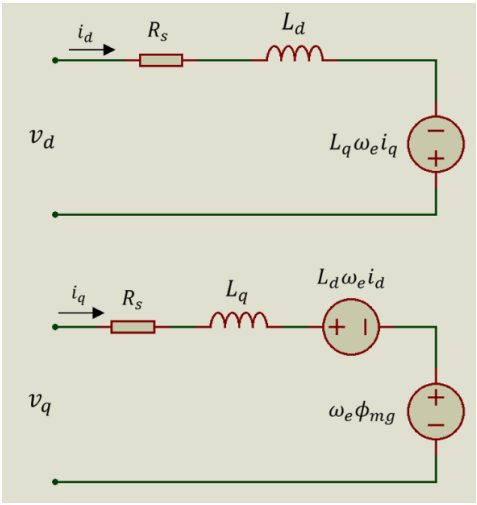

Figure 4.1 Dynamic stator equivalent circuits of the PMSM in the d-q reference frame

As can be seen, the dynamic stator equivalent circuits are conformed by resistors, inductors and parameter-dependent voltage sources. This circuits can be implemented in Proteus using the multiplier voltage source which allows to stablish the output voltage as the product of the two inputs and any arbitrary constant. The symbol of the multiplier voltage source is presented in figure 4.2

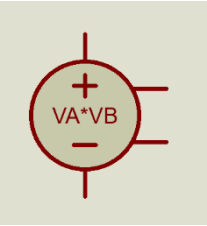

Figure 4.2 Proteus multiplier voltage source element

The analog-graphs feature of Proteus is used to plot the variables of the PMSM model. A voltage probe can be dragged and dropped over the analog-graph window. The PMSM model is developed in order to obtain all variables as voltage magnitudes.

A current controlled voltage source is used to obtain the d-q axis currents as voltage magnitudes. The Proteus implementation of the dynamic stator equivalent circuits is presented in figure 4.3

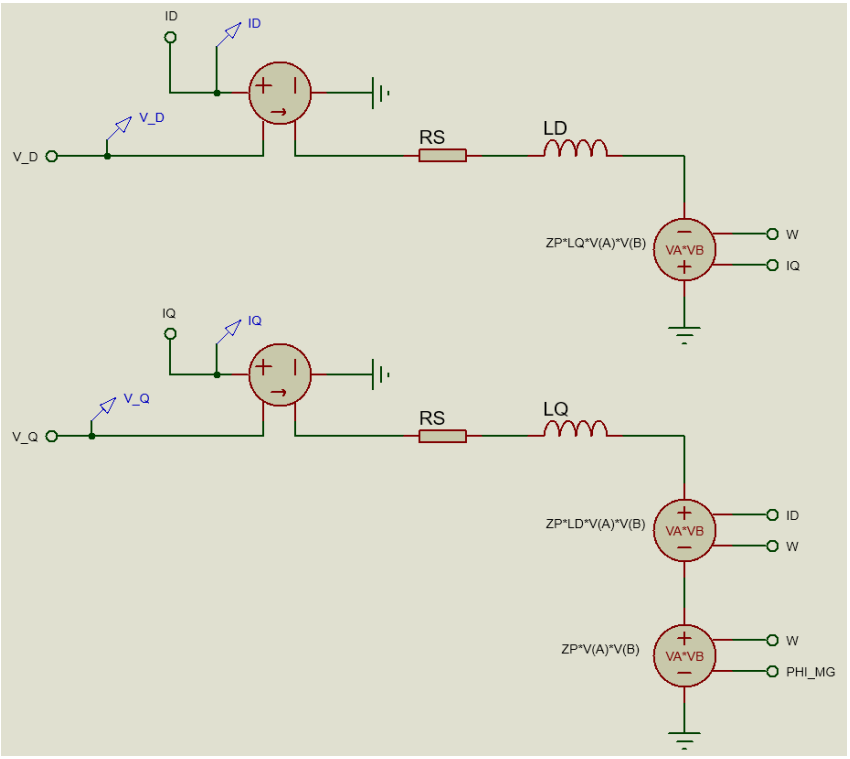

Figure 4.3 Proteus implementation of dynamic stator equivalent circuits of PMSM in d-q reference frame

4.2.2. Electromechanical Dynamic Equivalent Circuit

The electromechanical equation of the PMSM drive is implemented considering an equivalence with an R-C circuit with two current sources as presented in figure 4.4

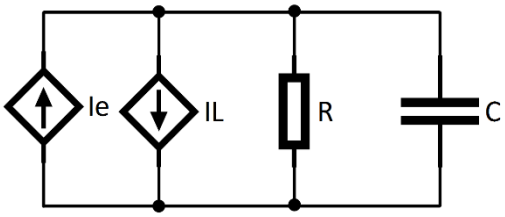

Figure 4.4 Parallel R-C circuit

Applying the Kirchhoff current law, the equation that governs the above circuit is obtained as follows

$$I_e - I_L = i_R + i_c$$

$$I_e - I_L = \frac{v}{R} + C\frac{dv}{dt}$$

$$C\frac{dv}{dt} = I_e - \frac{v}{R} - I_L \tag{4.1}$$

Now, considering the electromechanical equation for the electric machine

$$J_m\frac{d\omega_m}{dt} = T_e - B_v\omega_m - T_L \tag{4.2}$$

By comparing equations (4.1) and (4.2), the electric circuit analogy for the electromechanical equation is evident, with the parameter equivalence given by

$$J_m = C$$

$$B_v = \frac{1}{R}$$

$$\omega_m = v$$

The electromagnetic torque equation as obtained in chapter 2, is given by

$$T_e = \frac{3}{2}Z_p\left[\phi_{mg}i_q + \left(L_d - L_q\right)i_d i_q\right]$$

Based on the above equations, the equivalent circuit for the electromechanical part of the PMSM model is implemented in Proteus, as presented in figure 4.5

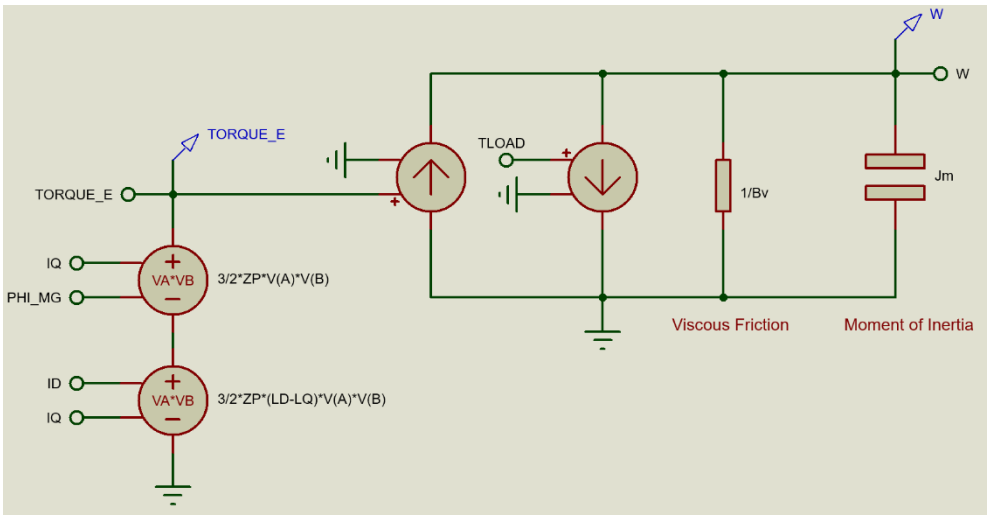

Figure 4.5 Electromechanical dynamic equivalent circuit of the PMSM model

4.2.3. Integrator Circuit for Angular Position

In order to obtain the angular position as a voltage magnitude, the speed signal is passed through an operational amplifier integrator circuit, which is reset every 2π radians. The reset circuit consist of a comparator and an ideal voltage controlled switch. All required elements are available in the Proteus library. The circuit to obtain the angular position as a voltage magnitude is presented in figure 4.6

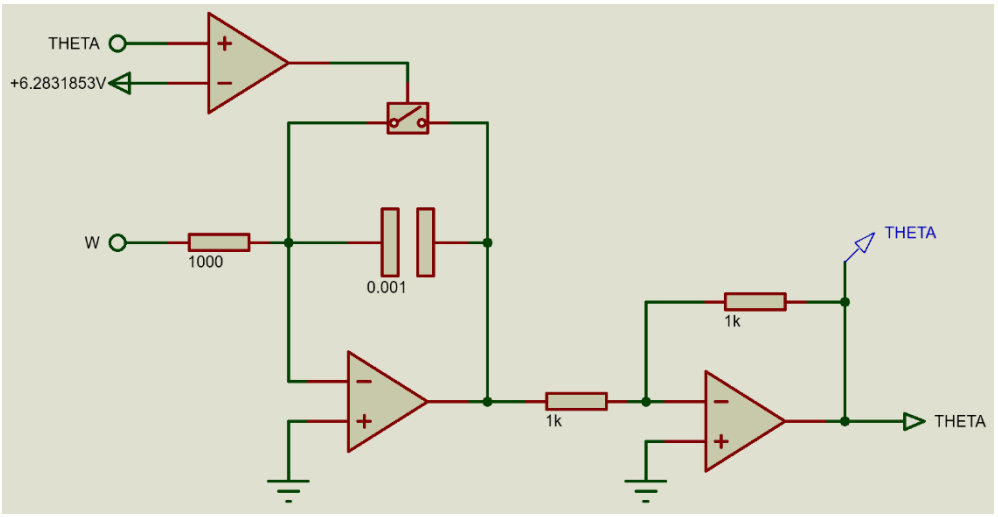

Figure 4.6 Integrator Circuit for Angular Position

4.2.4. Reference Frame Transformations

The three-phase model of the PMSM is implemented applying reference frame transformation circuits. The three-phase input voltage is converted to a bi-phase voltage source in the fixed $\alpha - \beta$ reference frame (Clarke's transformation). Since the machine model is developed in the rotating reference frame, the $\alpha - \beta$ to $d - q$ transformation (Park's transformation) must be applied.

The $d - q$ currents obtained from the model are transformed to the three-phase fixed reference frame applying the corresponding inverse transformations (inverse Parks' and inverse Clark's transformation), thus, completing the three-phase machine model.

The Proteus implementation of the required Clarke's and Parke's transformations are carried out using voltage controlled voltage sources and multiplier voltage sources.

