ABSTRACT

CHITNIS, SANDESH MOHANIRAJ. Six Step Control of a Permanent Magnet Synchronous Motor with Improved Current Dynamics. (Under the direction of Dr. Subhashish Bhattacharya).

Battery utilization for permanent magnet motor control can be improved considerably using six-step control algorithm. Traditionally current control in six-step algorithm is not possible due to improper operation of PI controller needed for faster current control. This is because maximum voltage already is already available in six-step mode. Using proper control strategy, dynamic current control can be achieved even in six-step operating mode. This is achieved by modifying the voltage references by feedback calculation of current.

Six Step Control of Permanent Magnet Synchronous Motor with Improved Current Dynamics

by
Sandesh Chitnis

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2015

APPROVED BY:

_____	_____
Dr. Subhashish Bhattacharya Committee Chair	Dr. Mesut Baran
_____	
Dr. David Lubkeman	

BIOGRAPHY

Sandesh Chitnis was born in Mumbai, India on 9th April, 1991. He completed his Bachelors in Technology degree in Electrical Engineering from Veermata Jijabai Technological Institute in Mumbai, India. After his degree, he joined the Masters of Science in Electrical Engineering program at North Carolina State University at Raleigh, NC. He has worked as intern at Joby Motors at Santa Cruz, CA and Altaeros Energies at Boston, MA as an engineering intern.

ACKNOWLEDGMENTS

I wish to express my gratitude to my adviser, family, and to friends for their patience and unending support making this research possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Motivation

Permanent Magnet Synchronous Motors are getting more and more attention in the recent years since the advance in magnet technology. As new magnetic materials are devised, stronger magnets and hence highly efficient PM motors with high power density can be constructed. They are used widely in many applications such as electric vehicles, washing machines, HVAC, et al.

The operating speed of the PMSM is directly proportional to its back-emf and hence the voltage supplied. Thus when higher operating speeds are needed, the inverter supply voltages have to be increases. When the voltage limit is reached at the base speed, the speed can no longer go up without decrease in the available torque. Thus the higher the voltage, the higher the operating speed with constant torque.

The maximum available output voltage of the inverter essentially depends on the DC link voltage. This DC link voltage is the output of the input rectifier in most cases, while it is the battery voltage available in automotive applications.

In automotive applications like electric cars or drones, the DC voltage is provided by the battery which is limited in nature. This battery voltage dictates the highest possible speed that can be attained. Also maximum current that can be extracted from the battery, which will in turn ensure maximum torque of the motor. Thus it is important that the battery utilization needs to be maximum.

Conventional PWM methods used to control the inverter such as Sine-PWM (SPWM) and Space-vector PWM (SV-PWM) cannot make use of the complete limited DC voltage. Six-step

method makes full use of the battery voltage available. Thus for the same system, higher speeds can be attained without change in the hardware setup.

In the six-step mode, current control is not possible since voltage output has the maximum possible value. By adopting a proper control strategy, current control over a short range can be achieved in the six-step mode. This control strategy can extend the operating region of the PMSM used in applications like electric vehicles. Thus higher speeds can be attained with proper current control without modifications in hardware.

Joby Motors has been working on an unmanned aerial vehicle (UAV) application. The motor controller board which has been developed by the engineers is used to implement the six-step control method with no modification of hardware. Thus highest speed can be attained with sufficient torque control for this application.

CHAPTER 2: Introduction

2.1 Past Trends

DC motors have been in use in the industry for a long time. Their main advantages included low cost, lack of complexity in drive design, and precise control. This is mainly due to the fact that in DC drives there is a decoupling in the speed and torque control i.e. we can control the speed and torque of the motor independently. This is especially true for a separately excited DC machine. DC motors however have some inherent disadvantages, e.g. Use of brushes and commutators brings down the efficiency and shelf life of the motor. Armature reaction can also result into incorrect commutations degrading the efficiency of the DC motor further.

In the last two decades, there has been great advancement in power electronics technology. Development of high-voltage, high-current MOSFETs and IGBTs which can be switched at very high speeds of up to 150 kHz have brought about improvement of waveform quality. Since these devices typically also show a low on-resistance and voltage drop, the converter can be made very efficient. There has also been tremendous development in control techniques for converters. Numerous PWM techniques have been developed and researched thoroughly. A simple two-level three-phase inverter with six switches can be controlled in a variety of schemes to achieve required performance. Switching techniques are also employed to optimize specific criteria, e.g. choosing discontinuous PWM technique can reduce switching losses optimizing efficiency. Different techniques can also be chosen to reduce specific harmonics e.g. odd harmonics, even harmonics. This type of progress has enabled use of power electronic converters in many applications such as earphones, computer power supplies, wall chargers, AC drives, grid integration of distributed generation sources.

Many control techniques have been also developed for the control of AC motors. These control schemes are implemented with the use of power electronic converters and sophisticated control circuitry associated with AC drives. Using these control schemes, very precise control of AC drives could be obtained. Thus eventually AC drives have replaced DC drives in most applications. Some other advantages also include low maintenance requirements, lack of requirement of brushes, high efficiency, and high availability. Some disadvantages include high initial cost, control is comparatively complicated.

2.2 AC motor basics

AC motors are predominantly divided into:

1. Induction motors: An induction motor is one of the most commonly used motors in industry applications due to its rugged nature and reliability. It works on the principle of induction i.e. the stator current induces a flux in its coils. The rotor wires arrangement is such that this flux induces a voltage in the rotor windings and thus current flows causing electromagnetic action. Thus due to interaction of stator and rotor flux cause the motion of induction motor [1]. Induction motor is also called as asynchronous motor since stator flux and rotor rotate at different speeds. The stator flux rotates at the synchronous speed which is equal to

$$N_s = \frac{120 * f}{p} \quad (1)$$

Where N_s is the electrical synchronous speed (RPM)

f is the frequency of the AC voltage supplied to stator (Hz)

p is the number of poles of the motor.

Induction motor electrical structure is given in Figure 1.

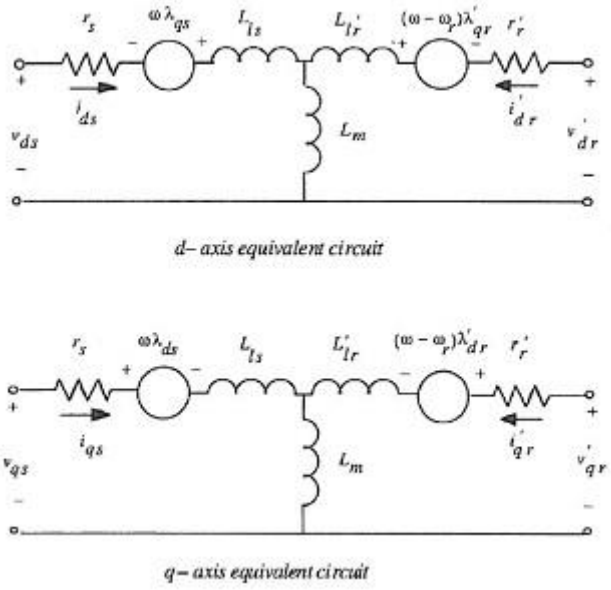

Figure 1. Induction machine dq-axis model [1]

The motor mechanical operating speed is always lower than the synchronous speed due to operating principle of the induction motor. This difference in speed is called the slip.

Induction motors have been traditionally used in single speed applications, but more recently their application in Variable Frequency Drives (VFD) has also seen an increase.

2. Synchronous motors:

Synchronous motors are motors which have a separate DC field excitation and an AC armature. Their operating speed is the synchronous speed in synchronism with operating frequency. Since the magnetic flux is supplied by the field, the induction principle isn't used in

synchronous motors. The magnetic flux is also supplied sometimes by permanent magnets. Thus a separate DC source is not needed for excitation.

Recent advances in materials technology has enabled the creation of high-intensity permanent magnets, such as neodymium magnets and also some rare earth magnets, allowing the development of compact, high-power motors which are lighter and more efficient than their wound-rotor counterparts [2]. Permanent Magnet Synchronous Motors (PMSM) are widely used in various applications such as washing machines, dishwashers, automotive applications. They can be further divided into:

1. Surface Magnet PMSM: In surface magnet PMSM, the magnets are mounted on the surface of the rotors. These motors are easier to construct, but are more susceptible to mechanical stresses and armature reaction [3].

2. Interior Magnet PMSM: In interior magnet PMSM, the magnets are buried deep beneath the rotor surface. They protect the magnet from mechanical stresses, but there is a relatively high leakage flux loss.

Surface magnet PMSM and Interior Magnet PMSM structure is shown in Figure 2.

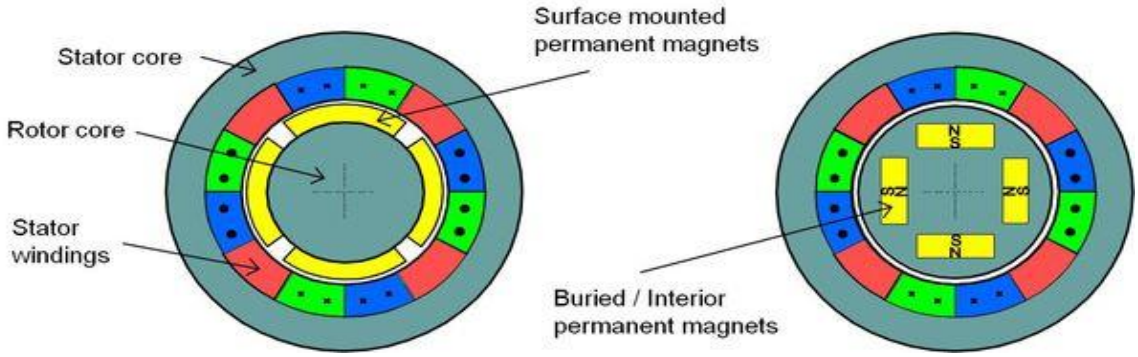

Figure 2. Surface Magnet PMSM and Interior Magnet PMSM mechanical structure [6]

2.3 AC Motor Control Schemes

The various control techniques that were developed for control of AC drives can be divided into two main divisions:

2.3.1 Scalar control

Scalar control is the control where two parameters to the motor are varied simultaneously [4]. Usually the voltage and frequency of the supply to the motor are varied for speed and torque control of the motor. This method is called the Volts/Hz control of motor. The speed depends on the frequency while the torque depends on the current and hence the voltage. It is necessary to increase or decrease voltage and frequency proportionally so as to avoid saturation in the coils.

Using this method it is possible to increase the torque and speed up to the base speed of the motor. At the base speed, the maximum voltage is applied to the motor. Voltage cannot be increased further as it may lead to damage to insulation of the coils. Hence in operation in the region above the base speed, only frequency is increased in order to attain higher speeds. In

this region, the torque of the motor decreases. The torque-speed characteristics of an induction machine are given in Figure 3.

Scalar control can be implemented in either open-loop mode or closed-loop mode. The Volts/Hz method described earlier is a type of open-loop scalar control technique. In such techniques, there is no need for feedback measurement of any signals such as speed or current. These techniques are simple to implement and are typically low-cost. However a big shortcoming is that the control over speed is not precise enough and the response is susceptible to change in load.

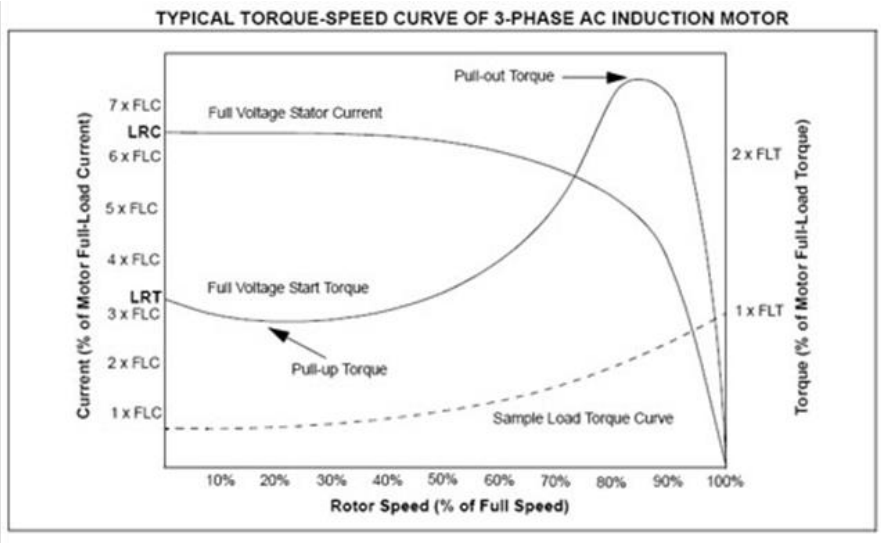

Figure 3. Torque-Speed characteristics of Induction Motor [5]

Closed-loop scalar control employs a speed feedback loop. This measured speed is compared to the reference speed that is set, and the error can be passed to a PI controller. Hence precise control of speed can be obtained. [4]

2.3.2. Vector control

Vector control involves conversion of three-phase quantities to the dq- reference frame. The dq-reference frame is a rotating reference frame which is usually rotating at synchronous speed. Thus all quantities such as three phase voltages and currents appear to be DC values in this reference frame. This conversion also allows for decoupling of the speed and torque control components.

Field-oriented control (FOC) of motor is the most widely used vector control strategy for motors. In this strategy, the stator flux is always kept orthogonal to the field flux due to permanent magnets in PMSMs, induced rotor flux in induction machines, and DC field flux in wound rotor synchronous machines. Maximum torque is obtained when both the fluxes are orthogonal. For proper electronic commutation of the PWM inverter to achieve orthogonality, it is necessary to estimate or measure the rotor flux position at any given instant. This is done by employing Hall flux sensors to sense flux position in Direct FOC techniques as seen in Figure 4 and by estimating rotor flux position from the rotor mechanical position in Indirect FOC techniques as seen in Figure 5.

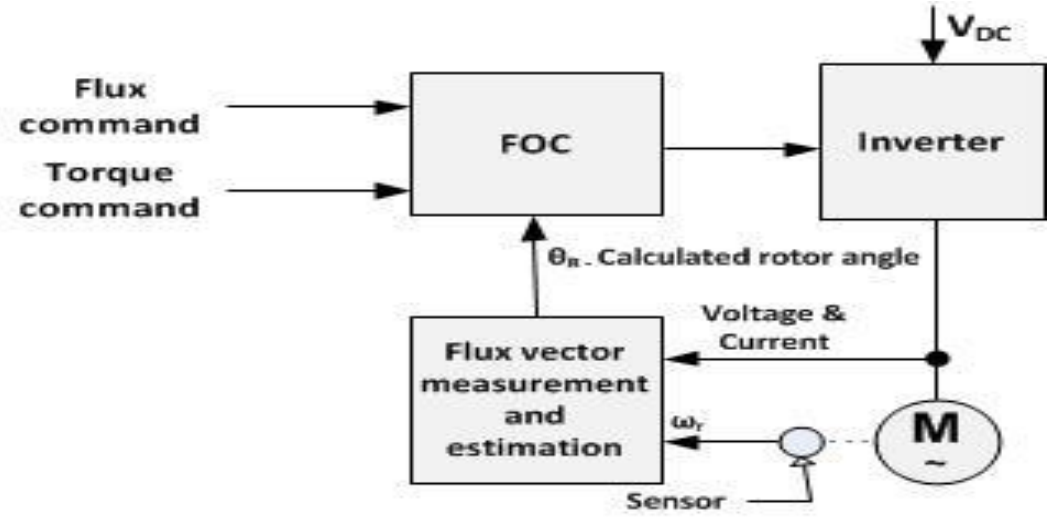

Figure 4. Direct Field Oriented Control [4]

Field oriented control is one of the most widely used strategies as it gives independent control over the speed and the torque and the speed and torque responses are precise.

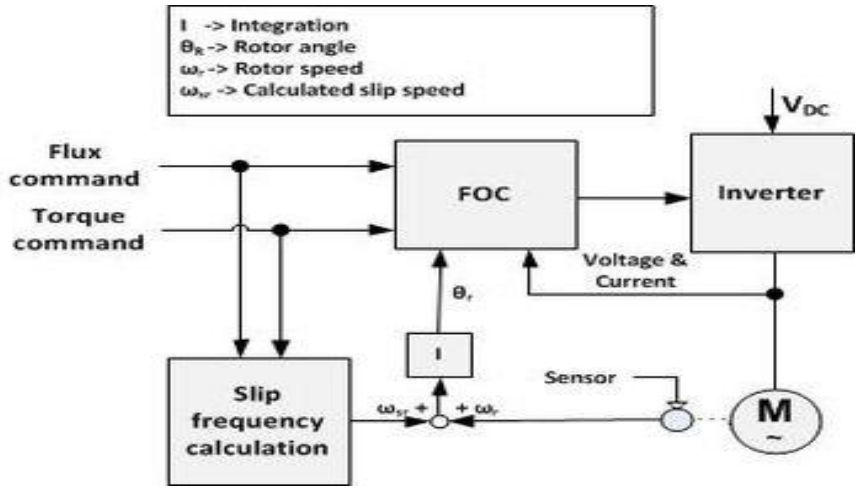

Figure 5. Indirect Field Oriented Control [4]

The block diagram of a conventional motor control scheme using a Sine-PWM modulator is shown in Figure 6.

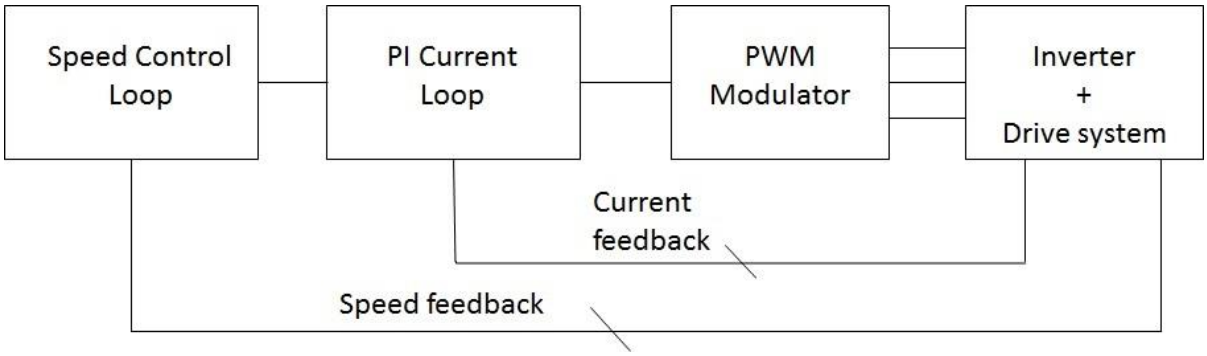

Figure 6. Basic motor control scheme

CHAPTER 3: Modulation Methods and Six-Step Control

3.1 Conventional Modulation methods

There exist a multitude of pulse width modulation methods in literature. Some popular methods are described below.

3.1.1 Sine-PWM

Sine-PWM is one of the most common techniques used to provide appropriate switching signals to switches such that the required output voltage. In this method, a sinusoidal reference modulation signal is compared to a triangle carrier [7]. The intersection points of the modulation and carrier signals decide the switching signals of the individual switches. The on-off signal is decided as follows,

$$V_{ref} > V_{car}, \mathrm{S1} = 1. \quad (2)$$

$$V_{ref} < V_{car}, \mathrm{S1} = 0 \quad (3)$$

Thus the width of the PWM signals varies according to the amplitude of the modulating signal as shown in Figure 6. As it can be seen, the amplitude of the carrier signal is ($V_{dc}/2$). Thus whenever the reference signal exceeds the carrier waveform amplitude, pulse-dropping occurs and linearity of the output voltage waveform is lost.

The voltage linearity range of this method is poor and it can only reach 78.5% of the six-step voltage fundamental value. Thus utilization of the DC link voltage is very low.

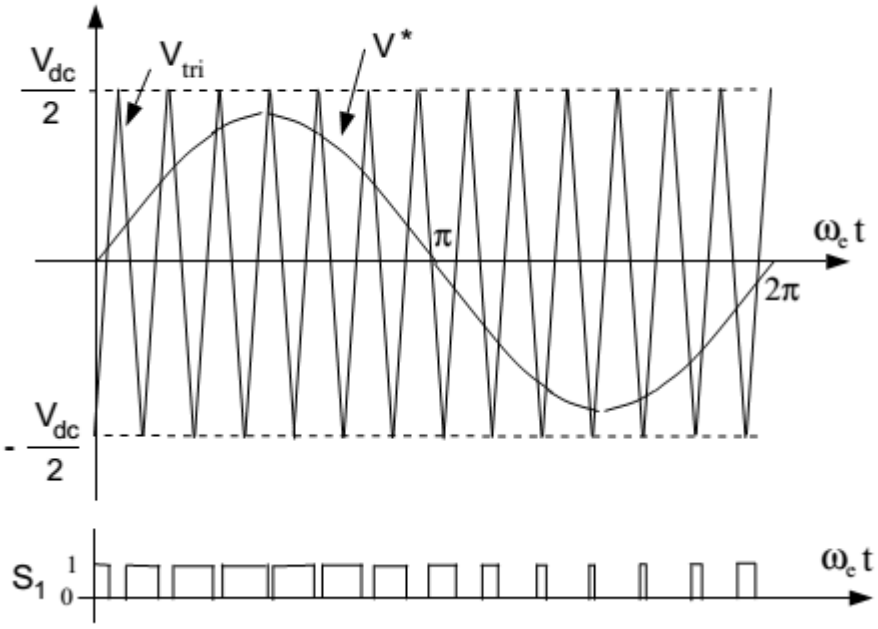

Figure 7. Sine PWM visualization [8]

The modulation index (M.I.) is defined as the ratio of voltage output of the inverter to the maximum possible output voltage, which is attained by six-step method.

$$M.I. = \frac{V_{1m}}{V_{1SixStep}} \tag{4}$$

M.I. for SPWM is 0.785 i.e. it can generate up to 78.5% of the voltage generated by six-step method.

For applications with isolated neutrals like motors, voltage linearity of the SPWM method can be extended by injecting a zero-sequence signal to the reference waveform. There are various choices for the zero-sequence signals and different choices lead to results of a varying nature. For different zero-sequence signals, different criteria of the output can be chosen for

optimization. For example, a third-harmonic signal equal to one-fourth of the fundamental can lead to minimum RMS harmonic distortion [9].

3.1.2 Space Vector Modulation

Space Vector modulation technique is a direct digital implementation version of one of the zero-sequence injection techniques. In this method, the three reference signals are compared and the one with the minimum magnitude is found. After scaling the minimum magnitude reference vector by 0.5, it is added to all voltage references. This method has superior performance and the voltage linearity is extended to 90.5% of the six-step voltage output. [10]

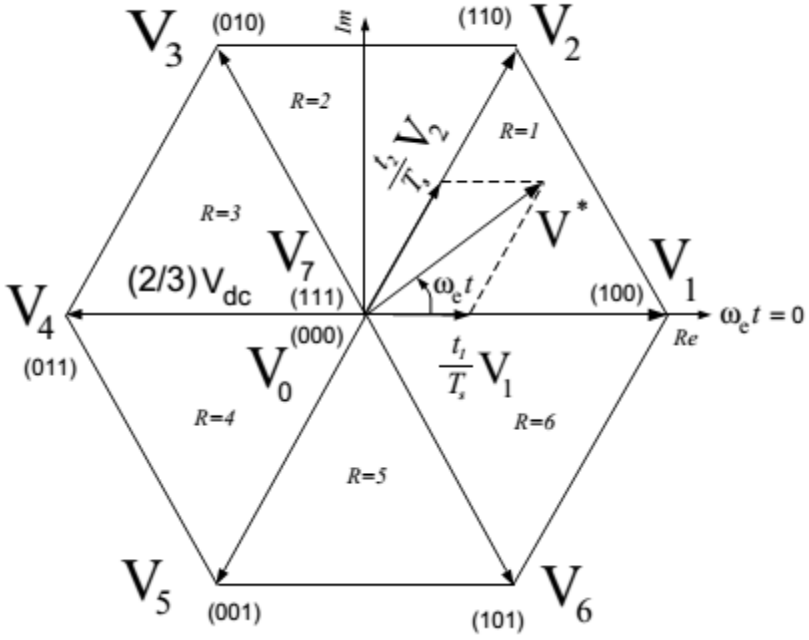

Figure 8. Space Vector PWM [8]

This method is also quite intuitive and easy to understand. The voltage vectors available are the six vertices of the hexagon and the two zero states. As can be seen in Figure 7, the voltage output equal to voltage reference is generated by calculating the time to be spent on each voltage vector. This calculation is done online, and the need for generating a triangular wave carrier wave is eliminated.

3.1.3 Generalized Discontinuous PWM [8]:

This method of PWM has been developed in [8]. In discontinuous PWM methods, one switch ceases switching for some period of the switching cycle. Due to this, switching losses of the overall inverter system are reduced considerably. In this method, the phases with maximum positive and negative reference values are held to the positive and negative DC rails values respectively for one 60° period.

This method gives good response in the high modulation region while its response in the lower modulation region is not satisfactory. It gives a better performance in six-step mode than any other modulation method [8]. Since it will be used in the simulation later, it will be covered in greater detail here.

In this method, the voltage reference with the maximum positive or negative magnitude is chosen and the zero sequence signal is calculated from that reference as,

$$V_0 = \pm V_{DC} * 0.5 - V_{max} \quad (5)$$

The new modulation signals are then calculated as,

$$V_{Anew} = V_{Aold} + V_0 \quad (6)$$

$$V_{Bnew} = V_{Bold} + V_0 \quad (7)$$

$$V_{Cnew} = V_{Cold} + V_0 \quad (8)$$

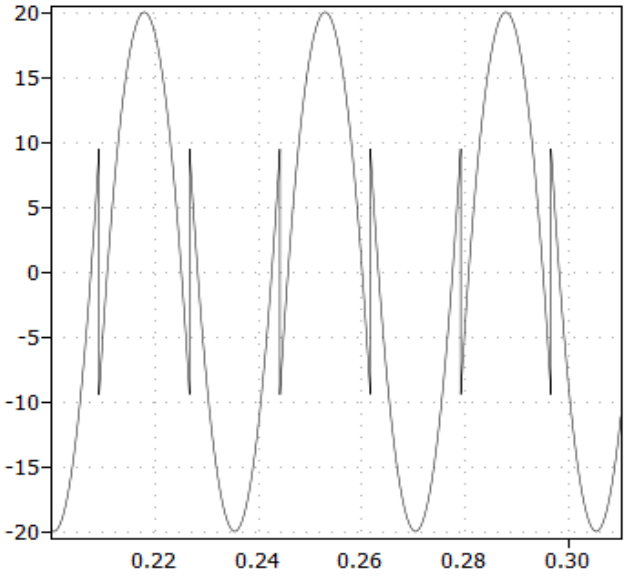

Figure 9. GD-PWM zero sequence signal

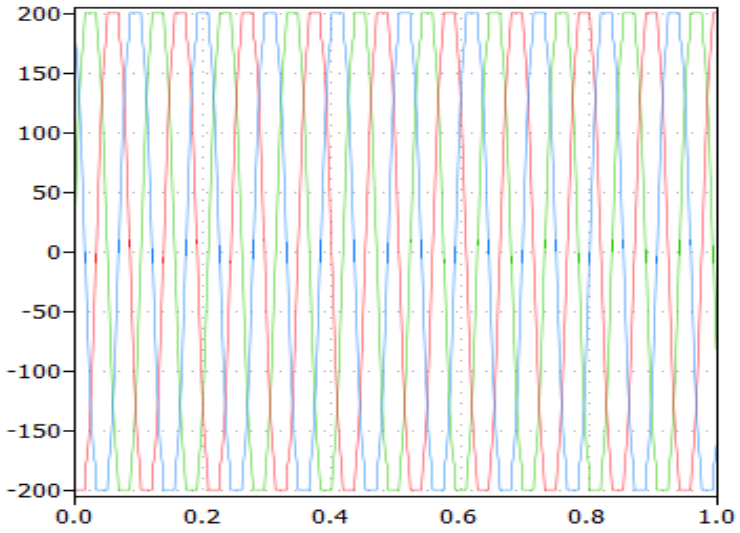

Figure 10. GD-PWM modulation waveforms

3.2 Over-modulation

Over-modulation region is the region where output voltage becomes a non-linear function of the modulating reference voltage. In conventional Sine-PWM method, over-modulation starts when the reference modulating waveform amplitude is greater than the triangle carrier amplitude. In such a case, the part of the reference waveform above the carrier magnitude has no effect on the output voltage magnitude. This leads to pulse-dropping causing non-linearity. Thus for sine-triangle PWM, over-modulation starts when

$$|V_{ref}| > V_{dc}/2 \tag{9}$$

In space vector theory, over-modulation can be said to occur when the voltage reference rotates outside the voltage hexagon. This can be seen in Figure 11.