The Clarke's transformation is implemented in Proteus as shown in figure 4.7. For a balanced three-phase source, only two of the three phases are required to perform the transformation. The equations implemented for the Clarke's transformation are

$$i_\alpha = i_a$$

$$i_\beta = \frac{1}{\sqrt{3}} v_a + \frac{2}{\sqrt{3}} v_b$$

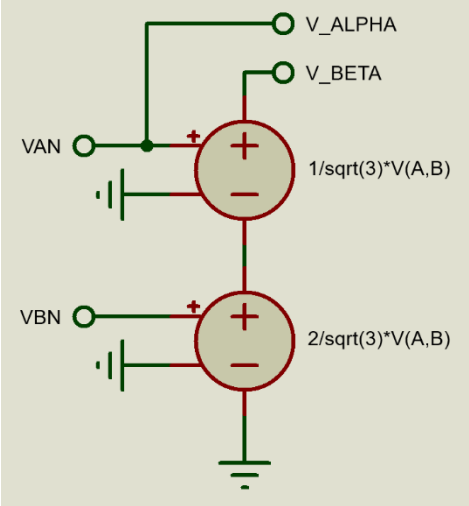

Figure 4.7 Clarke's transformation

The Park's transformation is implemented in Proteus taking advantage of the trigonometric functions that can be placed as product terms in any controlled voltage source. Figure 4.8 shows the implementation

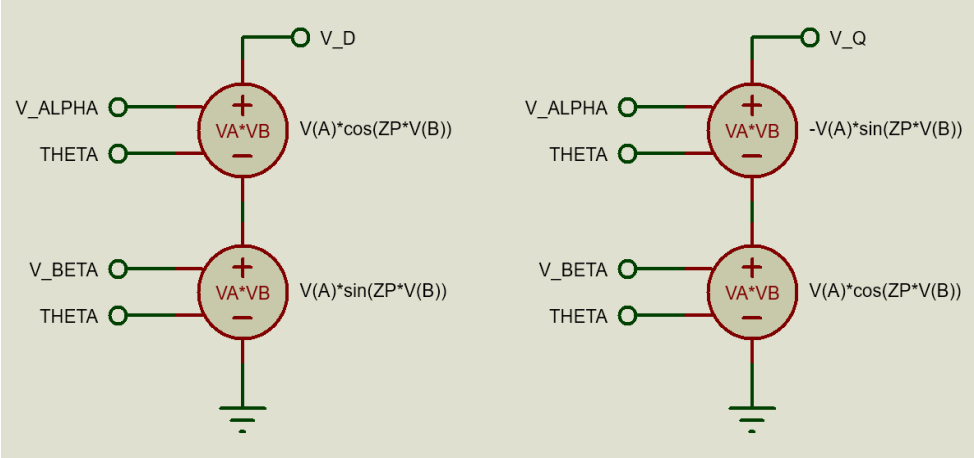

Figure 4.8 Park's transformation

The equations implemented for the Park's transformation are

$$v_d = v_\alpha \cos(\theta_e) + v_\beta \sin(\theta_e)$$

$$v_q = v_\beta \cos(\theta_e) - v_\alpha \sin(\theta_e)$$

The d-q axis currents have to be passed to the three-phase reference frame. Therefore, the inverse Park's and inverse Clarke's transformations have to be applied.

The Proteus implementation of the inverse Park's transformation is presented in figure 4.9. The equations implemented for the inverse Park´s transformation are

$$i_\alpha = i_d \cos(\theta_e) - i_q \sin(\theta_e)$$

$$i_\beta = i_q \cos(\theta_e) + i_d \sin(\theta_e)$$

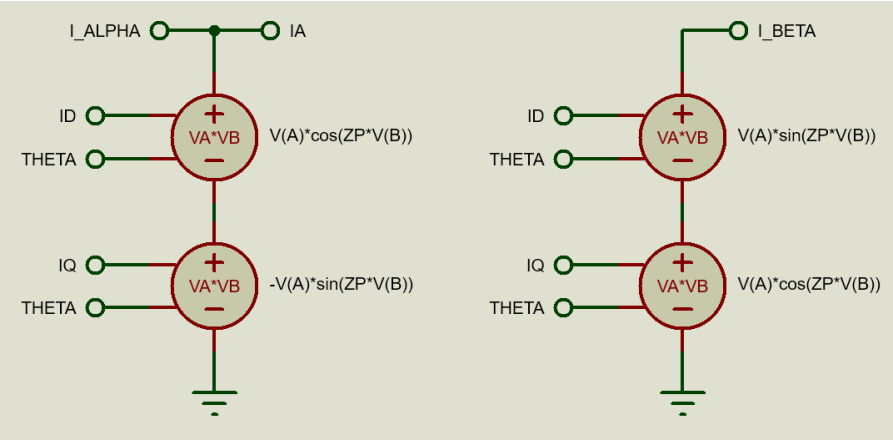

Figure 4.9 Inverse Park's transformation

The inverse Clark's transformation is implemented in Proteus as shown in figure 4.10. The equations for the inverse Clarke's transformation are

$$i_a = i_\alpha$$

$$i_b = -\frac{1}{2}i_\alpha + \frac{\sqrt{3}}{2}i_\beta$$

$$i_c = -\frac{1}{2}i_\alpha - \frac{\sqrt{3}}{2}i_\beta$$

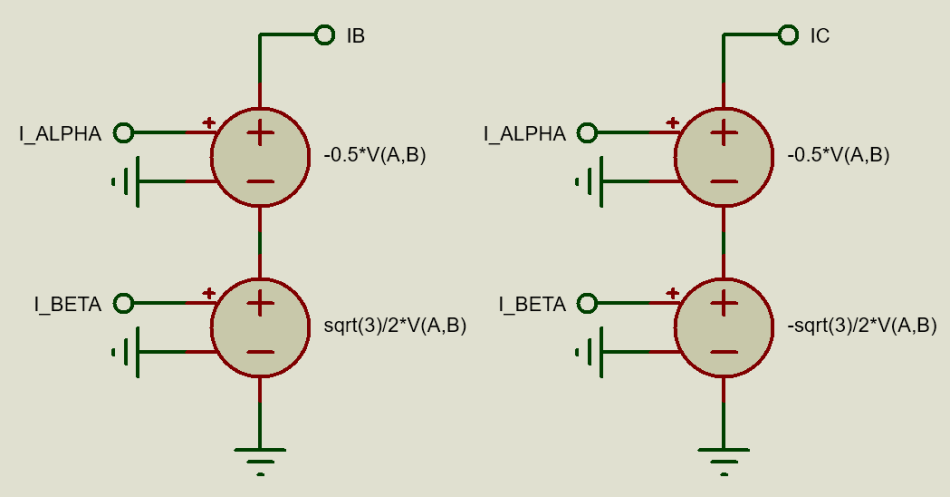

Figure 4.10 Inverse Clarke's transformation

4.2.5. Inverter Model

Considering the basic topology of a three-phase inverter as the one presented in figure 4.11, the following equations are obtained

$$V_a = S_a V_{dc}$$

$$V_b = S_b V_{dc}$$

$$V_c = S_c V_{dc}$$

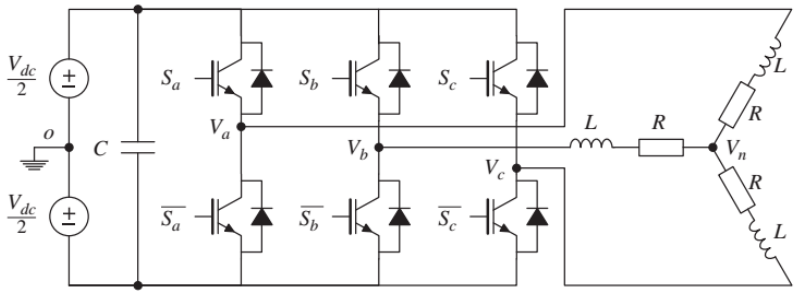

Figure 4.11 Basic topology of a three-phase inverter

Where S_a, S_b, S_c represent the logic state of the three upper switches. With this consideration, the phase-to phase voltages are

$$V_{ab} = V_a - V_b = (S_a - S_b)V_{dc}$$
$$V_{bc} = V_b - V_c = (S_b - S_c)V_{dc}$$
$$V_{ca} = V_c - V_a = (S_c - S_a)V_{dc}$$

Now, considering a balanced load

$$i_{an} + i_{bn} + i_{cn} = 0$$
$$\frac{V_{an}}{Z} + \frac{V_{bn}}{Z} + \frac{V_{cn}}{Z} = 0$$

Finally

$$V_{an} = \frac{V_{dc}}{3}(2S_a - S_b - S_c)$$
$$V_{bn} = \frac{V_{dc}}{3}(2S_b - S_a - S_c)$$
$$V_{cn} = \frac{V_{dc}}{3}(2S_c - S_a - S_b)$$

This simplified representation of a three-phase inverter is used in order to reduce the computational load and the required simulation time. The Proteus implementation of the above equations are presented in figure 4.12

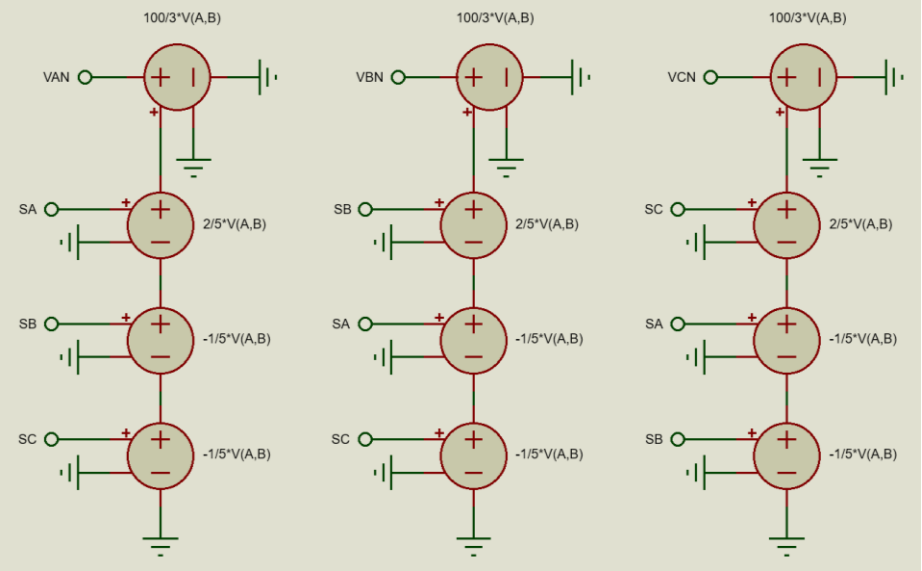

Figure 4.12 Three-phase inverter model

Proteus uses 5V as the 'on' logic state by default, therefore, a factor of 1/5 is required in the inverter implementation.

4.3. dsPIC33FJ32MC204 and Interface Sensors

The microcontroller used to implement the control algorithms is the dsPIC33FJ32MC204. This digital signal controller was selected due to its peripherals availability for motor control applications, such as: center-aligned PWM module, high-speed analog to digital converter and quadrature encoder interface module. Furthermore, the model of this controller is available in the Proteus library, allowing a co-simulation between the electric drive model and the microcontroller code.

4.3.1. ADC Module and Simulated Current Sensor

The analog to digital converter module of the dsPIC33fj32MC204 has a resolution of 10-bits when configured to operate in simultaneous sampling mode. This sampling mode is used because at least two currents have to be acquired at the same time.

A current sensor with 165mV/A is simulated. The operating voltage of the microcontroller is 3.3V, therefore, the scaling factor for the currents is calculated as follows

$$I_{sf} = \frac{1A}{0.165V}\frac{3.3V}{2^{10}} = 0.01953125$$

This scaling factor is represented as a fixed-point number with format Q16.16 (16-bit for the integer part and 16-bit for the fractional part). Therefore, the currents scaling factor used in the source code will be

$$I_{sf} = 0.01953125 * 2^{16} = 1280$$

4.3.2. Simulated Tachogenerator

Since all variables in the Proteus model have voltage magnitudes, a tachogenerator with 1V per 120rad/s is simulated. An optical encoder can also be used to calculate the speed but the tachogenerator is more straightforward to implement in terms of simulation.