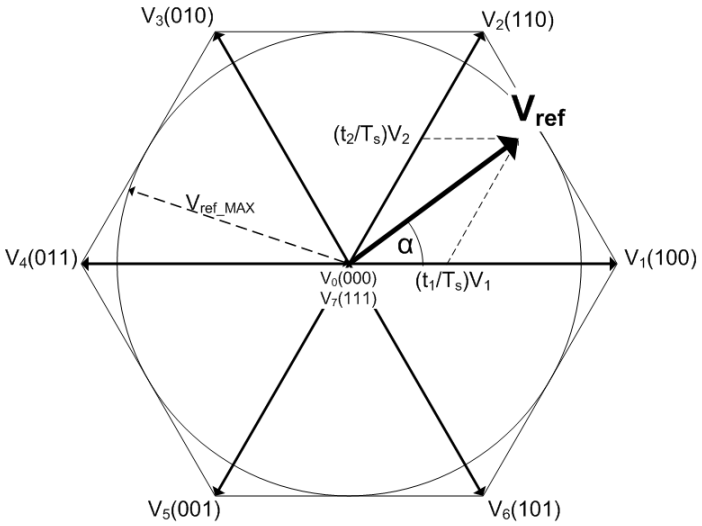

Figure 11. Voltage Modulation limit [8]

The voltage linearity range of this method is poor and it can only reach 78.5% of the six-step voltage fundamental value. Thus utilization of the DC link voltage is very low.

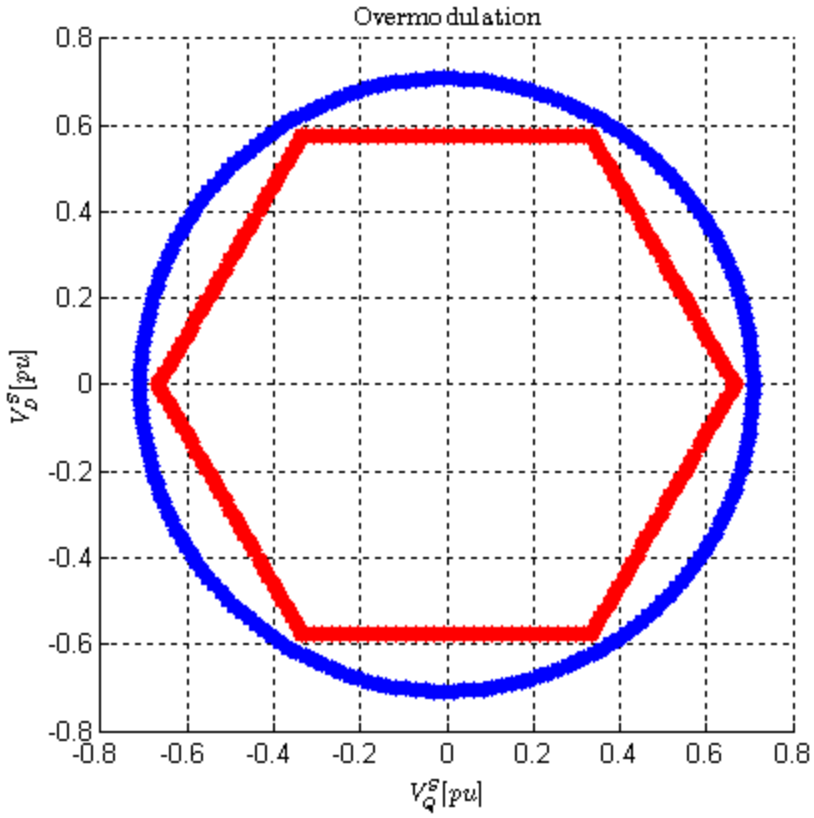

Figure 12. Over-modulation [12]

The modulation index (M.I.) is defined as the ratio of voltage output of the inverter to the maximum possible output voltage, which is attained by six-step method.

$$M.I. = \frac{V_{1m}}{V_{1SixStep}} \tag{10}$$

M.I. for SPWM is 0.785.

For applications with isolated neutrals like motors, voltage linearity of the SPWM method can be extended by injecting a zero-sequence signal to the reference waveform. There are various choices for the zero-sequence signals and different choices lead to results of a varying nature.

Perhaps the most intuitive way to perform over-modulation is using methods involving space vector implementation. The maximum output voltage that can be synthesized is limited by the voltage hexagon geometry. As can be seen in Figure 10 the modulation region is limited by the circle inscribed inside the voltage hexagon. If the voltage reference goes outside this circle, the voltage linearity is lost. Over-modulation methods are required to synthesize voltage utilizing the corners of the hexagon that are not covered by the circle. The method introduced in [13] makes use of a pre-processor which processes the voltage reference such that the modified voltage reference can be used with the conventional PWM methods and maintain linearity. The method divides the over-modulation region into two regions, over-modulation region I that extends till the M.I. is equal to 0.952 and over-modulation region II which extends all the way till six-step operation i.e. M.I equal to 1. This method gives good results but requires a lot of processing power and memory as use of a look-up table is involved.

Other methods for over-modulation are given in [8]. Minimum Magnitude Error Over-modulation technique involves choosing a point on the voltage vector which is vectorially closest to the voltage reference.

Minimum Phase Error PWM technique is choosing a point on the voltage hexagon which has the same angle as the voltage reference.

Dynamic Field Weakening technique for over-modulation has been developed in [14]

Six step control marks the end of the over-modulation region. The maximum possible output voltage is obtained using six-step control. There isn't much literature available about six-step control of inverters since it isn't used widely and it is supposedly the simplest technique out there. However a qualitative study of this method will prove useful for this thesis.

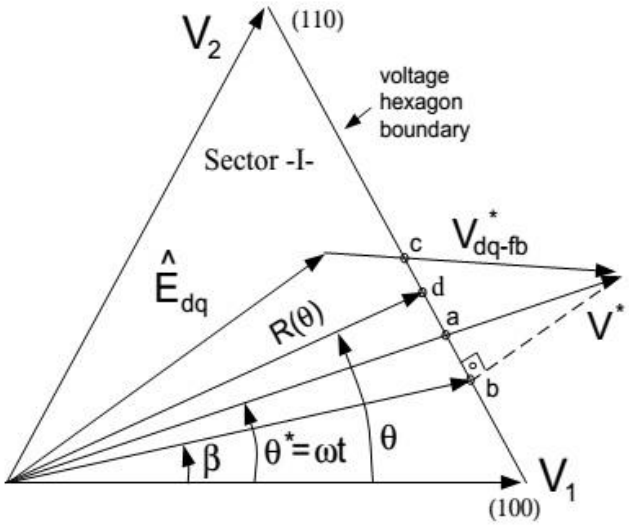

Figure 13. Over-modulation methods [8]

3.3 Six-Step Control

Six-step control is usually used as an open-loop technique i.e. feedback calculation of current is not carried out. This leads to no control over the transient nature of the change in current command. In motor applications, six-step control is used as a scalar method with steady increase in voltage magnitude and frequency in order to start the motor successfully.

In this method, the voltage command to the PWM inverter can have six possible states, hence the name six-step method. These six states are the six vertices of the voltage hexagon. The voltage reference stays at one vertex for a time interval 't_m /6' where t_m is the modulation frequency or the frequency of the output voltage. In every switching interval only one switching action occurs. As can be seen, due to the low number of switching actions, switching

losses are reduced to a great extent. However, due to the reduced switching frequency the harmonic content of the voltage and current increase. The odd harmonics, excluding the third harmonic and its multiples, dominate the harmonic content of the output voltage and current. The switching pattern is as follows:

Table 1. Six-step commutation angles

$0 < \theta \leq 60°$	A = 1, B = 0, C = 1
$60° < \theta \leq 120°$	A = 1, B = 0, C = 0
$120° < \theta \leq 180°$	A = 1, B = 1, C = 0
$180° < \theta \leq 240°$	A = 0, B = 1, C = 0
$240° < \theta \leq 300°$	A = 0, B = 1, C = 1
$300° < \theta \leq 360°$	A = 0, B = 0, C = 1

Note that switch status 1 implies upper switch is on while switch status 0 implies that the lower switch is on.

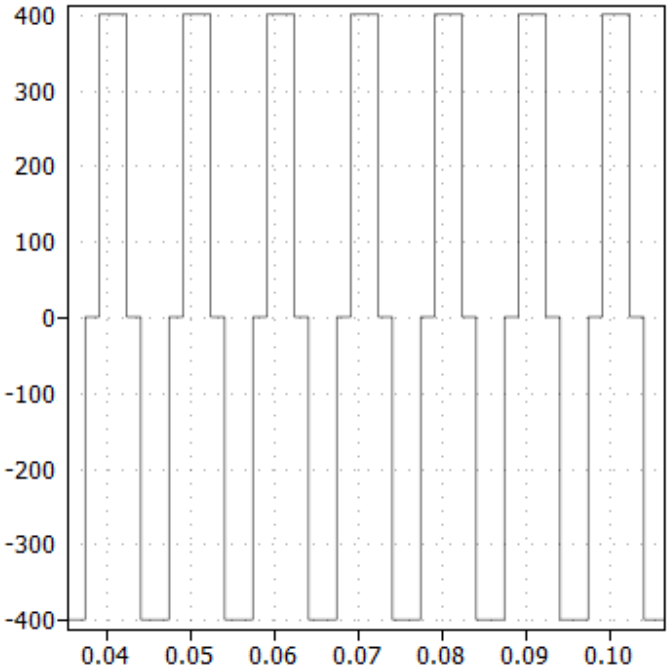

Figure 14. Six-step voltage waveform

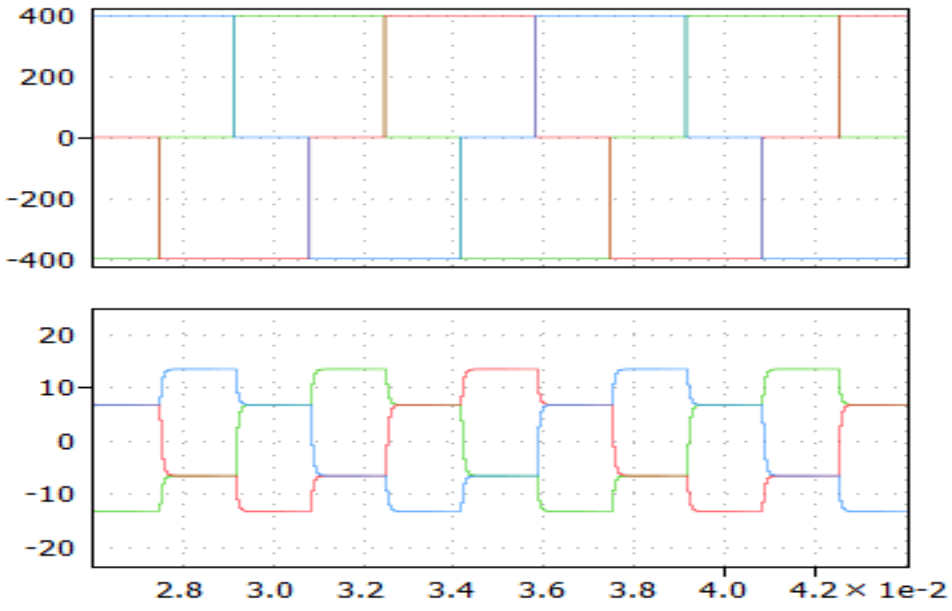

Figure 15. Line voltage and currents

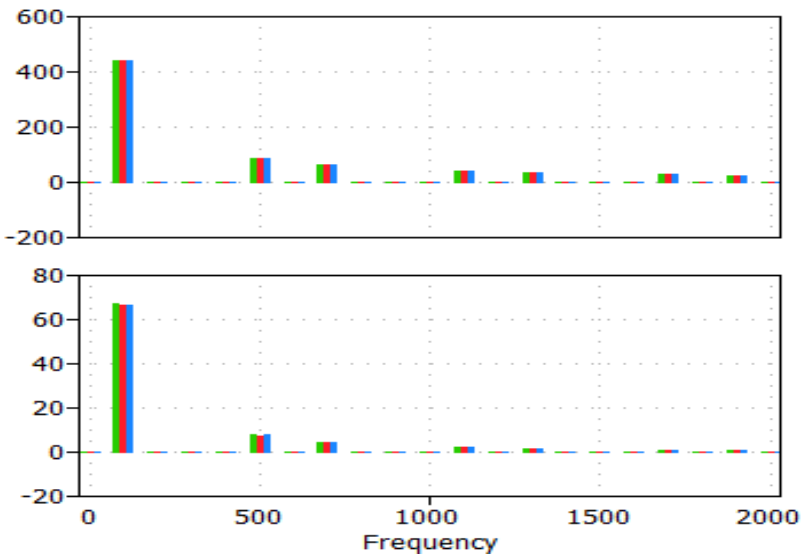

Figure 16. Odd harmonics in Six-Step waveform

CHAPTER 4: PMSM and Control System Modeling

4.1 PMSM mathematical model

The mechanical structure of the PMSM has been explained in chapter 1. The motor behavior can be described by a set of differential equations. These differential equations are then modeled using Simulink/MATLAB for use in the simulations. The model is described in the dq-rotating reference frame for simplicity. The modeling approach is given below.

The effective motor model in the dq reference frame can be represented as in Figure 16. The equations representing the electrical system are:

$$V_{ds} = I_{ds}R_S + L_{ds}\frac{dI_{ds}}{dt} - \omega_r L_{ds} I_{qs} \tag{11}$$

$$V_{qs} = I_{qs}R_s + L_{qs}\frac{dI_{qs}}{dt} + \omega_r (L_{qs} I_{ds} + \lambda_m) \tag{12}$$

where V_{ds} is the d-axis voltage (V)

V_{qs} is the q-axis voltage (V)

I_{ds} is the d-axis current (A)

I_{qs} is the q-axis current (A)

R_s is the stator resistance (ohms)

L_{ds} is the d-axis inductance (H)

L_{qs} is the q-axis inductance (H)

ω_r is the reference frame rotating speed

λ_m is the magnetizing flux

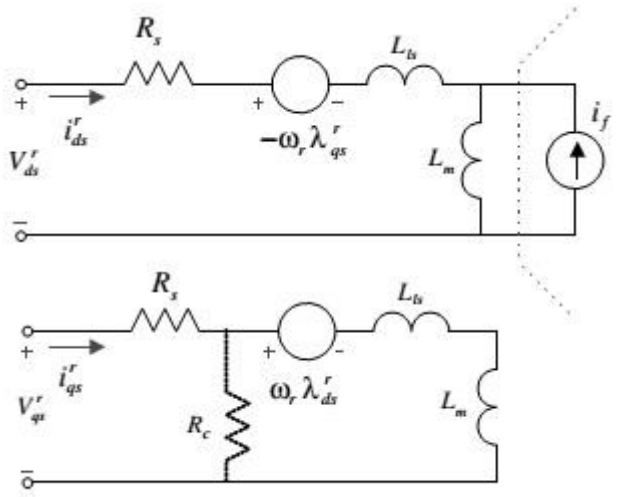

Figure 17. PMSM dq-axes modeling [15]

Now, the motor being considered in this thesis is the Surface Mounted Permanent Magnet Synchronous Motor (SM-PMSM). Since in a synchronous motor, the speed of the stator flux is the same as rotor speed, ω_r is taken to be the electrical speed of the machine. In a SM-PMSM, the d-axis and q-axis inductances are equal i.e.

$$L_{ds} = L_{qs} = L_s \tag{13}$$

The torque of the machine can be represented as

$$T_e = \frac{3}{2}\frac{P}{2}(\lambda_{ds} I_{qs} + \lambda_{qs} I_{ds}) \tag{14}$$

The equation can be simplified using the following relation.

$$\lambda_{dqos} = L_{dqos} I_{dqos} + \lambda_{dqom} \tag{15}$$

However, the magnetic flux in a SM-PMSM exists only along the d-axis. Thus, the following equations can be written.

$$\lambda_{ds} = L_{ds} I_{ds} + \lambda_m \tag{16}$$

$$\lambda_{qs} = L_{qs} I_{qs} \tag{17}$$

In a balanced system, the o-axis components can be assumed to be zero. Using above equations in (14) we get,

$$T_e = \frac{3}{2}\frac{P}{2}(\lambda_m I_{qs} - (L_q - L_d) I_{qs} I_{ds}) \tag{18}$$

Where P is the number of poles of the motor.