The scaling factor for the tachogenerator is calculated as follows. A Q16.16 fixed point representation is used.

$$\omega_{sf} = \frac{120 * 3.3}{2^{10}} * 2^{16}$$

$$\omega_{sf} = 25344$$

4.3.3. Simulated Optical Encoder

The position signal in the Proteus electric drive model has also a voltage magnitude, thus, the ADC module is used to simulate a 1000ppr (pulses per revolution) encoder. The scaling factor, in Q16.16 fixed-point representation, is calculated as

$$\theta_{sf} = \frac{2\pi}{1000} * 2^{16}$$

$$\theta_{sf} = 412$$

4.3.4. Signal Conditioning Circuits

The signal conditioning circuits are not directly implemented in Proteus because the unnecessary computational load added. Instead, voltage controlled voltage sources with the appropriate multiplication factors are used to adequate the signal levels according to the aforementioned simulated sensors. Figure 4.13 shows this implementation

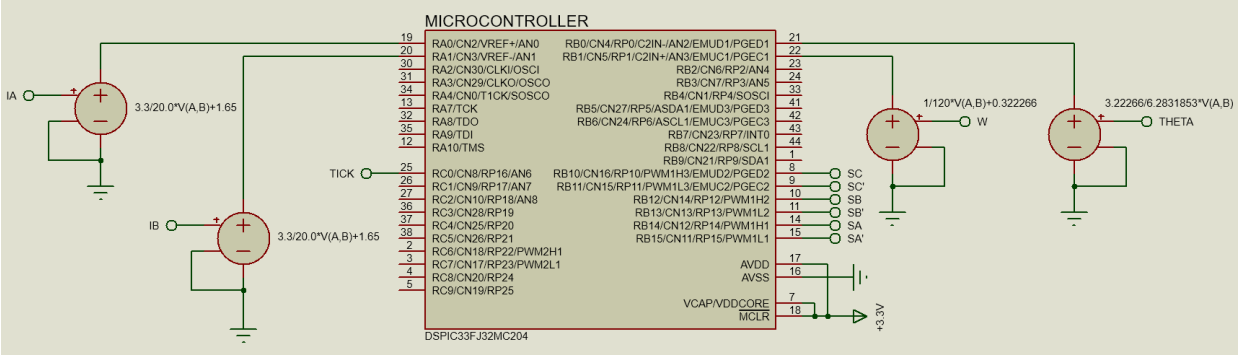

Figure 4.13 dsPIC33FJ32MC204 with emulated conditioning circuits

4.4. Space Vector PWM

The two-level three-phase inverter has eight possible switching states that produce eight voltage vectors as can be seen in table 4.1

	$\overrightarrow{V_0}$	$\overrightarrow{V_1}$	$\overrightarrow{V_2}$	$\overrightarrow{V_3}$	$\overrightarrow{V_4}$	$\overrightarrow{V_5}$	$\overrightarrow{V_6}$	$\overrightarrow{V_7}$
S_a	0	1	1	0	0	0	1	1
S_b	0	0	1	1	1	0	0	1
S_c	0	0	0	0	1	1	1	1

Table 4.1 Switching states of inverter

Table 4.2 shows the inverter output voltage for each vector

	$\overrightarrow{V_0}$	$\overrightarrow{V_1}$	$\overrightarrow{V_2}$	$\overrightarrow{V_3}$	$\overrightarrow{V_4}$	$\overrightarrow{V_5}$	$\overrightarrow{V_6}$	$\overrightarrow{V_7}$
v_a	$-\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$
v_b	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$
v_c	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$-\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$	$\frac{V_{dc}}{2}$

Table 4.2 Output voltage of inverter

There are six vectors ($\overrightarrow{V_1}$ to $\overrightarrow{V_6}$) that generate a non-zero three-phase output voltage (active vectors), and two vectors ($\overrightarrow{V_0}$ and $\overrightarrow{V_6}$) that produce a zero voltage (zero vector).

The Space Vector PWM (SVPWM) modulation technique is applied to derive the on-off time duration for each switch of the inverter. The modulation of the required space vector is obtained by the time average of its nearest active vectors and a zero vector. Figure 4.14 shows an example of a vector that can be modulated with the time average of the active vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ within one sampling period T_s [30].

$$T_s\overrightarrow{V_s^*} = T_1\overrightarrow{V_1} + T_2\overrightarrow{V_2}$$

where T_1 and T_2 are the on-time duration for the active vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ respectively.

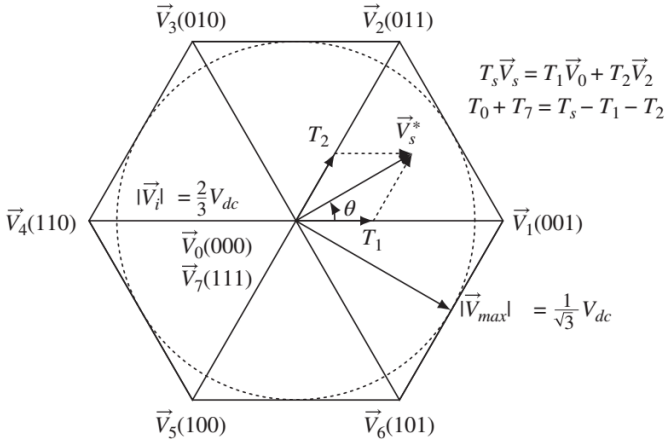

Figure 4.14 Principle of space vector modulation

4.4.1. Equations for turn-on Times

The SVPWM implementation on the dsPIC33FJ32MC204 is carried out configuring the PWM module in center aligned mode, and calculating the turn-on times for the switches according to the equations outlined in reference [34] and described below

$$T_{A-ON} = \begin{cases} \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[-v_\alpha - \frac{v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 1,4 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}[-2v_\alpha]\right) & For\ sectors: 2,5 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[-v_\alpha + \frac{v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 3,6 \end{cases}$$

$$T_{B-ON} = \begin{cases} \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[v_\alpha - \sqrt{3}v_\beta\right]\right) & For\ sectors: 1,4 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[-\frac{2v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 2,5 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[v_\alpha - \frac{v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 3,6 \end{cases}$$

$$T_{C-ON} = \begin{cases} \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[v_\alpha + \frac{v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 1,4 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[\frac{2v_\beta}{\sqrt{3}}\right]\right) & For\ sectors: 2,5 \\ \frac{T_s}{4}\left(1 + \frac{3}{2V_{dc}}\left[v_\alpha + \sqrt{3}v_\beta\right]\right) & For\ sectors: 3,6 \end{cases}$$

To simplify the implementation of the above equations, the following constants are defined

$$C_1 = \frac{T_s}{4}$$

$$C_2 = \frac{3}{2V_{dc}} C_1$$

$$C_3 = \frac{C_2}{\sqrt{3}}$$

$$C_4 = \sqrt{3} C_2$$

$$C_5 = \frac{3}{V_{dc}} C_1$$

$$C_6 = \frac{3}{\sqrt{3} V_{dc}} C_1$$

With these constants, the final equations to be implemented in the microcontroller are

For sectors 1, 4:

$$DC_A = C_1 - C_2 v_\alpha - C_3 v_\beta$$

$$DC_B = C_1 + C_2 v_\alpha - C_4 v_\beta$$

$$DC_C = C_1 + C_2 v_\alpha + C_3 v_\beta$$

For sectors 2, 5:

$$DC_A = C_1 - C_5 v_\alpha$$

$$DC_B = C_1 - C_6 v_\beta$$

$$DC_c = C_1 + C_6 v_\beta$$

For sectors 3, 6:

$$DC_A = C_1 - C_2 v_\alpha + C_3 v_\beta$$

$$DC_B = C_1 + C_2 v_\alpha - C_3 v_\beta$$

$$DC_C = C_1 + C_2 v_\alpha + C_4 v_\beta$$

The algorithm applied for sector identification in the SVPWM implementation is the same as the proposed in the Texas Instruments Application Report SPRA524 [35]. The algorithm is explained as follows

Defining the function

$$sign(x) = \begin{cases} 1 \ \ if \ x > 0 \\ 0 \ \ if \ x \ \leq 0 \end{cases}$$

The following variables are calculated

$$A = sing(v_\beta)$$

$$B = sing(\sqrt{3} v_\alpha - v_\beta)$$

$$C = sign(-\sqrt{3} v_\alpha - v_\beta)$$

$$N = A + 2B + 4C$$

Using the calculated value of N, the sector is determined according to table 4.3

N	Sector
3	1
1	2
5	3
4	4
6	5
2	6

Table 4.3 Sector identification according to N for SVPWM

4.4.2. Voltage Limits

To ensure a modulation within the linear range, the $d-q$ voltages must satisfy

$$\sqrt{v_d^2 + v_q^2} \leq \frac{1}{\sqrt{3}} V_{dc}$$

This constraint corresponds to a circle as presented in figure 4.15

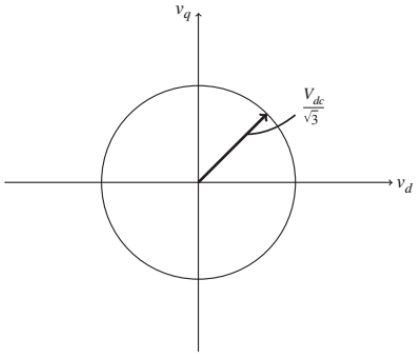

Figure 4.15 Voltage constraint for linear modulation

As can be seen, a square-root is involved in the voltage constraint equation. To reduce calculation time, a rectangular approximation is taken. Assuming a parameter $0 < \epsilon < 1$, where the maximum values for v_d and v_q are determined with

$$v_q^{max} = \epsilon \frac{V_{dc}}{\sqrt{3}}$$

$$v_d^{max} = \sqrt{1-\epsilon^2} \frac{V_{dc}}{\sqrt{3}}$$

Figure 4.16 shows the rectangular approximation constraint

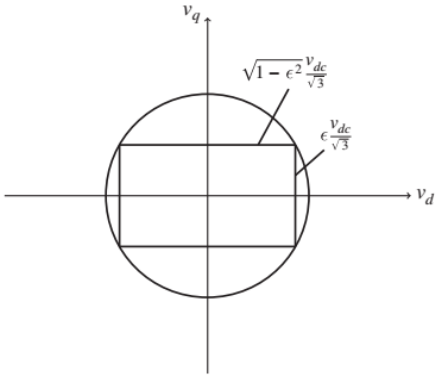

Figure 4.16 Rectangular approximation constraint

With this approximation, the $d-q$ voltages must satisfy

$$-v_d^{max} \leq v_d \leq v_d^{max}$$

$$-v_q^{max} \leq v_q \leq v_q^{max}$$

4.5. Standard PID Controllers Implementation

4.5.1. PI Current Controller

Assuming that the feedback error is $e(t)$ and the feedforward function is $f(t)$, the control signal $u(t)$ from a PI controller is defined as

$$u(t) = K_c e(t) + \frac{K_c}{\tau_I}\int_0^t e(\tau)d\tau + f(t)$$

Differentiating the above equation respect to time

$$\frac{du(t)}{dt} = K_c\frac{de(t)}{dt} + \frac{K_c}{\tau_I}e(t) + \frac{df(t)}{dt}$$

Taking an approximation of the derivatives as a first order difference, at sample time t_i, gives

$$\frac{du(t)}{dt} \approx \frac{u(t_i) - u(t_i - \Delta t)}{\Delta t} = \frac{u(t_i) - u(t_{i-1})}{\Delta t}$$

$$\frac{de(t)}{dt} \approx \frac{e(t_i) - e(t_i - \Delta t)}{\Delta t} = \frac{e(t_i) - e(t_{i-1})}{\Delta t}$$

$$\frac{df(t)}{dt} \approx \frac{f(t_i) - f(t_i - \Delta t)}{\Delta t} = \frac{f(t_i) - f(t_{i-1})}{\Delta t}$$

Using these approximations, the control signal in the discrete form is

$$u(t_i) = u(t_{i-1}) + K_c[e(t_i) - e(t_{i-1})] + \frac{K_c}{\tau_I}e(t_i)\Delta t + f(t_i) - f(t_{i-1})$$

The voltage control signals are calculated based on this equation. The voltage control signals, in the $d-q$ reference frame, are

$$v_d(t_i) = v_d(t_{i-1}) + K_c^d[e_d(t_i) - e_d(t_{i-1})] + \frac{K_c^d}{\tau_I^d}e_d(t_i)\Delta t + f_d(t_i) - f_d(t_{i-1})$$

$$v_q(t_i) = v_q(t_{i-1}) + K_c^q\left[e_q(t_i) - e_q(t_{i-1})\right] + \frac{K_c^q}{\tau_I^q}e_q(t_i)\Delta t + f_q(t_i) - f_q(t_{i-1})$$

$$f_d(t_i) = -\omega_e(t_i)L_q i_q(t_i)$$

$$f_q(t_i) = \omega_e(t_i)L_d i_d(t_i) + \omega_e(t_i)\phi_{mg}$$

$$e_d(t_i) = i_d^*(t_i) - i_d(t_i)$$

$$e_q(t_i) = i_q^*(t_i) - i_q(t_i)$$

4.5.2. PI Speed Controller

The control signal i_q^* given by the PI speed controller is

$$i_q^*(t) = K_c(\omega_e^*(t) - \omega_e(t)) + \frac{K_c}{\tau_I}\int_0^t (\omega_e^*(t) - \omega_e(t))d\tau$$

Taking the derivative of this control signal with $e(t) = \omega_e^*(t) - \omega_e(t)$ gives

$$\frac{di_q^*(t)}{dt} = K_c\frac{de(t)}{dt} + \frac{K_c}{\tau_I}e(t)$$

Approximating the derivatives as

$$\frac{di_q^*(t)}{dt} = \frac{i_q^*(t_i) - i_q^*(t_{i-1})}{\Delta t}$$

$$\frac{de(t)}{dt} = \frac{e(t_i) - e(t_{i-1})}{\Delta t}$$

Using these approximations, the control signal in the discrete form is

$$i_q^*(t_i) = i_q^*(t_{i-1}) + K_c e(t_i) - K_c e(t_{i-1}) + \Delta t\frac{K_c}{\tau_I}e(t_i)$$

Rewriting this equation using $e(t) = \omega_e^*(t) - \omega_e(t)$ gives

$$i_q^*(t_i) = i_q^*(t_{i-1}) + K_c\omega_e^*(t_i) - K_c\omega_e(t_i) - K_c\omega_e^*(t_{i-1}) + K_c\omega_e(t_{i-1}) + \Delta t\frac{K_c}{\tau_I}(\omega_e^*(t_i) - \omega_e(t_i))$$

The proportional control action is usually applied on the feedback signal only, which has an effect of reducing overshoot in the closed-loop set-point response [22]. Therefore, suppressing the terms $K_c\omega_e^*(t_i)$ and $K_c\omega_e^*(t_{i-1})$ gives

$$i_q^*(t_i) = i_q^*(t_{i-1}) - K_c\omega_e(t_i) + K_c\omega_e(t_{i-1}) + \Delta t\frac{K_c}{\tau_I}(\omega_e^*(t_i) - \omega_e(t_i))$$

$$i_q^*(t_i) = i_q^*(t_{i-1}) - K_c\big(\omega_e(t_i) - \omega_e(t_{i-1})\big) + \Delta t\frac{K_c}{\tau_I}(\omega_e^*(t_i) - \omega_e(t_i))$$

4.5.3. P Position Controller

The position control consists of a simple proportional controller, with the addition of a feedforward speed signal calculated as the derivative of the reference angular position. The control signal is calculated as

$$\omega_e^* = K_p(\theta_e^* - \theta_e) + \widehat{\omega}_e^*$$

Where

$$\widehat{\omega}_e^* = \frac{d\theta_e^*}{dt}$$

Using and inner speed control loop adds robustness against parameter variations in the rotor flux linkages.

4.6. Fuzzy-Logic Controller Implementation

In chapter 3, a hybrid control system design based on fuzzy-logic controllers were presented. The algorithm to implement a general fuzzy controller is presented here. This algorithm will be applied to implement the required controllers for the hybrid PMSM drive system.

Considering a fuzzy-logic controller with two inputs namely: error and error variation; one output, and membership functions with 50% overlap symmetrically distributed across the universe of discourse. With these considerations, each input will correspond to only two membership functions.

For instance, consider figure 4.17 which shows a case when the error crisp input cuts the membership function 'PM' at point a, and the membership function 'PS' at point b; and the error variation crisp input cuts the membership function 'Z' at point c, and the membership function 'NS' at point d. The points a, b, c, d are simply calculated by linear interpolation.

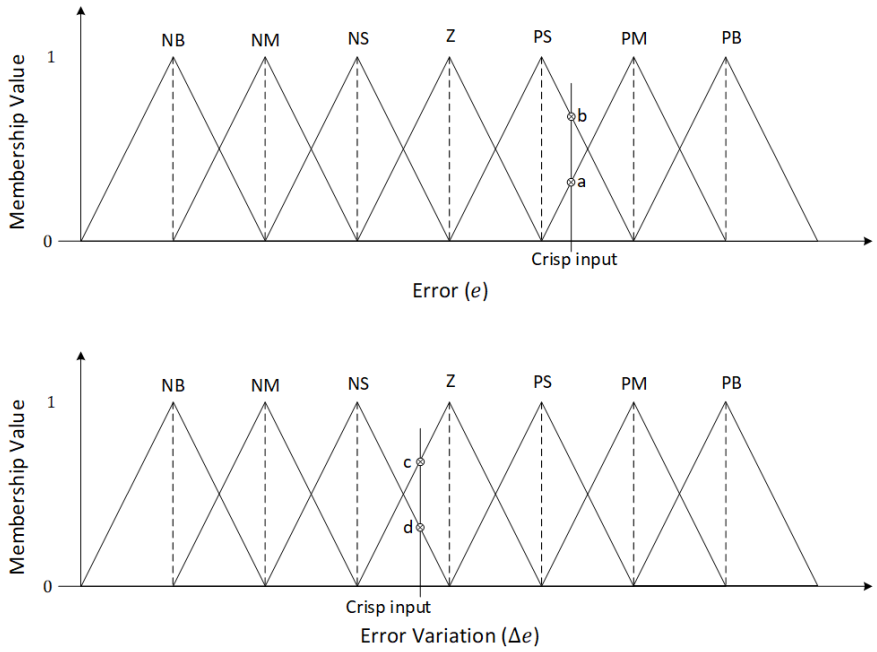

Figure 4.17 Example of input fuzzification

Defining the following variables

$omfA$ = output membership function A

$omfB$ = output membership function B

$omfC$ = output membership function C

$omfD$ = output membership function D

$omvA$ = output membership value A

$omvB$ = output membership value B

$omvC$ = output membership value C

$omvD$ = output membership value D

$ocvA$ = output center value A

$ocvB$ = output center value B

$ocvC$ = output center value C

$ocvD$ = output center value D

Considering linguistic fuzzy rules based on the Macvicar-Whelan matrix, the obtained output membership functions are.

If $error$ is **PM** and $\Delta error$ is **Z** then output is **PM** → $omfA = PM$

If $error$ is **PM** and $\Delta error$ is **NS** then output is **PS** → $omfB = PS$

If $error$ is **PS** and $\Delta error$ is **Z** then output is **PS** → $omfC = PS$

If $error$ is **PS** and $\Delta error$ is **NS** then output is **Z** → $omfD = Z$

The output membership function values are calculated considering the minimum value of the antecedent membership functions, that is

$$omvA = \min(a, c)$$

$$omvB = \min(a, d)$$

$$omvC = \min(b, c)$$

$$omvD = \min(b, d)$$

Since the weighted average method will be applied to obtain the crisp output (defuzzification), the center values of the output membership functions are required.

$$ocvA = center(omfA)$$

$$ocvA = center(omfB)$$

$$ocvA = center(omfC)$$

$$ocvA = center(omfD)$$

Finally, the crisp output value is calculated as

$$crisp\ output = \frac{(ocvA)(omvA) + (ocvB)(omvB) + (ocvC)(omvC) + (ocvD)(omvD)}{omvA + omvB + omvC + omvD}$$

4.7. Parameters Variation

Machine parameters will vary during normal operation, principally due to temperature variations. Stator resistance and permanent magnet flux are the most affected parameters. In order to observe the effects of these parameter variations, step changes are applied to resistance and flux in the Proteus PMSM electric drive model.

4.7.1. Stator Resistance Variation

Since the stator resistance sensitivity is overcome in the current control loop, a considerable step variation is required to observe the effects of resistance variation. Using a voltage controlled switch, a resistor with approximately 200% of the nominal stator resistance value is placed in series with the nominal resistance of the machine. Figure 4.18 shows this implementation in the $d-q$ model of the PMSM.

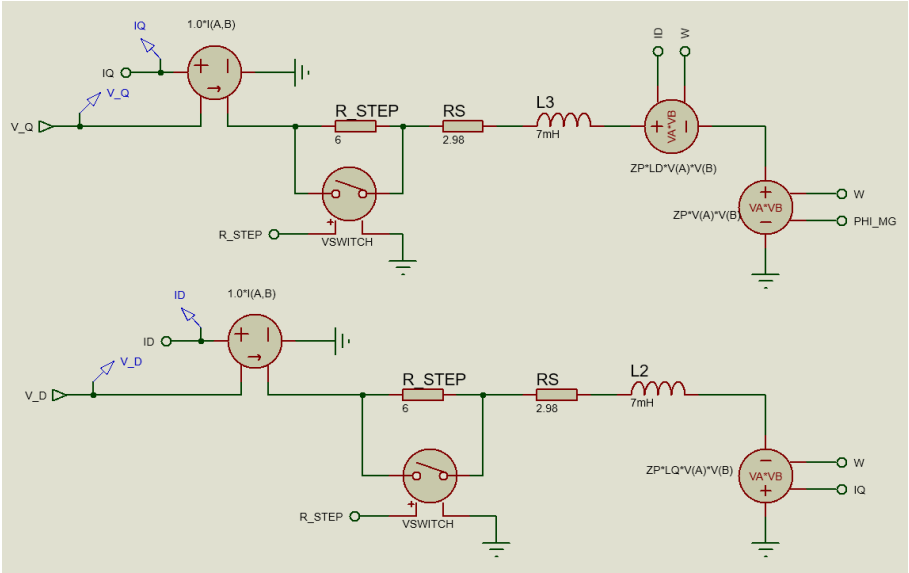

Figure 4.18 Stator resistance step variation model

According to figure 4.18, 'R_STEP' added in series with the stator resistance is controlled by the voltage controlled switch. The switch is initially closed, thus, only the nominal resistance value is effectively placed in the model. When the switch is open, a stator resistance increase is produced. In this way, a step resistance variation is simulated.

4.7.2. Permanent Magnet Flux Variation

The effect due to the loss of magnetism with temperature variations is predominant compared to the effect of stator resistance variation on the performance of the drive system. The sensitivity of residual flux density in magnets for 100ºC rise in temperature

in ferrite, neodymium and samarium cobalt magnet are -19%, -12% and -3%, respectively, from their nominal values [7].

A ferrite magnet is considered, so a -19% step flux variation will be used in the simulation. For the Proteus implementation, the nominal value of the permanent magnet flux linkage is passed through a voltage multiplier. The first factor of the multiplier will be the nominal flux linkage value. The second factor of the multiplier is connected to a switch for enabling or disabling the flux step variation. The implementation is presented in figure 4.19

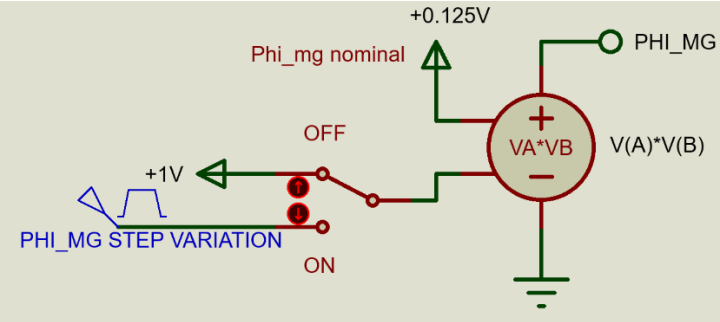

Figure 4.19 Permanent magnet flux step variation model

When the switch is in the 'off' position, the multiplier factor is 1 and thus, the nominal value of the permanent magnet flux linkage is taken. When the switch is placed in the 'on' position, a step signal with amplitude equal to 0.81 (corresponding to a -19% variation) is selected as the multiplier factor.

4.8. Simulation Results and Comparison

All the controller parameters are further tuned via simulation in order to achieve approximately the characteristics presented in table 4.4

Control Loop	Bandwidth	Control Sampling Frequency
Current	900Hz	16KHz
Velocity	50Hz	4KHz
Position	10Hz	1KHz

Table 4.4 Bandwidths and frequencies of the PMSM control loops

The bandwidth is estimated using the rise time with the following formula

$$Bandwidth = \frac{0.35}{t_{rise}}$$

The control sampling frequency is configured upon the base of the PWM frequency using the PWM interrupt period. Counter variables that divide the PWM interrupt period are used to obtain the corresponding sampling frequency for velocity and position.