According to equation (lqs = lds),

$$T_e = \frac{3}{2}\frac{P}{2}\lambda_m I_{qs} \tag{19}$$

Thus, the electrical torque of the SM-PMSM is solely dependent on q-axis current.

The mechanical equation of the motor is given as follows:

$$J\frac{d\omega_m}{dt} = T_e - T_l - B\omega_r \tag{20}$$

where J is the inertia of the motor (kg-m^2)

T_e is the electrical torque generated by the motor (N-m)

T_l is the load torque applied to motor (N-m)

B is the viscous friction coefficient (N-m-s)

ω_m is the mechanical speed of the motor (rad/s)

The electrical speed and position equations are

$$\omega_r = \frac{P}{2}\omega_m \tag{21}$$

$$\theta_e = \int \omega_r \tag{22}$$

Thus equations (Iqs, Ids, Wr, theta) are used to model the behavior of the motor. Figures 17-20 show the implementation in Simulink.

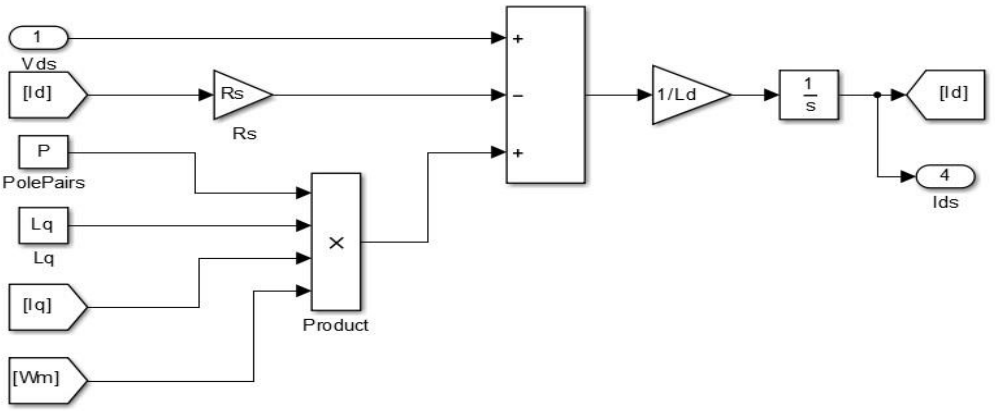

Figure 18. PMSM modeling -I_{ds}

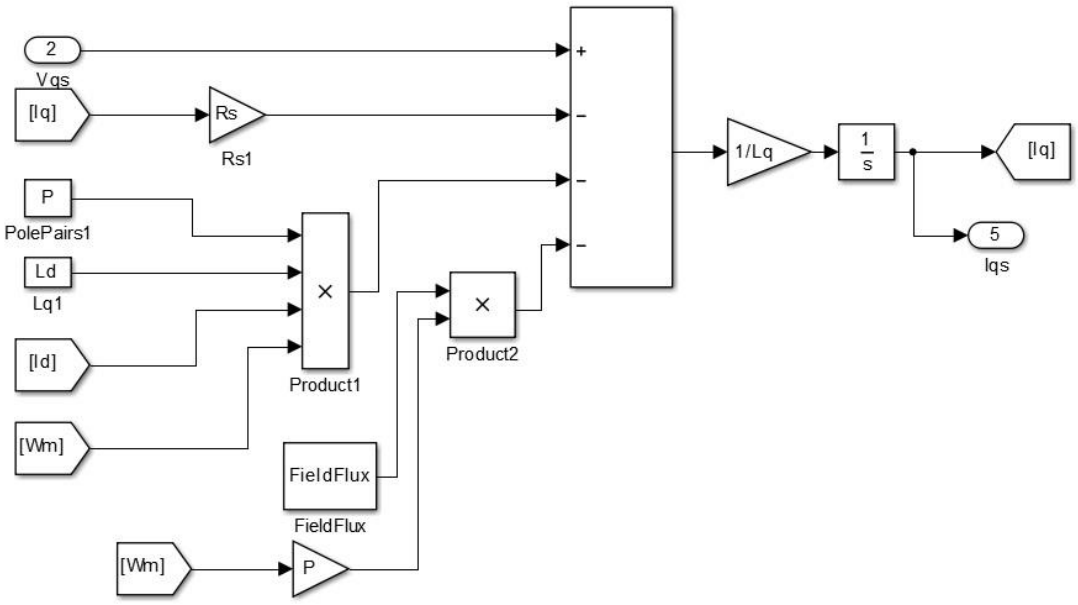

Figure 19. PMSM Modeling – I_{qs}

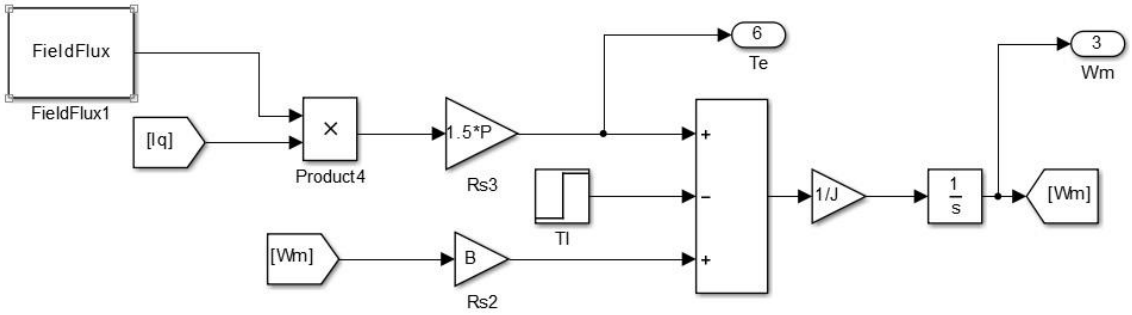

Figure 20. PMSM Modeling - W_m

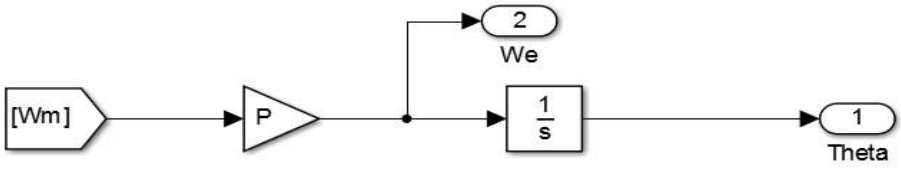

Figure 21. PMSM Modeling – Theta

This model will be used in all simulations. The parameters for the motor are listed in Table 2.

Table 2. Motor parameters

Power	12 kW
L_{ds}	1mH
L_{qs}	1mH
R_s	0.5 ohms

Table 2 continued

P	12
J	0.001 kg-m^2
B	0.005 Nms
λ_m	0.3 Wb

4.2 Motor control system description

The biggest limitation to using six-step method of generating output voltages is that conventional PI control cannot be implemented due to incapability of the PI controller to increase its output further in response to changes in error input. Whenever there is a step increase in the reference parameter (current in this case), the error input to the PI controller suddenly increases and the PI output responds with a large output to bring the controlled parameter close to the reference value. Thus for faster control, it is recommended to have a larger potential overshoot area.

In six-step control the output voltage is operating at its maximum value. Thus for any change in the reference torque, which subsequently results in change in reference current won't affect the rate of change of the load current i.e. load current. Thus the current dynamics are not satisfactory and this method of control is thus not useful in high-performance drive applications. There needs to be a way to control the dynamic rate of change of current for higher performance.

In [16], a method to improve the dynamic performance of six-step control has been introduced. The rate of increase of current in a PMSM depends on the difference between the output voltage of the inverter and the back-emf of the motor. The key part of this method is that a

decrease in the back-emf of the motor can be achieved and this can be utilized to get faster change in current. Neglecting the stator resistance from equations (11) and (12) we get,

$$V_{ds} - E_{ds} = L_s \frac{dI_{ds}}{dt} \tag{23}$$

$$V_{qs} - E_{qs} = L_s \frac{dI_{qs}}{dt} \tag{24}$$

Thus it can be mathematically shown that for if the difference between the back-emf E_s and the applied voltage V_s can be increased, the current control can be made faster. The back-emf equations are,

$$E_{ds} = -\omega_r L_{ds} I_{qs} \tag{25}$$

$$E_{qs} = \omega_r (L_{qs} I_{ds} + \lambda_m) \tag{26}$$

Thus d-axis back-emf can be changed by changing q-axis current and vice-versa. This can be utilized in control of the motor.

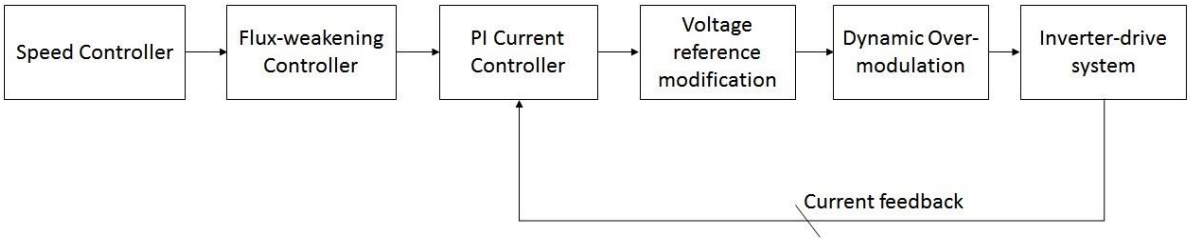

Figure 22. Proposed Control System [16]

The proposed control system is given in Figure 21. The complete system is simulated in MATLAB/Simulink system. Figures 22-23 show the simulation block diagram. It is described below.

4.2.1 Speed Controller

The speed control loop is the outer loop for the drive system. It regulates the speed of the motor to the set reference value. The speed control loop consists of a PI controller which works on the speed error and the output of this PI loop is the torque reference current I_q.

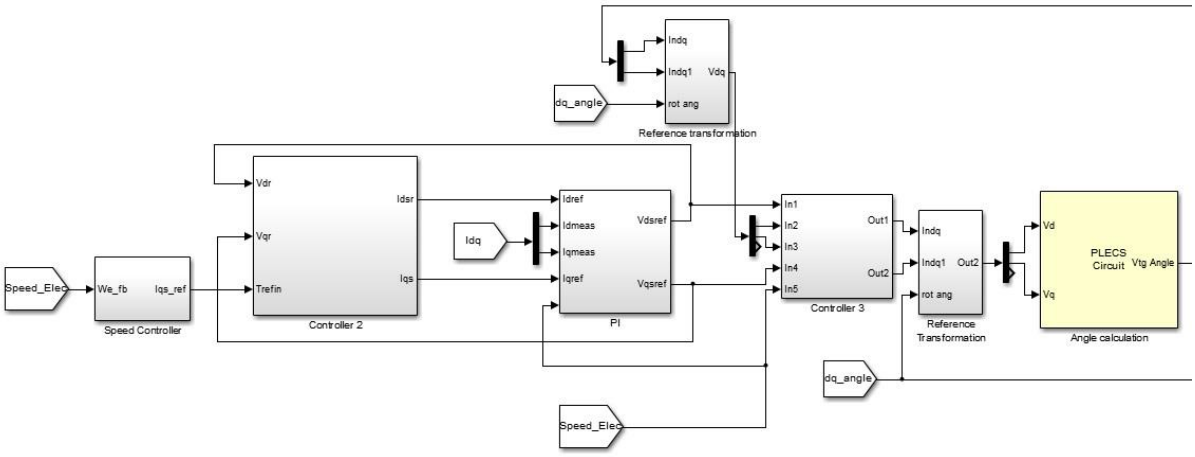

Figure 23. Simulation block diagram – I

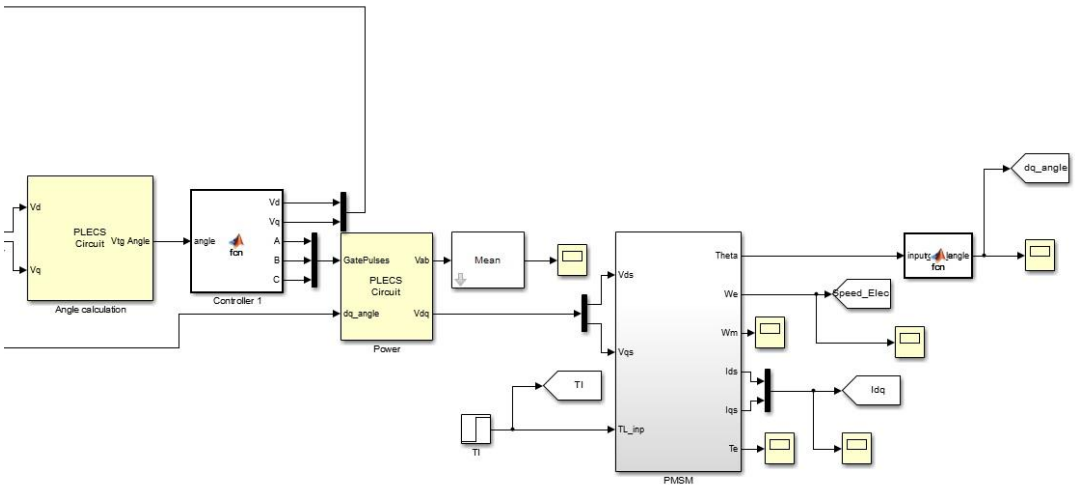

Figure 24. Simulation block diagram – II

The simulation block diagram is as shown in Figure 24. The details about tuning of the controller are given in section 4.2.3.

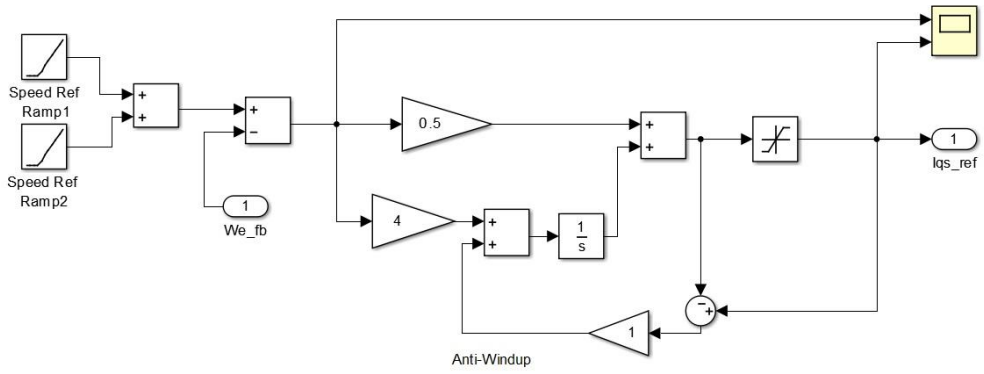

Figure 25. Speed Control Loop

The parameters of the PI loop are given in Table 4.

4.2.2 Flux-weakening controller

As described in section 1.3, the speed of a motor can be increased till its base speed which is proportional to the back-emf of the motor and hence the applied inverter output voltage. Output voltage cannot be increased beyond its maximum value and hence speed cannot be increased beyond the base speed by conventional methods. For further increase in speed a flux-weakening approach has to be taken.

In a PMSM, the flux is supplied by a permanent magnet and hence is constant. Thus to negate the action of this flux, the d-axis current I_d is decreased below zero. The q-axis current has to be reduced according to the maximum current limit for the motor such that,

$$-\sqrt{\mathrm{I}_{\max}^2 - i_{ds}^2} \leq i_{qs} \leq \sqrt{\mathrm{I}_{\max}^2 - i_{ds}^2} \tag{27}$$

The reduction of I_q leads to a reduction in available torque according to (19). The simulation block diagram for the flux-weakening controller is shown in Figure 25.

The voltage limit is taken to be (2/3)*V_{DC} since we are operating in six-step mode. Thus whenever the output magnitude of the PI controller goes above this value, the integrator will act on the error and the magnitude of I_d will be decreased below its set value of zero.

As can be seen in the Figure 25, a MATLAB function is used to regulate the magnitude of the current reference below the maximum permissible current value according to (27). The MATLAB code is given in the appendix.

The current reference output values from this controller are fed to the PI controller.

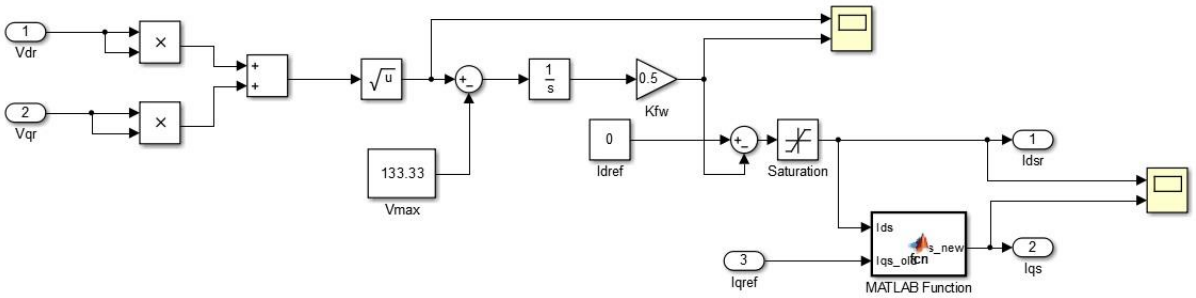

Figure 26. Flux weakening controller

4.2.3 PI Current Controller

The inclusion of PI current controller is one of the most significant improvements in this topology over conventional six-step control. Tuning of PI controller needs knowledge of the desired bandwidth of the controller. This bandwidth depends on the sampling time or switching time of the inverter. There is no defined switching frequency in six-step control. Since the voltage reference lies on a vertex for ($1/6^{th}$) of the total switching time, there is no leeway for additional switching. The speed of rotation or modulating frequency for six-step control depends on the speed of the motor.

Thus knowledge of the operating speed of the motor is essential for proper tuning of the PI controller.

The operating speed of the motor is taken to be 2500 rpm. Thus the operating frequency according to (1) is 250 Hz i.e. 1570.8 rad/s. Since there are six switching actions in one rotation of the voltage vector, it can be said that switching frequency is around six times the operating frequency. Thus the switching frequency is 9424.8 rad/s.

The bandwidth can be chosen to be half of the switching frequency following Nyquist's theorem which is 4712.5 rad/s.

From [17], the values for K_p and K_i can be chosen depending on the values of resistance and inductance of the respective axes.

The speed loop has to be slower than the internal current loop. If this is not followed, the outer loop may interfere in the operation of the inner current loop. Thus the bandwidth is chosen to be 10 times smaller than the bandwidth for current loop i.e. 471.25 rad/s. The values of K_p and K_i are derived from the values of the inertia and frictional coefficient of the machine.

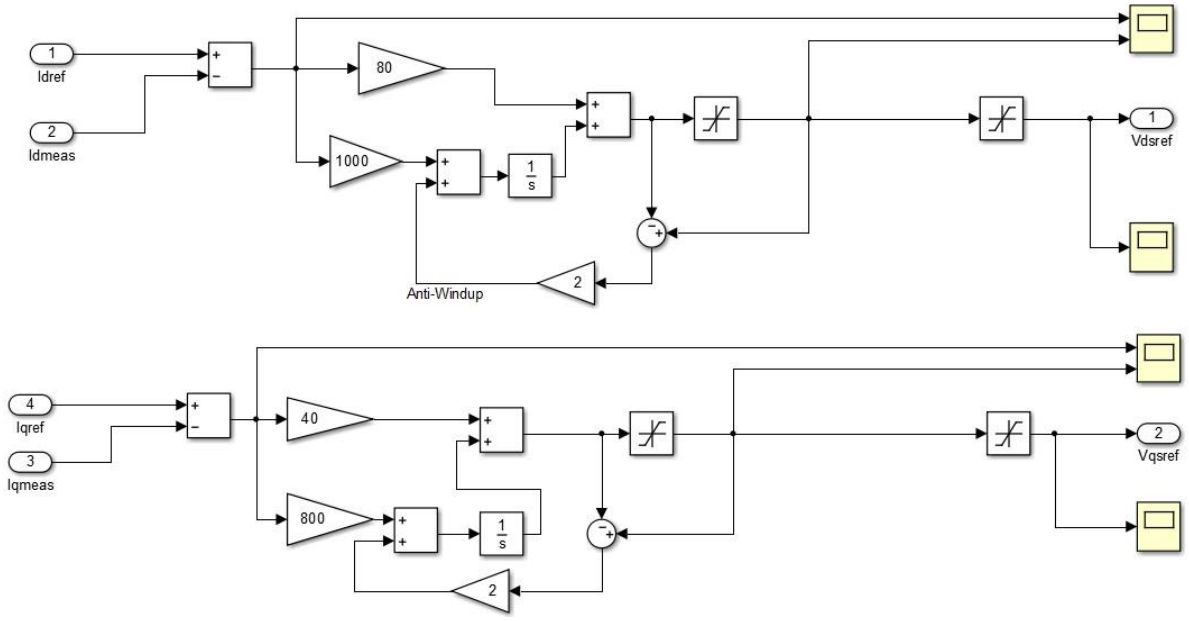

Figure 27. PI controller

4.2.4 Voltage reference modification

As shown before, for rapid changes in current, it is necessary to reduce the respective back-emf values for the motor. So for example, if a rapid increase in torque, i.e. I_q is demanded, it would be necessary to decrease the q-axis back emf to facilitate higher rate of increase of current. According to (26), it can be shown that q-axis back-emf depends on I_d. Thus by bringing about a decrease in I_d can increase I_q rapidly. This action is only temporary and as soon as the I_q current reaches steady state, I_d attains its original value.

This decrease in d-axis current is achieved by decrease in the d-axis reference voltage. This can be seen in Figure 27.

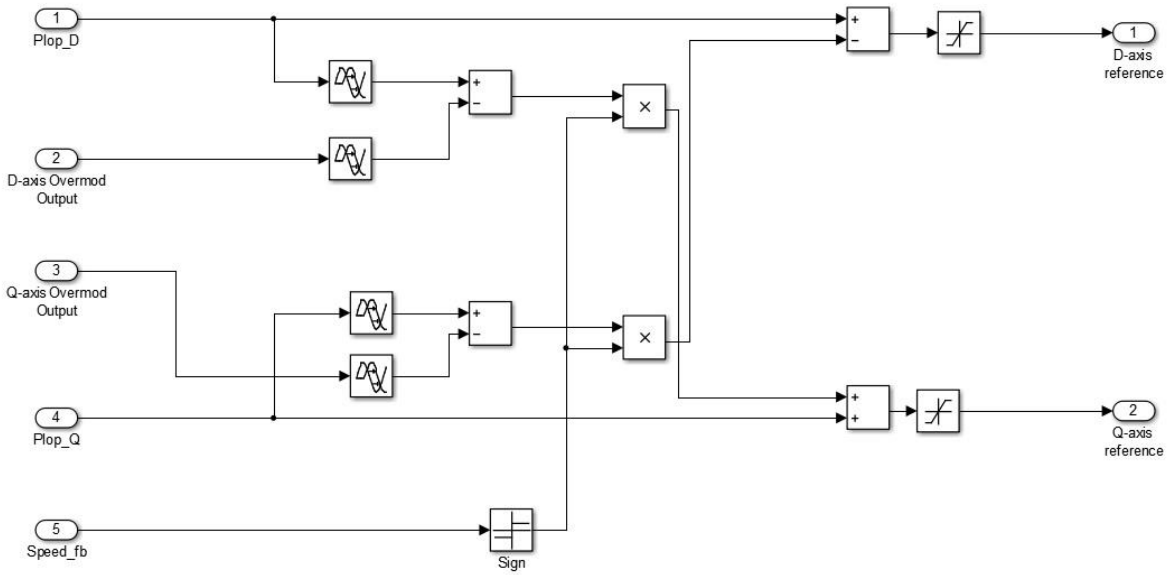

Figure 28. Voltage reference modification control

4.2.5 Dynamic Over-modulation

All the over-modulation methods mentioned in Section 3.2 are used when it is needed to utilize the over-modulation region. Six-step control marks the end of the over-modulation region. Since in this thesis the main aim is to implement six-step algorithm, the over-modulation method adopted is very simple.

Whenever the voltage reference goes outside the voltage hexagon, the vertex closest to the vector is chosen. This is a simplified version of the minimum-distance over-modulation technique. Thus the angle of the reference vector is calculated and logic is developed to choose the proper voltage vector. This is shown in Figures 28-29 and the code to do so is given in the appendix.

The strategy for voltage reference selection is given in Table 3.

Table 3. Strategy for choosing voltage reference vector

If voltage reference angle is…	..choose
$-\frac{\pi}{6} <$ angle $\leq \frac{\pi}{6}$	V_1
$\frac{\pi}{6} <$ angle $\leq \frac{\pi}{2}$	V_2
$\frac{\pi}{2} <$ angle $\leq \frac{5\pi}{6}$	V_3
$-\frac{5\pi}{6} \geq$ angle $< \frac{5\pi}{6}$	V_4
$\frac{-5\pi}{6} \leq angle \leq \frac{-\pi}{2}$	V_5
$\frac{-\pi}{2} \leq angle \leq \frac{-\pi}{6}$	V_6

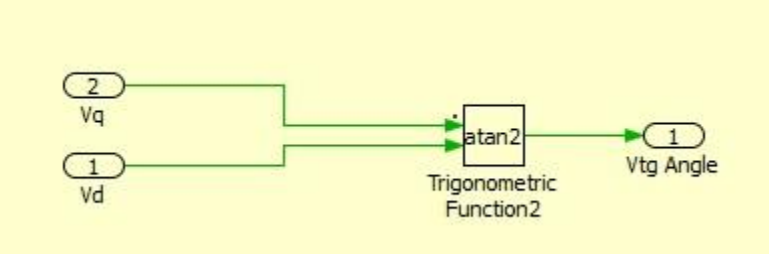

Figure 29. Angle calculation for Dynamic Over-modulation

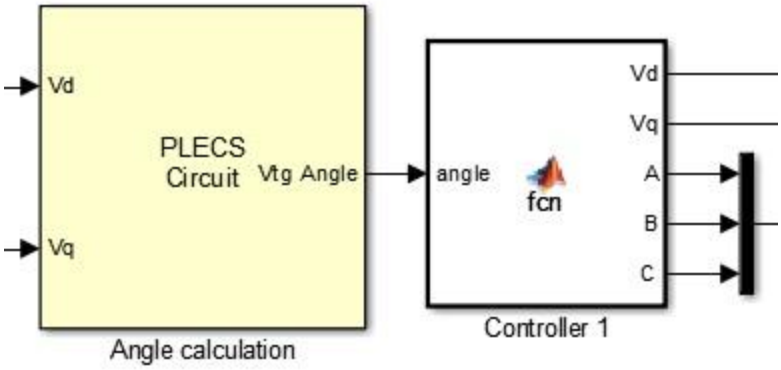

Figure 30. Dynamic Over-modulation controller

4.3 Inverter-Drive System

The PMSM motor model has been described and derived in section 4.1. A basic two-level three-phase inverter is used to generate the output voltage for the motor. Since the motor model requires a dq-reference frame input, the output voltage of the inverter is converted to the rotor reference frame using abc-dq transformation. The simulation model is shown in Figure 30. The load torque on the motor is modeled as a constant, and a step increase is applied to measure the current dynamics.