4.8.1. Current Control Loop

The PI current controller is implemented and tested in Proteus. Setting the q-axis reference current at 1A and the d-axis reference current at 0A. The simulation result is presented in figure 4.20.

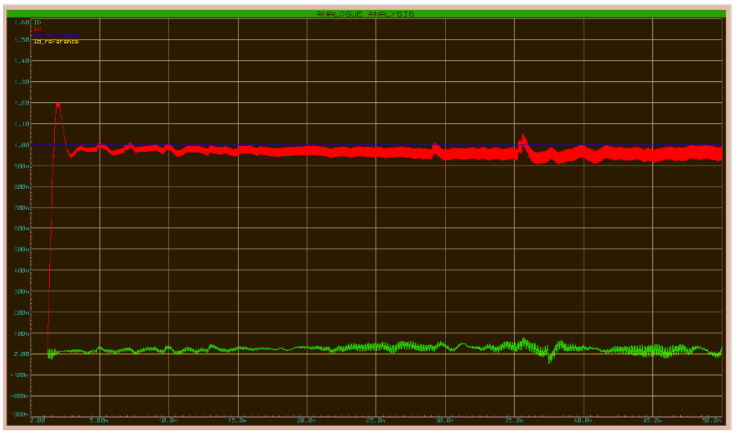

Figure 4.20 PI current controller test

4.8.2. Speed Control Loop

The speed control loop is implemented and tested for two controllers namely: the PI controller and the fuzzy-tuned PI controller.

4.8.2.1. PI Speed Controller

The PI speed controller is tested setting a reference of 100 rad/s. The simulation result is presented in figure 4.21. The y-axis is configured to display the speed response between 95 and 105 rad/s.

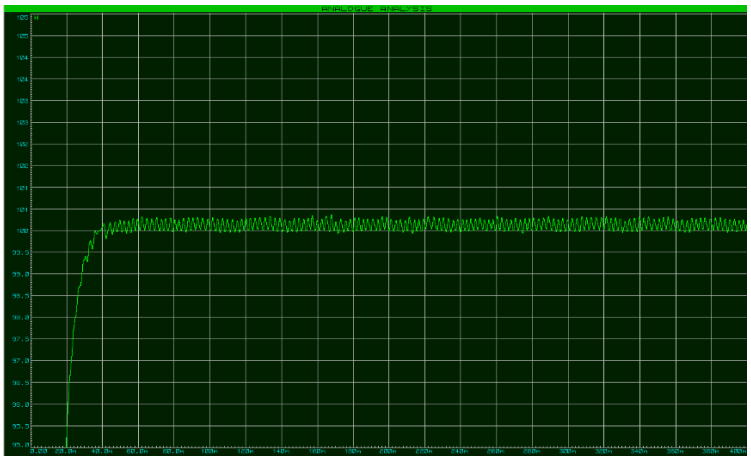

Figure 4.21 PI speed controller test

4.8.2.2. Fuzzy-tuned PI Speed Controller

The fuzzy-tuned PI controller is tested with the same graph configurations as for the standard PI controller. The simulation result is presented in figure 4.22

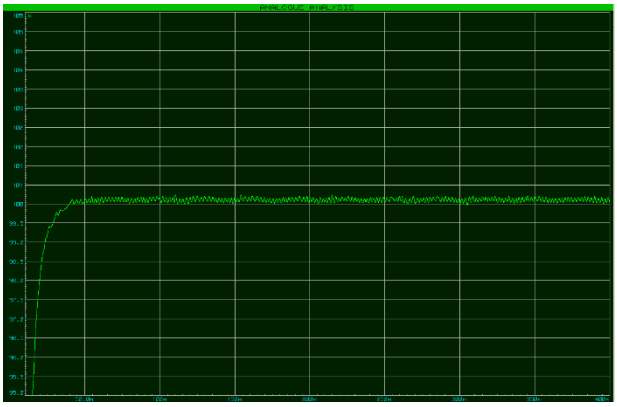

Figure 4.22 Fuzzy-tuned PI speed controller test

As can be seen, the fuzzy-tuned PI speed controller has less ripple in the steady-state response compared with the standard PI controller.

4.8.3. Position Control Loop

The final control objective is to regulate the position response of the electric drive. The position control loop is tested with two controllers namely the standard proportional controller and the fuzzy-logic position controller.

4.8.3.1. Proportional Position Controller

The proportional position controller is tested configuring the y-axis graph display with a range between 5.9 and 6.1 radians, for a command reference signal of 6 radians. The position response is presented in figure 4.23

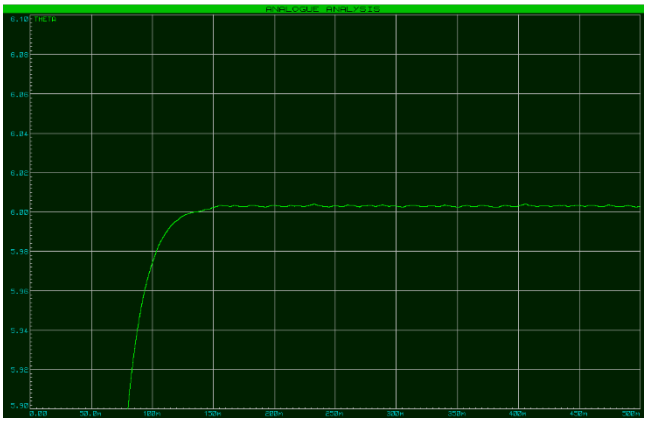

Figure 4.23 Response of proportional position controller

4.8.3.2. Fuzzy-Logic Position Controller with Proportional Action

The fuzzy-logic position controller with proportional action is tested under the same graph configurations as for the standard proportional controller. Figure 4.24 shows the position response for this controller. By comparing figure 4.23 and 4.24, an improve in the steady state response can be observed for the fuzzy-logic controller with proportional action. In terms of rise time, both controllers have similar performances.

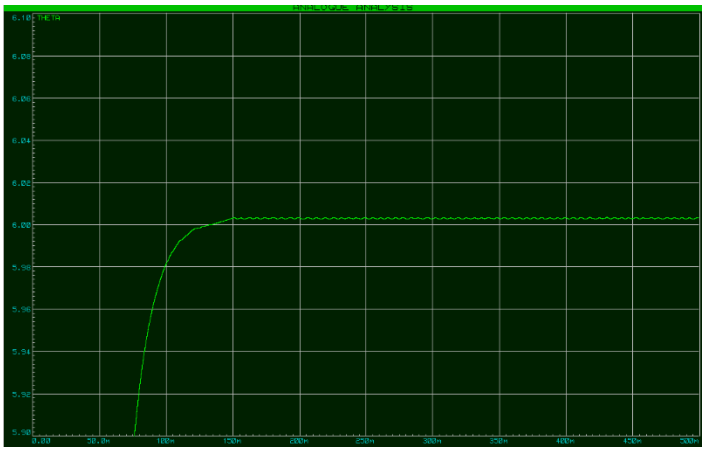

Figure 4.24 Response of fuzzy-logic position controller with proportional action

4.8.4. Controllers Comparison

The standard PID-based and the fuzzy-logic-based controllers are compared by means of error indicators namely integral square error, integral absolute error and root mean squared error. These error measurements are described by the following equations

- Integral square error

$$ISE = \int_{t_0}^{t_1} e^2 \, dt$$

- Integral absolute error

$$IAE = \int_{t_0}^{t_1} |e| \, dt$$

- Root mean square error

$$RMS = \sqrt{\frac{1}{t_1 - t_0} \int_{t_0}^{t_1} e^2 \, dt}$$

The comparison is carried out for speed control and for position control. In order to test the performance of the controllers, four different conditions are considered for simulation, which are:

a) No disturbance or perturbation
b) Periodical torque disturbance with 100ms period
c) 200% stator resistance variation step
d) -19% permanent magnet flux variation step

4.8.4.1. Speed Controllers Comparison

The standard PI speed controller, and the fuzzy tuned PI speed controller are simulated according to the conditions described previously. Table 4.5 summarizes the results obtained for the speed controllers. To facilitate comparison, bar charts are also included and presented in figures 4.25 to 4.27.

	STANDAR PI SPEED CONTROLLER			FUZZY TUNNED PI SPEED CONTROLLER		
	ISE	**IAE**	**RMS**	**ISE**	**IAE**	**RMS**
No disturbance	1.10429	0.19942	0.81546	0.83828	0.15454	0.67375
Torque disturbance	1.23717	0.28973	0.96921	1.22154	0.35060	1.15376
Resistance 200% step variation	1.09831	0.19235	0.81367	0.83885	0.15441	0.65768
Flux -19% step variation	1.09919	0.19310	0.82687	0.84888	0.15567	0.66763

Table 4.5 Comparison data for speed controllers

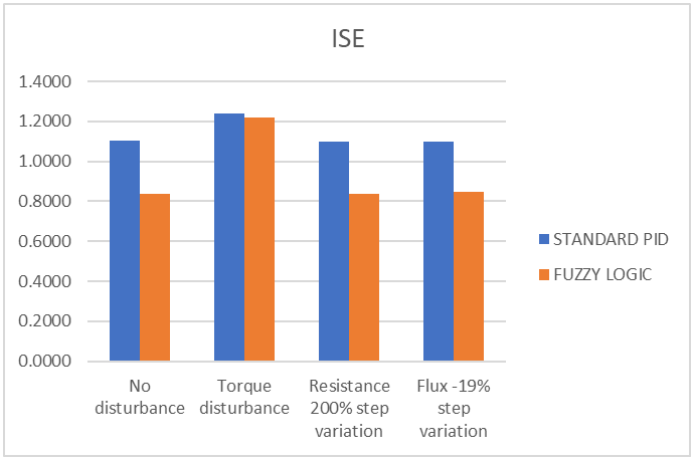

Figure 4.25 Comparison of speed controllers. ISE error indicator

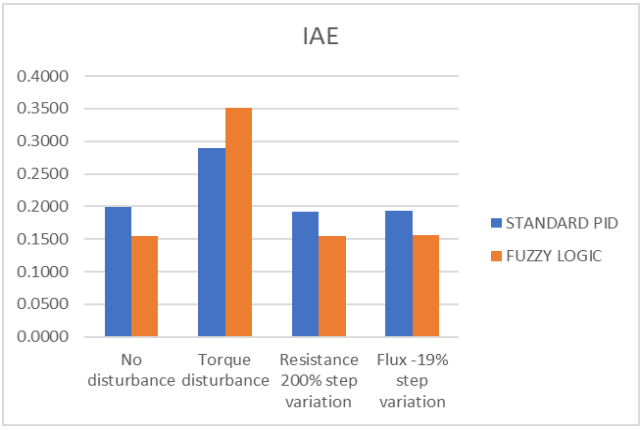

Figure 4.26 Comparison of speed controllers. IAE error indicator

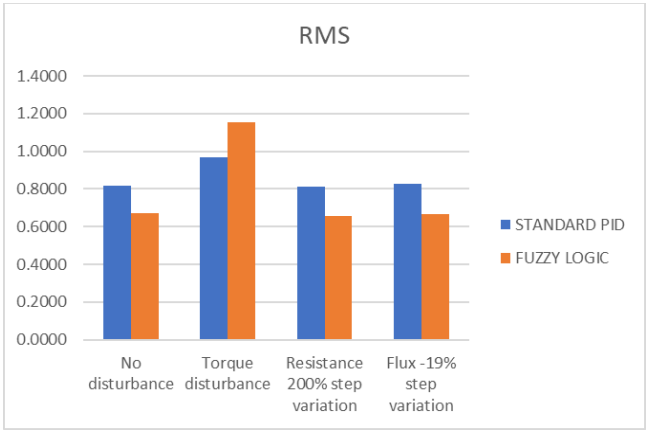

Figure 4.27 Comparison of speed controllers. RMS error indicator

A general view of the above data shows that the fuzzy-tuned PI speed controller has better performance compared with the conventional PI speed controller. Although for a periodic torque disturbance the standard PI speed controller slightly outperforms the fuzzy-tuned PI controller.