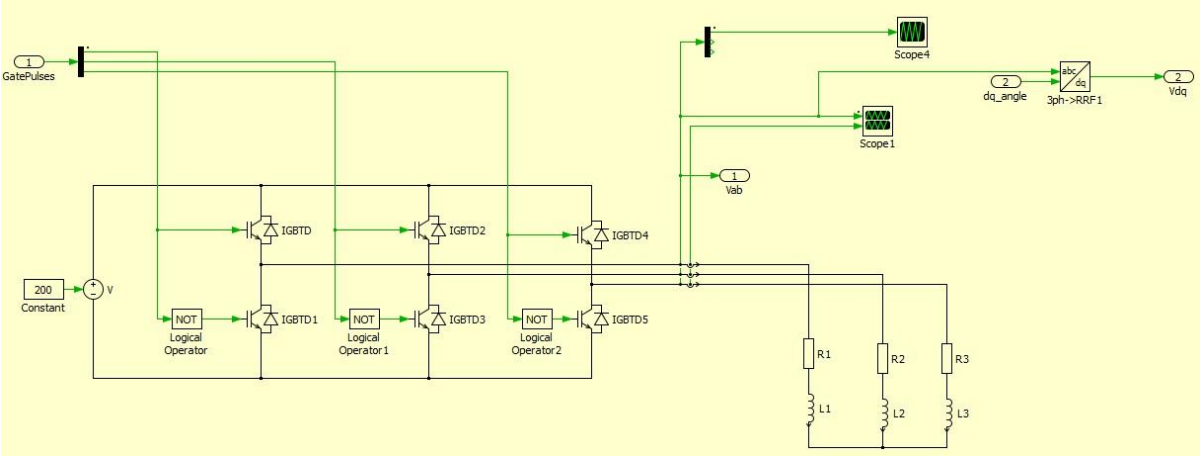

Figure 31. Inverter system

Table 4. Simulation parametric values

Parameters	Value
Speed controller Kp	0.5
Speed controller Ki	4
Operating speed	2500 RPM
V_{lim}	133.33 V
I_d current controller Kp	150
I_d current controller Ki	1000
I_q current controller Kp	80
I_q current controller Ki	800
I_d anti-windup gain	2
I_q anti-windup gain	2

4.4 Simulation results

The simulation results are presented below. The simulation is set up such that there is a step increase in torque reference periodically.

The line-to-line output voltage of the inverter is seen in Figure 31. It can be seen that proper six-step waveform can be obtained.

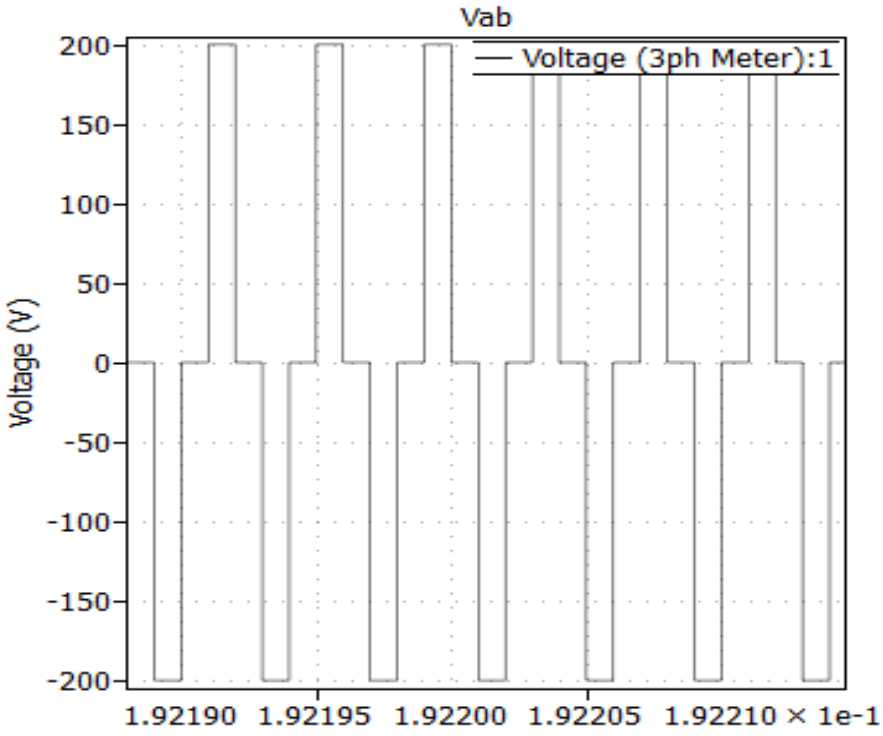

Figure 32. Six-step Voltage

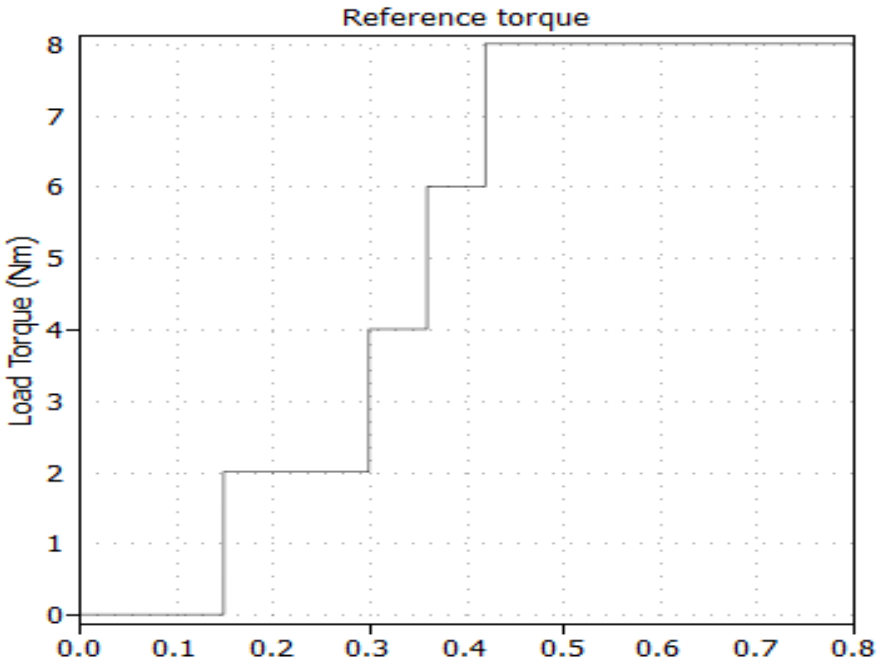

Figure 33. Load torque profile

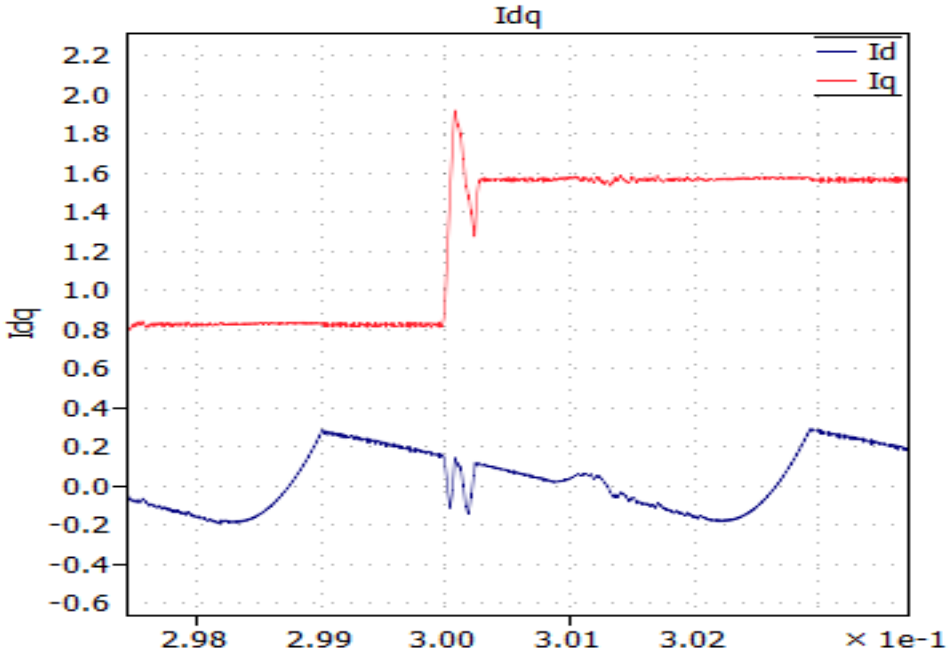

Figure 34. I_{dq} current response on reference step

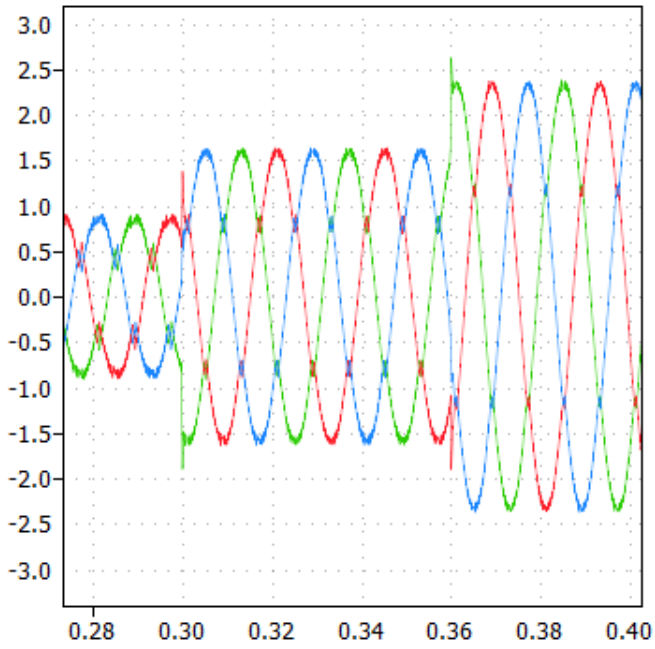

Figure 35. Phase abc current response

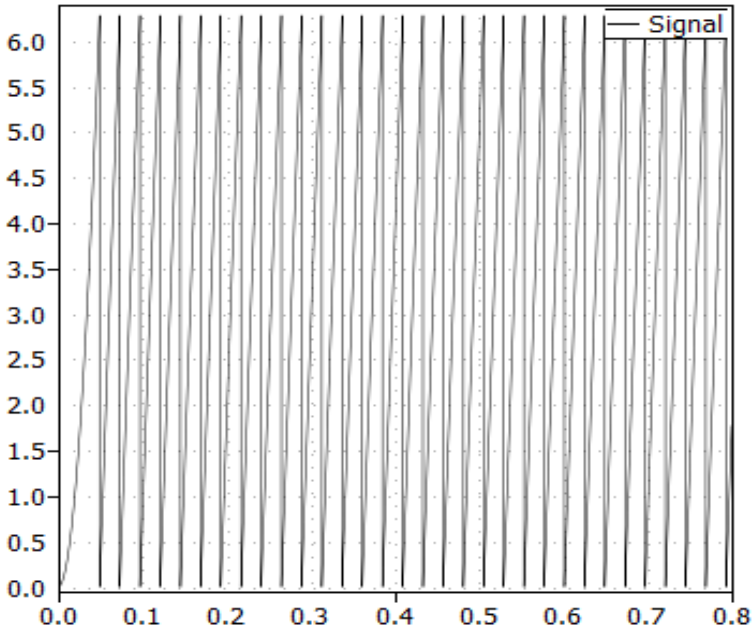

Figure 36. Rotation angle

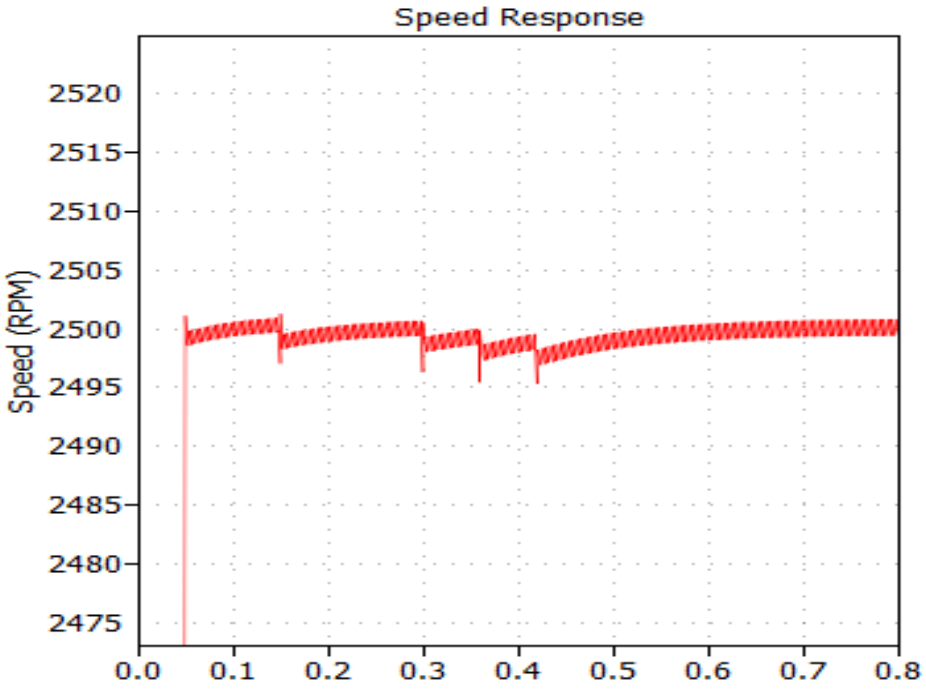

Figure 37. Speed Response

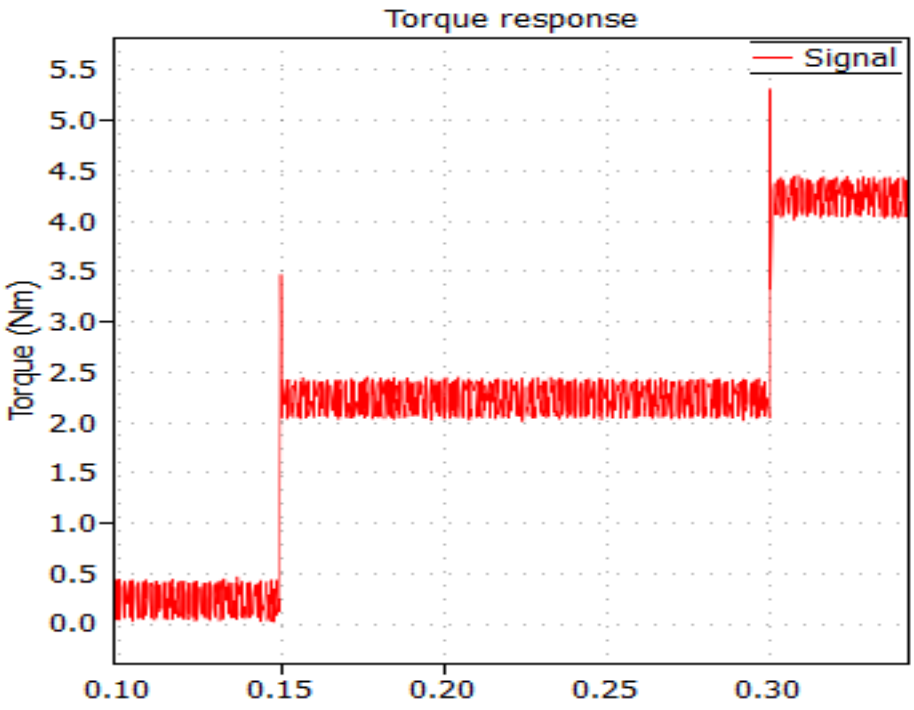

Figure 38. Torque Response

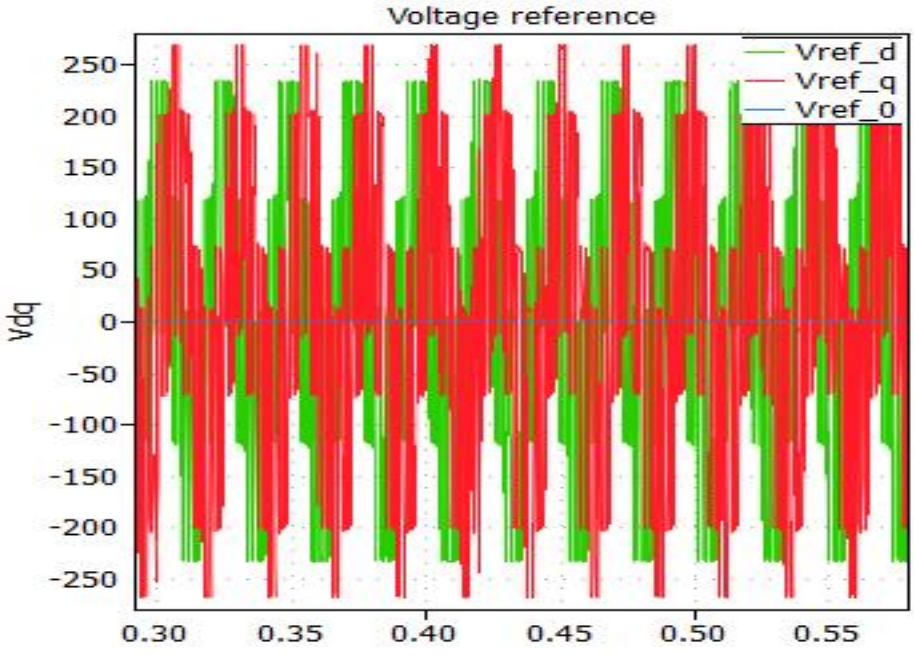

Figure 39. Voltage reference before dynamic over-modulation

4.5 Simulation of transition from SPWM to Six-step method

Six-step method is more efficient and suitable for use in high-performance drives at high speeds only. This is because at lower speeds, the switching speed of the power electronic devices is lower. This leads to higher current ripple especially in low inductance motors. As the speed is increased, the bandwidth of the current controller can be increased and thus better regulation of current can be achieved. In most drives, it is recommended to have at least two modulation methods for most efficient performance.

Sine-PWM is very efficient at lower modulation indexes. The harmonic content of SPWM is very well-defined and thus gives extremely good performance at lower voltage ratios. Thus at lower speeds it is recommended to use SPWM. As the speed increases and thus the modulation index increases, Space vector PWM or GDPWM is recommended, since it gives better performance at higher modulation indexes. GDPWM in general gives better performance in the over-modulation region as well. As the final stage, six-step method is recommended. The simulation model is presented below.

The transition from one method to other is somewhat tricky and if not done properly may lead to oscillatory behavior. It is necessary to switch at key angles in the voltage waveform. When switching to six-step control, it is important to switch when the voltage reference is close to or on a diagonal of the voltage hexagon. This prevents the voltage reference from ‘jumping’ to its next value.

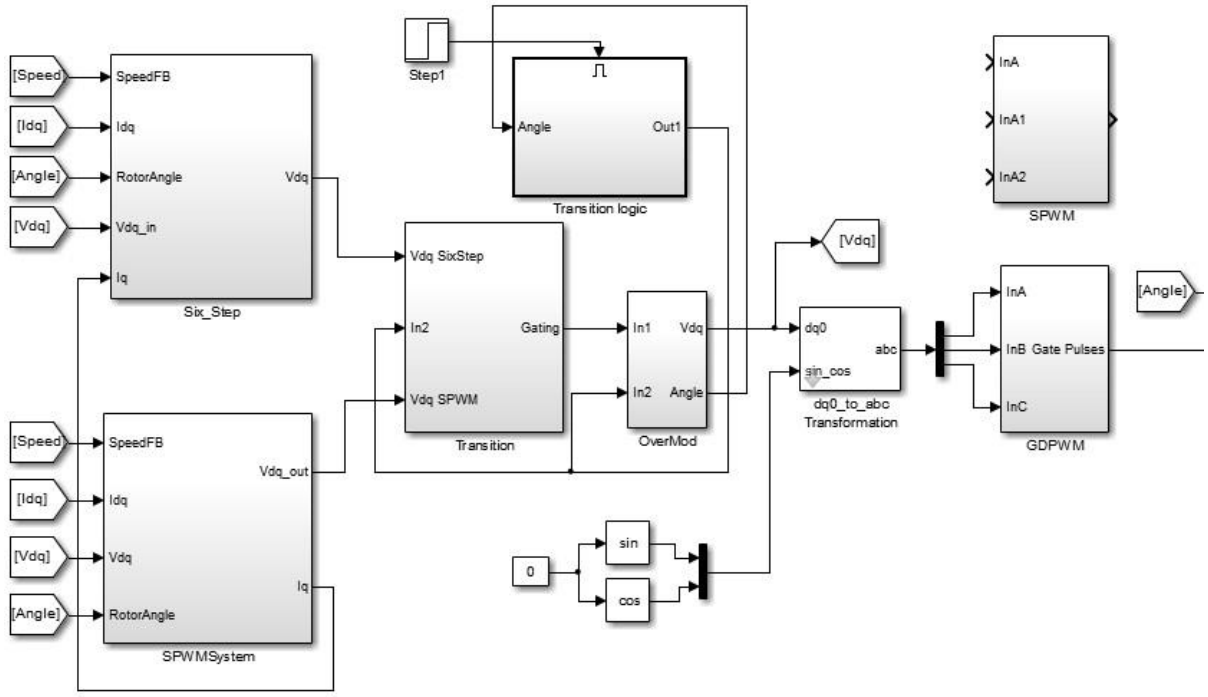

Figure 40. Transition from GDPWM to Six-Step – I

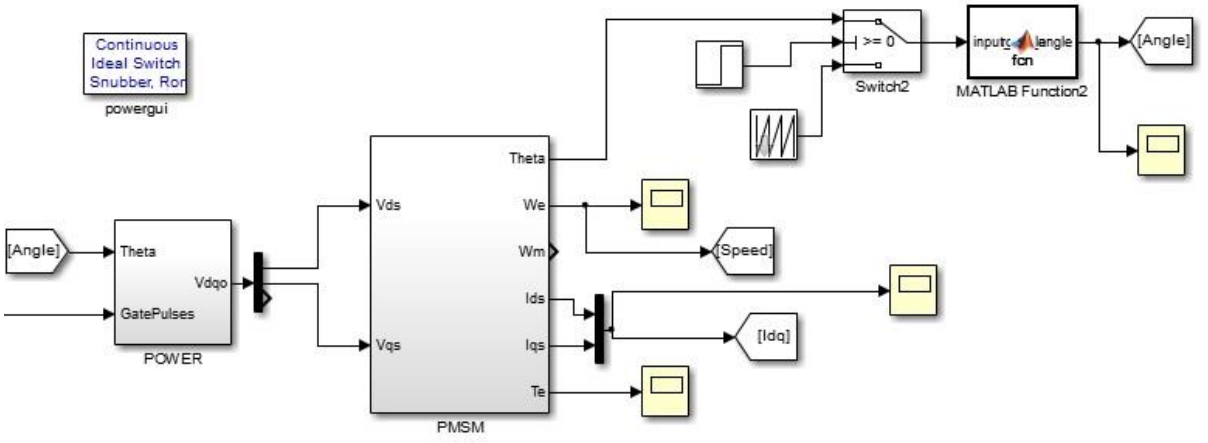

Figure 41. Transition from GDPWM to Six-step – II

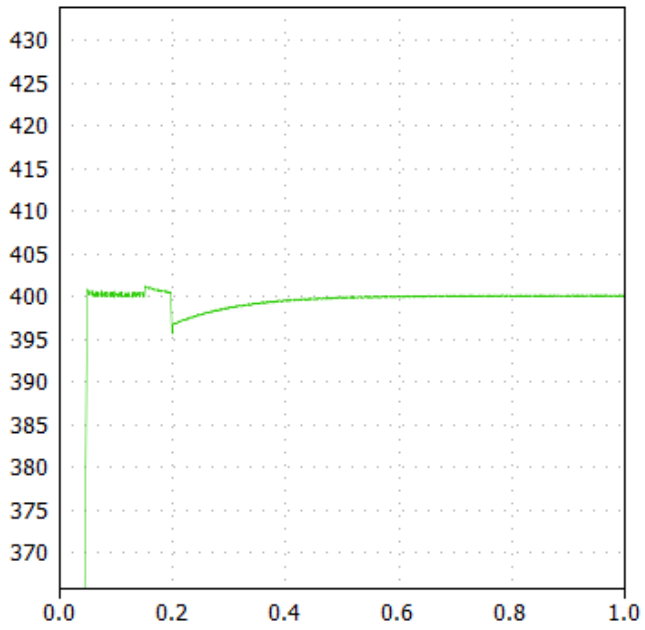

Figure 42. Speed Response after transition from GD-PWM to Six-step

It can be seen from the speed response. The command for transition is given at t=0.15s. When the vector is closest to a hexagon diagonal, the switch to six-step method is made. As can be seen in the response, due to the additional voltage magnitude suddenly available to the motor, the speed rises a little, but then immediately falls back to set speed. A torque increase commanded at t=0.2s and hence the speed of the motor droops and then gets back again to set value.

Since in this method, GD-PWM is used to implement six-step control, the response is somewhat diminished as compared to earlier standalone six-step converter. This is because the voltage commands were directly sent to the PWM inverter rather than as modulating waveforms.

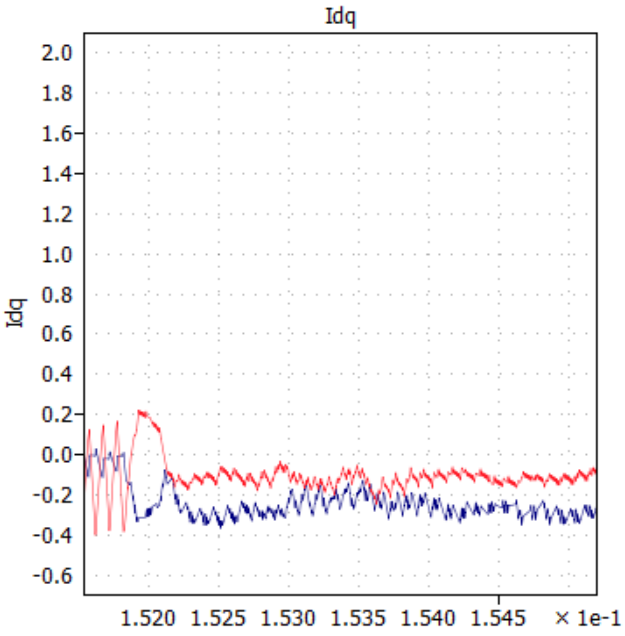

Figure 43. Idq response during transition

The current response during transition can be seen in Figure 40. The effect of the voltage reference modification can be clearly seen as current I_d decreases to accommodate sudden change in I_q. The settling time of the current is also considerably low.

CHAPTER 5. Hardware Tests

The algorithm was also tested on a hardware setup using a motor controller board developed at Joby Motors and Typhoon Hardware-in-the-Loop system (HIL 602). The motor controller board was originally developed for implementation of Field-Oriented Control of a PMSM using space vector PWM. Since the six-step algorithm proposed has no hardware requirements other than hardware present on any motor control board already, it can be implemented on almost any existing board.

The motor controller board developed is pictured in Figure 41. It has both a power stage and a control stage present on a single board. The board is powered from a DC supply and the output voltage is supplied through an inverter to the motor. Each phase of inverter is made up of eight paralleled MOSFETs. This is done to increase the current carrying capability of the inverter. The control board is mounted with a Texas Instruments C2000 TMS320F28069 DSP. It has a system clock of up to 90 MHz. It has sufficient processing power for implementing the six-step algorithm which is not very intensive anyway.

The inverter-motor setup is implemented in Typhoon HIL-602 which is a powerful emulator which can simulate the real-time behavior of various power electronic systems. It has up to 6 FPGAs and has very low step time capability and thus very accurate performance.

The overall circuit used in HIL-602 is shown in Figure 42.

The motor parameters in the HIL system are kept to be the same as in Table 2. The only additional parameter that is to be specified is the encoder pulse frequency. This is set to be 1000 pulses/rev.

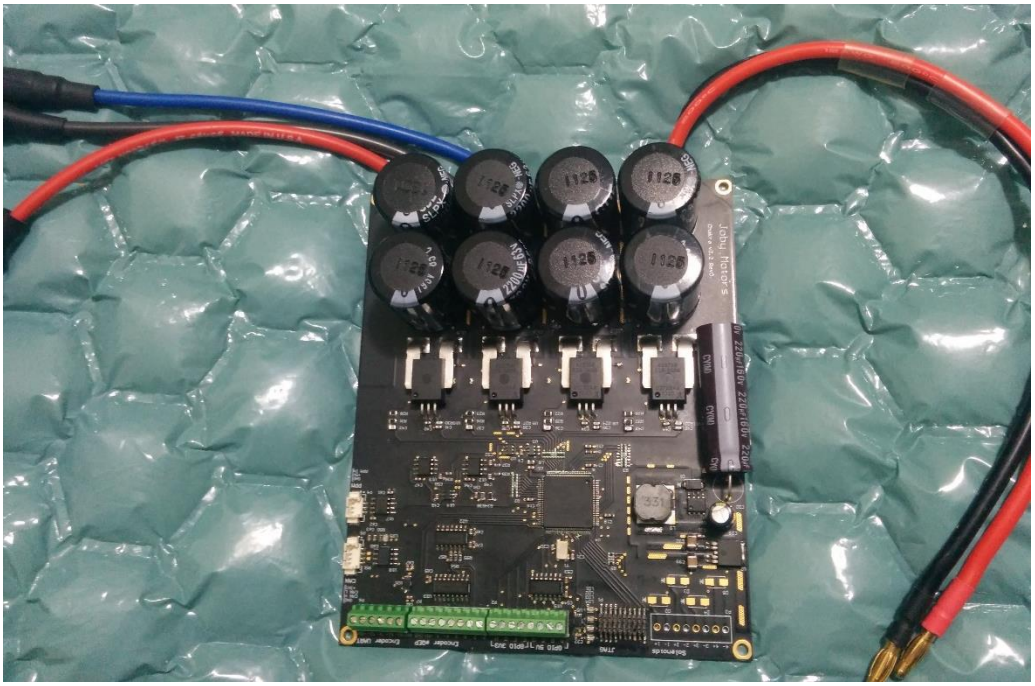

Figure 44. Motor controller board from Joby Motors

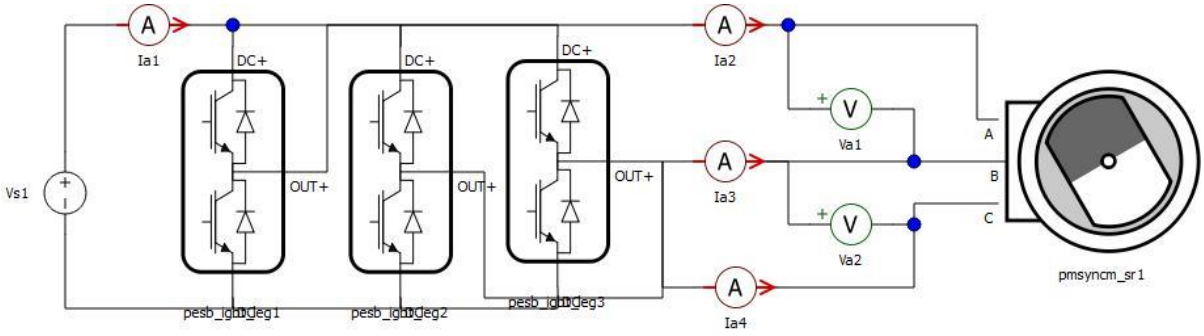

Figure 45. System schematic from HIL-602

The waveforms obtained are shown below. As seen, a very good six-step waveform is obtained. The current waveforms show a very high ripple current. This is because the speed of operation is considerably reduced with the HIL system due to limits reached there.