4.8.4.2. Position Controllers Comparison

As for the speed controllers, the standard proportional position controller, and the fuzzy-logic position controller, are simulated taking into account all the conditions described in point 4.8.4. Table 4.6 Summarizes the results. Bar charts are also included in figures 4.28 to 4.30.

	STANDARD P POSITION CONTROLLER			FUZZY POSITION CONTROLLER		
	ISE	**IAE**	**RMS**	**ISE**	**IAE**	**RMS**
No disturbance	1.62E-07	2.18E-04	6.31E-04	6.26E-08	1.41E-04	3.88E-04
Torque disturbance	2.89E-06	6.54E-04	2.64E-03	7.36E-06	1.34E-03	4.29E-03
Resistance 200% step variation	3.13E-07	3.04E-04	7.37E-04	6.84E-08	1.47E-04	3.94E-04
Flux -19% step variation	1.86E-07	2.29E-04	6.77E-04	6.26E-08	1.41E-04	3.88E-04

Table 4.6 Comparison data for position controllers

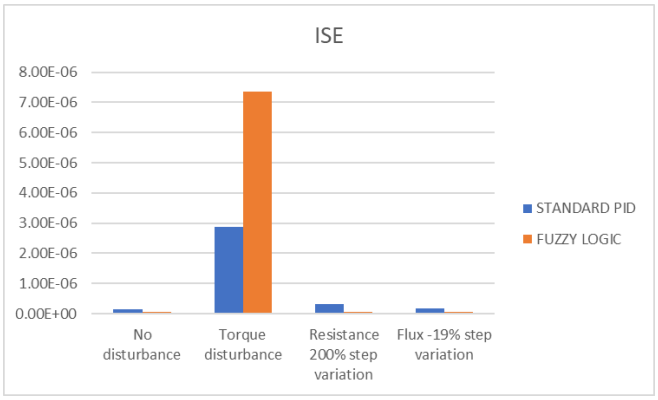

Figure 4.28 Comparison of position controllers. ISE error indicator

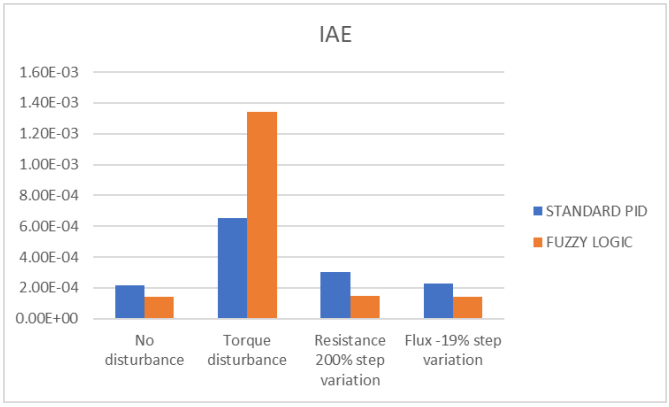

Figure 4.29 Comparison of position controllers. IAE error indicator

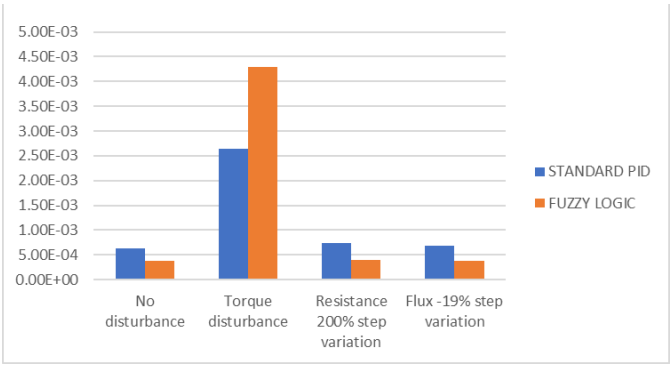

Figure 4.30 Comparison of position controllers. RMS error indicator

As can be seen in the bar charts, for the periodical torque disturbance condition, the error for the fuzzy-logic position controller is bigger than the error for the standard proportional controller. Nevertheless, the opposite occurs for the rest of conditions. Another point to be noted are the error magnitudes which are lesser than 0.005 for a 6 radians reference (error < 0.08%), suggesting a good performance for both controllers.

4.9. Practical Implementation for a BLDC motor

A brushless DC motor (BLDC) is used to perform the real-world validation of the proposed controllers. Although the BLDC motor is slightly different from the PMSM, both are synchronous machines, and therefore, the control algorithms can be implemented in any case.

The main difference between PMSM and BLDC motors is the winding type of each machine. The PMSM has a distributed winding which produces a sinusoidal air-gap flux density, whereas the BLDC motor has a concentrated winding which generates a trapezoidal air-gap flux density.

Torque control can be achieved through field oriented control for both types of motors, although depending on the application, the commutation method can be simpler than FOC e.g. sinusoidal or trapezoidal.

4.9.1. BLDC Motor Electrical Parameters Measurement

The BLDC motor was extracted from a floppy disk unit and therefore, there is not technical data available for this motor. The electrical parameters will be measured using a multimeter and a digital oscilloscope. The Application Note AN4680 from NXP Freescale Semiconductor [36] are used as a reference for the parameter measurement methodology.

4.9.1.1. Stator Resistance

The stator resistance is measured directly with a digital multimeter configured as an ohmmeter. The neutral point connection of the motor is accessible and thus, the resistance is measured directly from each phase to the neutral point. The measured resistance is

$$R_s = 3.3\Omega$$

To consider temperature effects, the above resistance value is recalculated using the cooper temperature coefficient for a 50ºC estimated operational point and with a room temperature of 22ºC. The resistance with these assumptions is

$$R_s = 3.96\Omega$$

4.9.1.2. Synchronous Inductances

The d-q inductances are measured using a digital oscilloscope, a DC voltage source, and a 1Ω shunt resistor, connected as presented in the schematic diagram of figure 4.31:

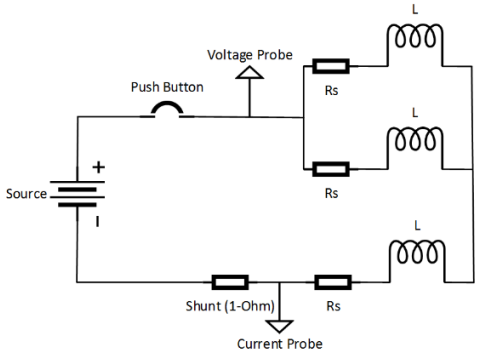

Figure 4.31 Schematic diagram for synchronous inductance measurement

The d-axis inductance is measured as follows

- Perform a rotor alignment with the d-axis (phase A +, phase B -, phase C-)
- Lock the rotor shaft
- Connect the phase B and phase C to the positive potential, and the phase A through the shunt resistor to the negative potential.
- Apply a voltage step with the push button
- Measure the time constant τ (time until current reaches 63.2% of its final value)
- Calculate the d-axis inductance with

$$L_d = \frac{3}{2}\tau R_{eq}$$

  Where R_{eq} is the resistance viewed at the terminals where the voltage source will be applied, and is measured with a multimeter.

The q-axis inductance is measured in a similar way, as follows

- Perform a rotor alignment with the q-axis (connect phase B +, phase C- and let phase A floating)
- Lock the rotor shaft
- Connect the phase A to the positive potential and phase B and phase C together through the shunt resistor to the negative potential.
- Apply a voltage step with the push button
- Measure the time constant τ (time until current reaches 63.2% of its final value)
- Calculate the q-axis inductance with

$$L_q = \frac{3}{2}\tau R_{eq}$$

Applying the above steps, the synchronous inductances are

$$L_d = \frac{3}{2}(720us)(6.4) = 3.072\ mH$$

$$L_d \approx 3.1\ mH$$

$$L_q = \frac{3}{2}(840us)(6.4) = 3.584\ mH$$

$$L_q \approx 3.6\ mH$$

4.9.1.3. Number of Poles

The number of pole-pairs of the BLDC motor are determined by performing a rotor alignment with the d-axis (phase A +, phase B -, phase C-), rotating the motor shaft by hand, and counting the number of stable positions within a complete revolution. The number of stable position will be the number of pole pairs of the machine. The measured pole pairs for the available BLDC are

$$Z_p = 10$$

4.9.1.4. Motor Constant

The motor constant is measured using an oscilloscope and a tachometer. An auxiliary DC motor is used to rotate the BLDC motor. Several measurements are taken in order to calculate the motor constant, namely: line-to-line voltage, line-to-neutral voltage, peak-to-peak voltage, RMS voltage, period, and angular speed. The measured values are presented in Table 4.7

Angular Speed [rad/s]	**Line-to-Line Voltage [V]**		**Line-to-Neutral Voltage [V]**		**Period [ms]**
	RMS	*Peak-Peak*	*RMS*	*Peak-Peak*	
351	8.95	24.4	5.18	13	1.8

Table 4.7 Measured values for back-EMF-constant calculation

With these measured parameters, the motor constant is calculated in several ways as follows:

- **For line-to-line voltages**
  a) Considering the RMS voltage
     - With the measured period T_c

$$K_{\mathrm{T}} = \frac{\sqrt{2}}{\sqrt{3}} V_{rms} \frac{T_c}{2\pi}$$

$$K_{\mathrm{T}} = \frac{\sqrt{2}}{\sqrt{3}}(8.95)\left(\frac{0.0018}{2\pi}\right)$$

$$K_{\mathrm{T}} = 0.002094\ \left[\frac{V-s}{rad}\right]$$

- With the measured angular speed ω_m

$$K_{\mathrm{T}} = \frac{\sqrt{2}}{\sqrt{3}} \frac{V_{rms}}{Z_p \omega_m}$$

$$K_{\mathrm{T}} = \frac{\sqrt{2}}{\sqrt{3}} \frac{(8.95)}{(10)(351)}$$

$$K_{\mathrm{T}} = 0.002082 \left[\frac{V-s}{rad}\right]$$

b) Considering the peak-to-peak voltage

- With the measured period T_c

$$K_{\mathrm{T}} = \frac{V_{pp} T_c}{4\pi\sqrt{3}}$$

$$K_{\mathrm{T}} = \frac{(24.4)(0.0018)}{4\pi\sqrt{3}}$$

$$K_{\mathrm{T}} = 0.002018 \left[\frac{V-s}{rad}\right]$$

- With the measured angular speed ω_m

$$K_{\mathrm{T}} = \frac{V_{pp}}{2\sqrt{3} Z_p \omega_m}$$

$$K_{\mathrm{T}} = \frac{24.4}{2\sqrt{3}(10)(351)}$$

$$K_{\mathrm{T}} = 0.002007 \left[\frac{V-s}{rad}\right]$$

➢ **For line-to-neutral voltages**

c) Considering the RMS voltage

- With the measured period T_c

$$K_{\mathrm{T}} = \sqrt{2}\, V_{rms} \frac{T_c}{2\pi}$$

$$K_{\mathrm{T}} = \sqrt{2}\, (5.18) \left(\frac{0.0018}{2\pi}\right)$$

$$K_{\mathrm{T}} = 0.002099 \left[\frac{V-s}{rad}\right]$$

- With the measured angular speed ω_m

$$K_{\mathrm{T}} = \sqrt{2} \frac{V_{rms}}{Z_p \omega_m}$$

$$K_{\mathrm{T}} = \sqrt{2} \frac{(5.18)}{(10)(351)}$$

$$K_{\mathrm{T}} = 0.002087 \left[\frac{V-s}{rad}\right]$$

d) Considering the peak-to-peak voltage

- With the measured period T_c

$$K_{\mathrm{T}} = \frac{V_{pp} T_c}{4\pi}$$

$$K_{\mathrm{T}} = \frac{(13)(0.0018)}{4\pi}$$

$$K_{\mathrm{T}} = 0.001862 \left[\frac{V-s}{rad}\right]$$

- With the measured angular speed ω_m

$$K_{\mathrm{T}} = \frac{V_{pp}}{2 Z_p \omega_m}$$

$$K_{\mathrm{T}} = \frac{13}{2(10)(351)}$$

$$K_{\mathrm{T}} = 0.001852 \left[\frac{V-s}{rad}\right]$$

Table 4.8 summarizes the above results

Measurements			$K_{\mathrm{T}} \left[\frac{V-s}{rad}\right]$
line-to-line	RMS	Tc	0.002094
		w	0.002082
	peak-to-peak	Tc	0.002018
		w	0.002007
line-to-neutral	RMS	Tc	0.002099
		w	0.002087
	peak-to-peak	Tc	0.001862
		w	0.001852

Table 4.8 BLDC motor constant measurements

As can be seen, the results are very similar except for the calculated values using the peak-to-peak voltage in the line-to-neutral voltage measurement, which present a considerable difference. This difference comes from the fact that the waveform obtained from the line-to-neutral voltage has a poor approximation to a sine wave.

The motor constant is taken as the average of the calculated values, discarding the values from the line-to-neutral voltage with peak-to-peak measurement. With the above considerations, the final value for the motor constant is

$$K_{\mathrm{T}} = 0.002065 \left[\frac{V-s}{rad}\right]$$

4.9.2. BLDC Motor Testbed

The testbed is composed by a BLDC motor with 18 slots and 10 pole pairs, an optical encoder with 360 pulses per revolution, a three-phase inverter, and a control breadboard based on the dsPIC30F2010 microcontroller. A serial communication is used to connect a computer with the microcontroller. Figure 4.32 shows the assembled testbed.

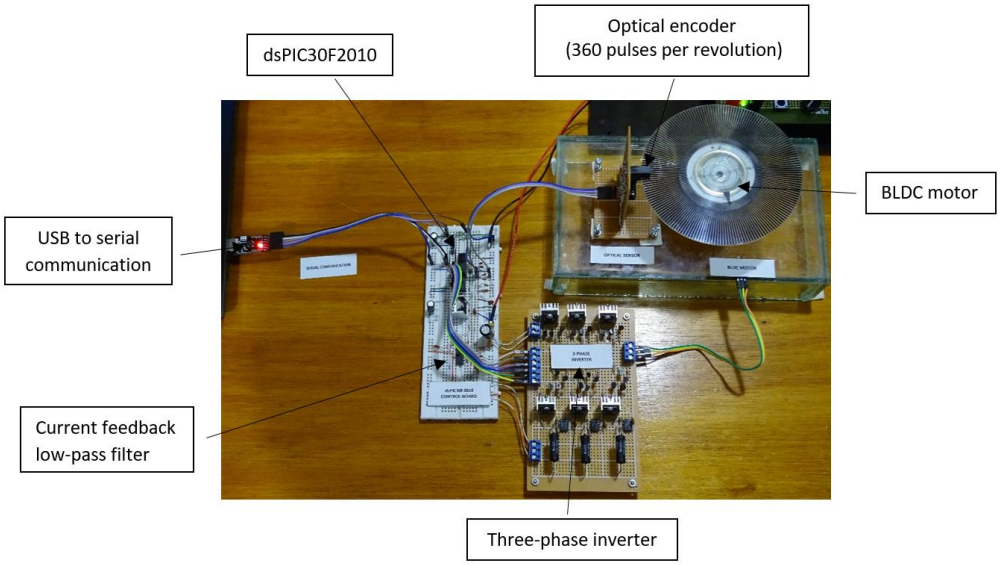

Figure 4.32 BLDC motor testbed

The available optical encoder does not have a quadrature output; therefore, it does not provide information about the direction of rotation. This encoder limitation restricts the implementation of a position controller. Nevertheless, the implementation of a speed controller is still possible.

4.9.3. Torque Control with FOC Commutation

The final goal in FOC is to regulate the machine flux producing component and torque producing component independently. To achieve this, a very precise angular position feedback is required.

The electrical angular position resolution given by the available optical encoder is determined considering the following facts:

- The optical encoder provides 360 pulses for a complete mechanical revolution of the rotor shaft.
- The BLDC motor has 10 pole pairs.

The electrical angular position resolution, in degrees, is calculated as

$$\theta_e^{res} = \frac{360}{\frac{PPR}{Z_p}} = \frac{360}{\frac{360}{10}}$$

$$\theta_e^{res} = 10$$

The above result indicates that the available optical encoder provides an electrical angular position feedback of 36 pulses per electrical revolution (10 degrees per pulse). With this resolution, the FOC implementation will not be feasible. Nevertheless, the FOC algorithm is implemented in the dsPIC30F2010 to confirm the hypothesis.

The algorithm for the FOC implementation is described as follows

- Perform a rotor alignment with the d-axis (phase A +, phase B -, phase C-), since the encoder does not provide absolute position information.
- Take a sample of the currents and calculate i_α, i_β applying the Clark's transformation
- Calculate the sine and cosine of the angular position, using a look-up table as follows
    - Determine the corresponding look-up table index for sine and cosine as the modulo operation of the actual pulse count by the total number of pulses per electrical revolution. In C language, this will be

      index = pulses % 36;
    - Retrieve the sine and cosine values from the look-up tables

      sin_theta = sin_table[index];

      cos_theta = cos_table[index];
- Calculate i_d, i_q applying the Park's transformation
- Obtain v_d, v_q from the PI current controllers
- Calculate v_α, v_β with the inverse Park's transformation
- Execute the space vector PMW modulation algorithm to update the duty cycle registers of the PWM module

The above algorithm was applied with at 8kHz PWM frequency on the dsPIC30F2010 running at 20MIPS. Results shown that with a poor encoder resolution, the motor shaft rotates with very high torque ripple and vibrations.

4.9.4. Torque Control with Trapezoidal Commutation

To overcome the issue produced in FOC due to a poor optical encoder resolution, the trapezoidal commutation method is applied, with the torque regulation performed for every commutation state. A simplified mathematical model for the BLDC motor is obtained based on reference [7].

Neglecting the mutual inductances, the equations for the BLDC motor will be

$$v_a = i_a R_s + L_s \frac{di_a}{dt} + e_a$$

$$v_b = i_b R_s + L_s \frac{di_b}{dt} + e_b$$

$$v_c = i_c R_s + L_s \frac{di_c}{dt} + e_c$$

Where R_s is the stator resistance per phase, and is assumed to be equal for all three phases. L_s is the self-inductance of each phase, also assumed equal for all three phases. e_a, e_b, e_c are the induced back-EMF, and are all assumed to be trapezoidal.

The torque, in Newton-meter is given by

$$T_e = \frac{[e_a i_a + e_b i_b + e_c i_c]}{\omega_m}$$

The instantaneous back-EMF values can be written as a function of the rotor position as follows

$$e_a = f_a(\theta_m)\lambda_p \omega_m$$

$$e_b = f_b(\theta_m)\lambda_p \omega_m$$

$$e_c = f_c(\theta_m)\lambda_p \omega_m$$

Where λ_p are the flux linkage, and $f_a(\theta_m), f_b(\theta_m), f_c(\theta_m)$ have the same shape as e_a, e_b, e_c with a maximum magnitude of ± 1.

With the above assumptions, the electromagnetic torque can be rewritten as

$$T_e = \lambda_p [f_a(\theta_m) i_a + f_b(\theta_m) i_b + f_c(\theta_m) i_c]$$

The current magnitude command I_p^* is obtained from the torque expression as

$$T_e^* = \lambda_p [f_a(\theta_m) i_a^* + f_b(\theta_m) i_b^* + f_c(\theta_m) i_c^*]$$

For the trapezoidal commutation, only two machine phases connected in series, conduct current at any time, therefore, the phase currents are equal in magnitude but opposite in sign. $f_a(\theta_m), f_b(\theta_m), f_c(\theta_m)$ have the same sign as the stator phase current in the motoring mode, and have opposite signs in regeneration. This sign relationship leads to a simplification of the torque command as

$$T_e^* = 2\lambda_p I_p^*$$

Finally, the stator current command will be

$$I_p^* = \frac{T_e^*}{2\lambda_p}$$

The individual stator phase currents commands are obtained according to the rotor position and the magnitude of I_p^*. The current-sense shunt resistors mounted in the inverter provide the feedback information required to regulate the phase currents.

4.9.5. Speed Controllers Comparison

The PI speed controller for the BLDC motor is implemented as described in point 4.5.2. The controller is tuned according to the measured BLDC motor parameters. The fuzzy-tuned PI speed controller is implemented as outlined in 3.2.5.1 with the fuzzy-logic implementation described in point 4.6

The speed feedback signal is calculated inside the microcontroller, using a timer and the pulse count provided by the encoder. The RPM measurement algorithm is based on the frequency measurement method described in [37]. The speed measurement algorithm is executed every 10ms, hence, the control sampling frequency for the speed controllers will be 100Hz.

Several speed commands are sent from the computer, using a MATLAB function, to the microcontroller via serial communication. In every control iteration, the actual RPM value is sent back to the computer in order to plot the results.

The experimental results are present in figures 4.33 and 4.34 for the PI speed controller and for the fuzzy-tuned PI speed controller respectively.

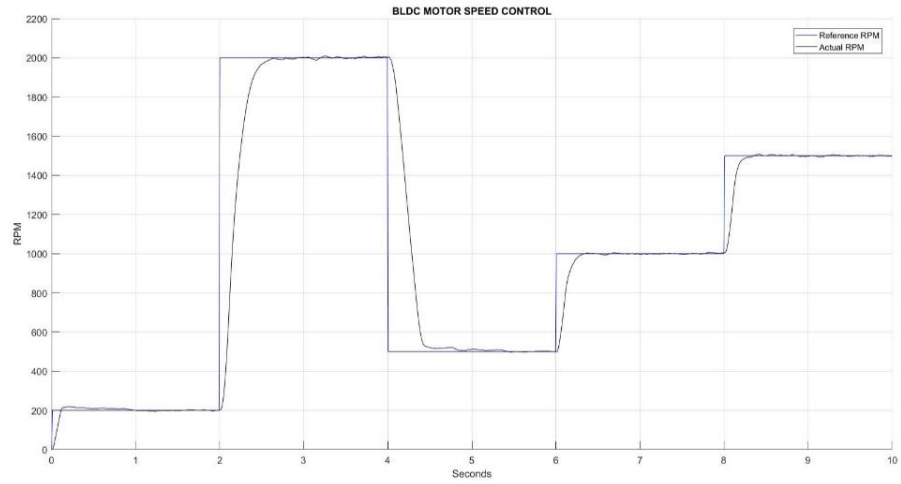

Figure 4.33 BLDC motor PI speed controller response

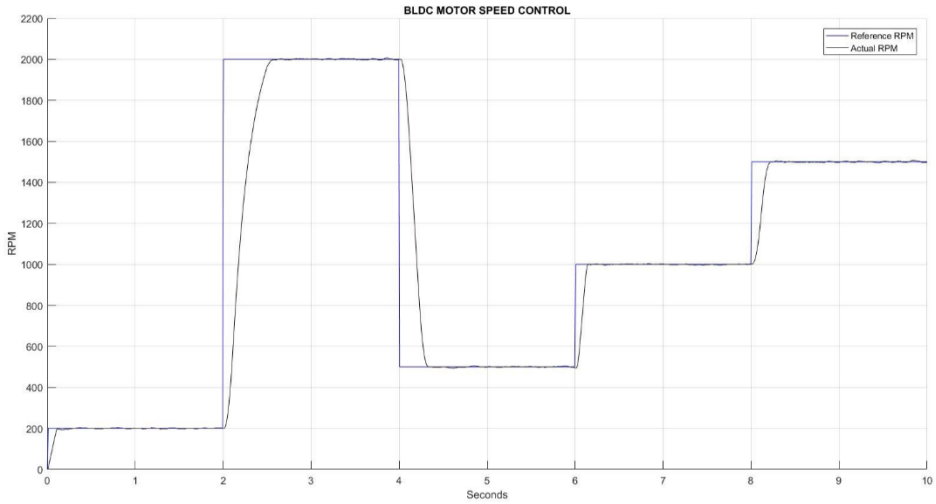

Figure 4.34 BLDC motor fuzzy-tuned PI speed controller response

The conventional PI speed controller is compared against the fuzzy-tuned PI speed controller by means of error indicators namely integral square error, integral absolute error and root mean squared error. The data of the speed controllers are exported from MATLAB as comma-separated-value (.csv) files, and loaded into a spreadsheet to calculate the error indicators. Figures 4.35 to 4.37 present the comparison results with bar charts for each error indicator.

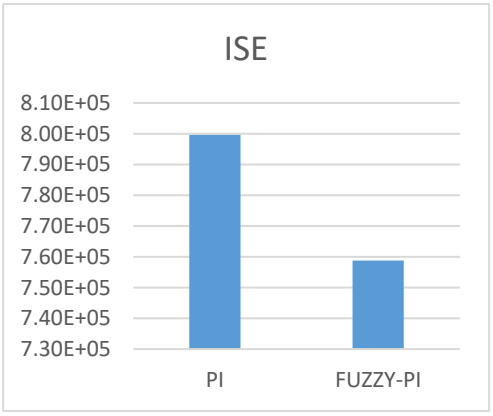

Figure 4.35 Comparison of speed controllers. ISE error indicator

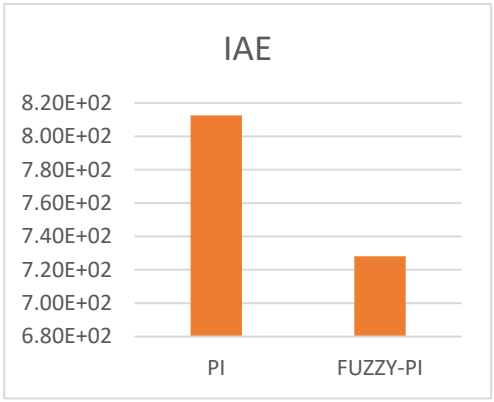

Figure 4.36 Comparison of speed controllers. IAE error indicator

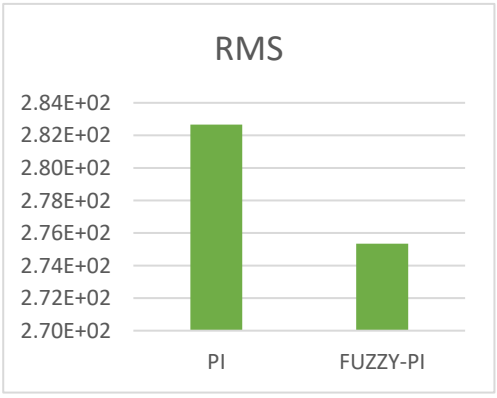

Figure 4.37 Comparison of speed controllers. RMS error indicator

The above results suggest a better performance for the fuzzy-tuned PI speed controller over the conventional PI controller. These results are consistent with the results obtained by computer simulations, and thus, the validity of the proposed speed controllers is verified.

This page was intentionally left blank

5. CONCLUSIONS

Fuzzy-logic systems have many advantages over the conventional controllers. For instance, fuzzy-logic controllers can handle non-linearities, do not require precise mathematical models, and work based on the intuition and experience of the human operator. Therefore, fuzzy inference systems can be applied to improve the performance of electric drive control systems based on conventional PID controllers, either modifying and dynamically tuning the PID gains, or directly replacing the PID regulator with a fuzzy-logic controller.

Computer simulations of control systems can reduce development time when the transition between the simulation stage to the actual implementation is straightforward. Proteus VSM software was used to simulate a PMSM control system, directly implementing the control algorithms in a microcontroller, therefore, helping to reduce the time required for a real-world implementation. This was demonstrated with a physical implementation of the speed controllers for a BLDC motor, where only the torque control scheme had to be changed.

Some practical issues must be taken into account when implementing electric drive control systems in the real-world, specifically the ones related with digital control systems and sensor interfacing. For field oriented control, the angular position feedback has to be as precise as possible, whether it is obtained from a sensor, or through a mathematical estimation.

A fuzzy-tuned PI speed controller was implemented in this work, presenting an improved performance in torque-ripple reduction and when coping with parameter variations, compared with a conventional PI controller. Similar results were obtained for the implemented fuzzy-logic position controller. Nevertheless, simulations with an applied periodical torque disturbance of considerable amplitude (about 70% of the machine rated torque) showed a slightly better performance of the conventional controllers over the fuzzy-logic-based controllers. The above results were obtained in terms of computer simulations within Proteus VSM software.

The physical implementation of the speed controllers gave consistent results with the previously obtained simulation results, and thus, the validity for the proposed speed controllers is proved. However, the validation of the position controllers was not performed due to the limitation in the optical encoder, and is left for future work together with the validation of all controllers in a PMSM.

The inclusion of artificial neural networks in a servo drive controller for PMSM would be an interesting topic for future research, specifically a research about possible advantages of ANN-based position controllers to adapt and operate correctly even after many hours/days of work.

References

[1] Haitham Abu-Rub, Atif Iqbal, and Jaroslaw Guzinski, *High Performance Control of AC Drives with Matlab/Simulink Models*, First. John Wiley & Sons, 2012.

[2] Minoru Kondo, “Application of Permanent Magnet Synchronous Motor to Driving Railway Vehicles.” Railway Technology Avalanche, 01-Jan-2003.

[3] Alexandru Bara, Calin Rusu, and Sanda Dale, “DSP Application of PMSM Drive Control For Robot Axis.” WSEAS Transactions on Systems and Control, 2009.

[4] Adel El Shahat and Hamed El Shewy, “Permanent Magnet Synchronous Motor Drive System for Mechatronics Applications.” IJRRAS, Aug-2010.

[5] John Chandler, “PMSM Technology in High Performance Variable Speed Applications.” Automotion Inc., 2006.

[6] Ion Boldea and S. A. Nasar, *Electric Drives*. CRC Press, 1999.

[7] R. Krishnan, *Permanent Magnet Synchronous and Brushless DC Motor Drives*. CRC Press, 2010.

[8] Seung-Ki Sul, *Control of Electric Machine Drive Systems*. John Wiley & Sons, 2011.

[9] Jacek F. Gieras, *Permanent Magnet Motor Technology Design and Applications*, Third. CRC Press, 2010.

[10] Bao-Zhu Guo and Zhi-Liang Zhao, *Active Disturbance Rejection Control for Nonlinear Systems*. John Wiley & Sons, 2016.

[11] Xing-Hua Yang, Xi-Jun Yang, and Jian-Guo Jiang, “A Novel Position Control of PMSM Based on Active Disturbance Rejection.” WSEAS Transactions on Systems and Control, 2010.

[12] Miroslav Krstic, Ioannis Kanellakopoulos, and Petar Kokotovic, *Nonlinear and Adaptive Control Design*. John Wiley & Sons, 1995.

[13] Kendouci KHADIJA, Mazari BENYOUNES, Bouserhane Ismain KHALIL, and Benhadria Mohamed RACHID, “A simple and robust Speed Tracking Control of PMSM.” PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review), 2011.

[14] Maurice Clerc, *Particle Swarm Optimization*. ISTE ltd, 2005.

[15] Ming Yang, Xingcheng Wang, and Kai Zheng, “Nonlinear Controller design for Permanent Magnet Synchronous Motor Using Adaptive Weighted PSO.” American Control Conference, 2010.

[16] Francisco Rodríguez Rubio and Manuel Jesús López Sánchez, *Control Adaptativo y Robusto*. Universidad de Sevilla, 1996.

[17] Liu Mingji, Cai Zhongqin, Cheng Ximing, and and Ouyang Minggao, “Adaptive Position Servo Control of Permanent Magnet Synchronous Motor.” 2004 American Control Conference, Boston, Massachusetts, Jul-2004.

[18] “Nonlinear dynamic inversion.” www.aerostudents.com.

[19] Zhang Yaou, Zhao Wansheng, and Kang Xiaoming, “Control of the Permanent Magnet Synchronous Motor Using Model Reference Dynamic Inversion.” WSEAS Transactions on Systems and Control, May-2010.

[20] Zdenko Kovacic and Stjepan Bogdan, *Fuzzy Controller Design Theory and Applications*. CRC Press, 2006.

[21] Timothy J. Ross, *Fuzzy Logic With Engineering Applications*, Third. John Wiley & Sons, 2010.

[22] Mohamed Kadjoudj, Noureddine Golea, and Mohamed El Hachemi Benbouzid, "Fuzzy Rule-Based Model Reference Adaptive Control for PMSM Drives." Serbian Journal of Electrical Engineering, Jun-2007.

[23] Ying-Shienh Kung and Pin-Ging Huang, "High Performance Position Controller for PMSM Drives Based on TMS320F2812 DSP." IEEE International Conference on Control Applications, Taipei, Taiwan, Sep-2004.

[24] Jagannathan Sarangapani, *Neural Network Control of Nonlinear Discrete-Time Systems*. CRC Press, 2006.

[25] Mahmoud M. Saafan, Amira Y. Haikal, Sabry F. Saraya, and Fayez F.G. Areed, "Artificial Neural Network Control of Permanent Magnet Synchronous Motor." International Journal of Computer Applications (0975-8887), Jan-2012.

[26] Wilfrid Perruquetti and Jean Pierre Barbot, *Sliding Mode Control in Engineering*. Marcel Dekker, Inc., 2002.

[27] Fadil Hicham, Driss Yousfi, Aite Driss Youness, Elhafyani Mohamed Larbi, and Nasrudin Abd Rahim, "Sliding-Mode Speed Control of PMSM with Fuzzy-Logic Chattering Minimization, Design and Implementation." MDPI Open Access Journals, 2015.

[28] Fouad Giri, *AC Electric Motors Control Advanced Design Techniques and Applications*, First. John Wiley & Sons, 2013.

[29] Fayez F.M. El-Sousy, "Adaptive hybrid control system using a recurrent RBFN-based self-evolving fuzzy-neural-network for PMSM servo drives." ELSEVIER, Applied Soft Computing, 2014.

[30] Liuping Wang, Shan Chai, Dae Yoo, Lu Gan, and Ki Ng, *PID and Predictive Control of Electrical Drives and Power Converters using MATLAB/Simulink*, First. John Wiley & Sons, 2015.

[31] Antonio Visioli, *Practical PID Control*. Springer, 2006.

[32] M. Sami Fadali and Antonio Visioli, *Digital Control Engineering Analysis and Design*, Second. Elsevier, 2013.

[33] Pavel Rech, "FlexTimer and ADC Synchronization for Field Oriented Control on Kinetis." NXP Freescale Semiconductor, 2011.

[34] Nicolau Pereira Filho, João Onofre P. Pinto, Luiz E. Borges da Silva, and Bimal K. Bose, "A Simple and Ultra-Fast DSP-Based Space Vector PWM Algorithm and its Implementation on a Two-Level Inverter Covering Undermodulation and Overmodulation." The 30th Annual Conference of the IEEE Industrial Electronics Society, Busan, Korea, Nov-2004.

[35] Zhenyu Yu, "Space-Vector PWM With TMS320C24x/F24x Using Hardware and Software Determined Switching Patterns." Texas Instruments, 1999.

[36] Viktor Bobek, "PMSM Electrical Parameters Measurement." NXP Freescale Semiconductor, 2013.

[37] OPTO 22 Technical Note, "RPM Measurement Techniques." OPTO 22, 2013.