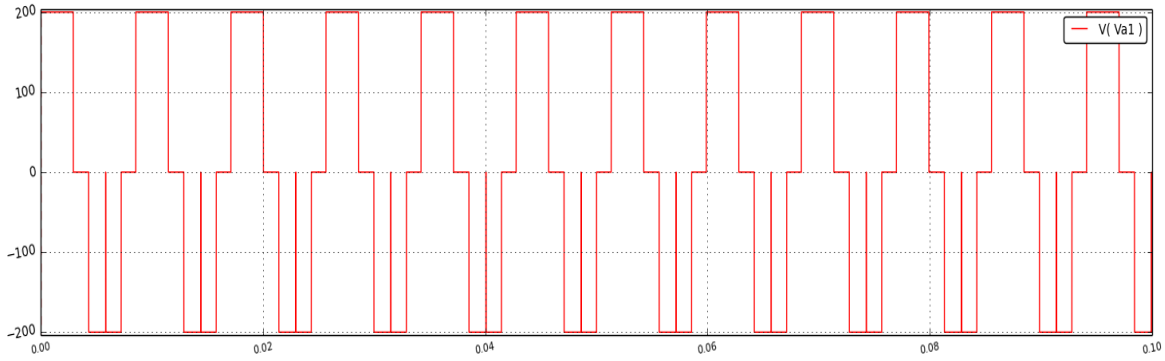

Figure 46. Six-step voltage as seen in T-HIL scope

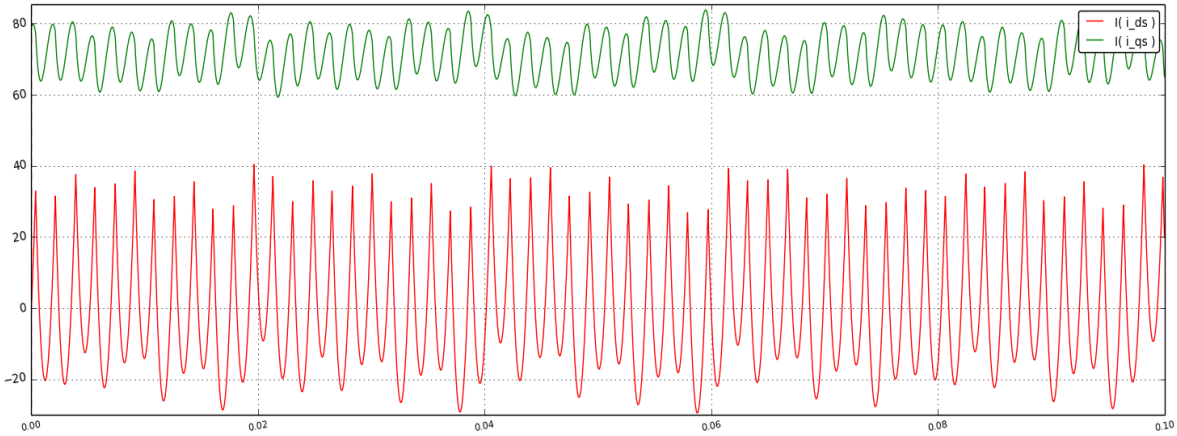

Figure 47. Idq response as seen in T-HIL scope

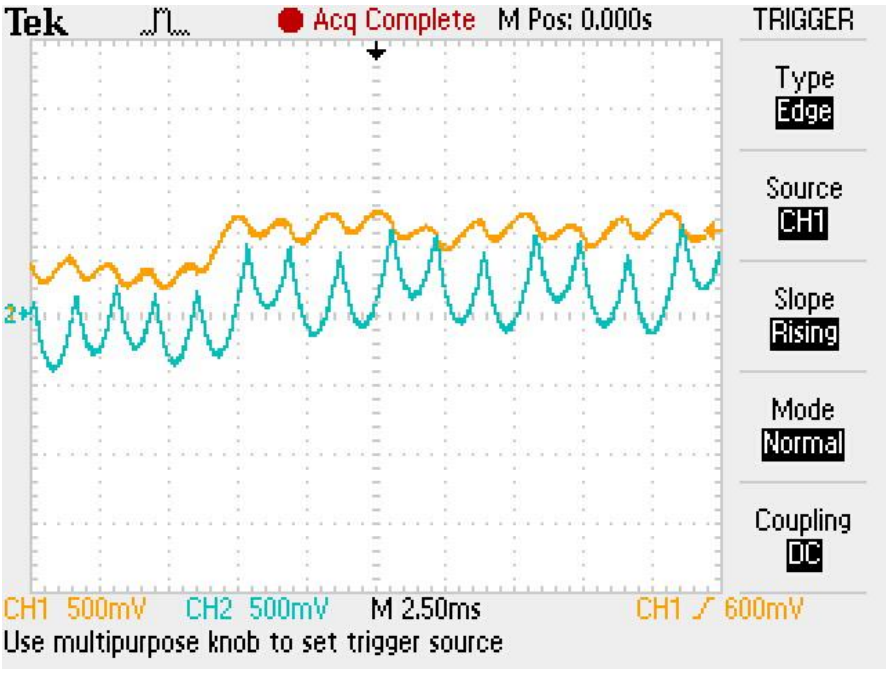

Figure 48. Current transient response

Thus it can be proven that a very good six-step waveform is achieved using the algorithm discussed. The current and torque responses show a lot of ripples but that can be improved by proper tuning of the real motor-inverter control system.

CHAPTER 6. Conclusion

The key takeaways from this work are listed below.

1. Under six-step control operating torque-speed region can be extended beyond that possible by traditionally used methods like Sine-PWM or Space vector PWM.

2. With the same available DC battery voltage, higher speeds can be reached at higher torque levels, so battery utilization of this method is superior. This can be important in systems where size and weight considerations are important which the case for electric vehicles is.

3. Dynamic torque control is achieved using this technique thus making it a good option to have as a final-step high-speed mode in motor controllers with limited battery potential.

4. It can potentially be used in as a stand-alone control system in applications where motion quality is not of prime importance.

5. There is a sizeable current ripple at twice the switching frequency. This can be pushed to higher frequencies by operation at higher speeds and then filtered out. Thus six-step control is suitable predominantly for high-speed operation. It may be possible to increase the DC voltage link by use of rectifier control in proportion to increase in speed, thus limiting the magnitude of the switching ripple.

6. The current ripple may be too high for use in some applications such as high precision speed drives or applications where precise position control is needed.

REFERENCES

[1] D.W. Novotny and T.A. Lipo, “Vector Control and Dynamics of AC Drives.” Oxford Science Publications.

[2] Wikipedia contributors. "Vector control (motor)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 25 Nov. 2014.

[3] Sekerak Peter, Hrabovcova Valéria, Rafajdus Pavol, Kalamen Lukáš, Onufer Matúš, 2012. “Effect of Permanent Magnet Rotor Design on PMSM Properties,” Transactions on Electrical Engineering, Vol (1), No.3.

[4] G. Kohlrusz and D. Fodor, 2011. “Comparison of scalar and vector control methods of an induction motor,” Hungarian Journal of Industrial Chemistry, 39(2), 265-270.

[5] LHP Drives Website. “VFD School,” URL: http://www.lhp.co.in/index_without_right.php?file=vfd_school

[6] University of Paderborn website. “Control of Permanent Magnet Synchronous Motors for Automotive Applications,” URL: http://wwwlea.uni-paderborn.de/en/research/pmsm-control.html

[7] National Programme on Technological Enhanced Learning, “Module 5: DC to AC Converters, Lesson 37 Sine PWM and its Realization.”

[8] Ahmet Hawa dissertation. “Carrier-based PWM-VSI Drives in Over-modulation region.” University of Wisconsin Madison, 1998.

[9] S. R. Bowes and A. Midoun. ”Suboptimal switching strategies for microprocessor controlled PWM inverter drives". IEE Proceedings Vol. 132, Pt. B, No. 3, pages 133{148, May 1985

[10] Dorin O. Neacsu. “Space Vector Modulation – An Introduction,” IECON’01: The 27th Annual Conference of the IEEE Industrial Electronics Society.

[11] Wikipedia contributors, “Space Vector Modulation.” Wikipedia, The Free Encyclopedia, 2 Dec, 2014.

[12] Tomas Sadilek, “A Quick Look on Three-Phase Overmodulation Waveforms.” URL: https://tomontheroll.wordpress.com/2014/12/25/a-quick-look-on-three-phase-overmodulation-waveforms/

[13] J. Holtz, W. Lotzkat, and M. Khambadkone, "On continuous control of PWM inverters in the overmodulation range including the six-step mode," IEEE Trans. Power Electron., vol. 8, no. 4, pp. 546–553, Oct. 1993.

[14] S. Jul-Ki and S. Sul. “A new overmodulation strategy for induction motor drive using space vector PWM". IEEE-Applied Power Elect. Conf, pages 211-216, Dallas, USA, March 1995.

[15] Seung-Ki Sul, “Control of Electric Machine Drive Systems.” Wiley Publications, January 2011.

[16] Yong-Cheol Kwon, Sungmin Kim, Seung-Ki Sul, “Six-Step Operation of PMSM with Instantaneous Current Control.” Industry Applications, IEEE Transactions, Volume: 50, Issue: 4, Pages: 2614-2625

[17] David Mundel Vinoz, Master’s thesis “Design, Simulation and Implementation of a PMSM Drive System.” Chalmers University of Technology, 2011.

APPENDIX

Appendix A

MATLAB code for calculating maximum I_q current.

```
function Iqs_new = fcn(Ids,Iqs_old)
%#codegen

Ilim = 50;

x = sqrt(abs((Ilim^2-Ids^2)));
if (Iqs_old>=x)

    Iqs_new = x;

elseif ((-x)>=Iqs_old)
    Iqs_new = -x;

else
    Iqs_new = Iqs_old;

end
end
```

MATLAB code for generation of six-step vector reference using dynamic over-modulation.

```
function [Vd,Vq,A,B,C] = fcn(angle)
%#codegen

A = 0;
B = 0;
C = 0;
Vd = 0;
Vq = 0;
i = 0;
Vdc = 200;

double ANGLE1;
ANGLE1 = ((1.0/6)*pi);
double ANGLE2;
ANGLE2 = ((1.0/2)*pi);
double ANGLE3;
ANGLE3 = ((5.0/6)*pi);
double ANGLE4;
ANGLE4 = (-(5.0/6)*pi);
double ANGLE5;
ANGLE5 = (-(3.0/6)*pi);
```

```
double ANGLE6;
ANGLE6 = (-(1.0/6)*pi);

if (angle>ANGLE6)&&(angle<=ANGLE1)

A = 1;
B = 0;
C = 1;
i = 1;
Vd = (2.0/3)*Vdc;
Vq = 0;


elseif (angle>ANGLE1)&&(angle<=ANGLE2)

A = 1;
B = 0;
C = 0;
i = 2;
Vd = ((2.0/3)*Vdc)*cos(pi/3);
Vq = ((2.0/3)*Vdc)*sin(pi/3);


    elseif (angle>ANGLE2)&&(angle<=ANGLE3)

A = 1;
B = 1;
C = 0;
i = 3;
Vd = ((2.0/3)*Vdc)*cos(2*pi/3);
Vq = ((2.0/3)*Vdc)*sin(2*pi/3);

    elseif (angle>ANGLE3)||(angle<=ANGLE4)
A = 0;
B = 1;
C = 0;
i = 4;
Vd = ((2.0/3)*Vdc)*cos(pi);
Vq = ((2.0/3)*Vdc)*sin(pi);

    elseif (angle>ANGLE4)&&(angle<=ANGLE5)

A = 0;
B = 1;
C = 1;
i = 5;
Vd = ((2.0/3)*Vdc)*cos(4*pi/3);
Vq = ((2.0/3)*Vdc)*sin(4*pi/3);

    elseif (angle>ANGLE5)&&(angle<=ANGLE6)
```

```
A = 0;
B = 0;
C = 1;
i = 6;
Vd = ((2.0/3)*Vdc)*cos(5*pi/3);
Vq = ((2.0/3)*Vdc)*sin(5*pi/3);

end
end
```

Code Composer Studio firmware code:

Main code file:

```
// Masters thesis firmware code for Six-step control of PMSM

#include "F2806x_Device.h"
#include "IQmathLib.h"
#include "Motor_Control_Math.h"
#include <math.h>
#include "Controllers.h"

// external function prototypes
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);

void PWM_GEN(pwm *pwm1);
void Dyn_OverMod(alphabeta *albeta, pwm *pwm1, alphabeta *FBRef);
void Controller3 (dqo *PIop, dqo *cont3op, dqo *cont1fb,int32 speed);
void Controller2 (dqo *Iref, dqo *PIfb);
void abc_to_dq(dqo *p, angle *w, abc *v, int flag, int zero_term);
void dq_to_abc(abc *ph, angle *w, dqo *p, int flag, int zero_term);
void dq_to_alphabeta(alphabeta *albeta, angle *w, dqo *p);
void alphabeta_to_dq(alphabeta* albeta, angle *w, dqo *p);
void Controller1(alphabeta *albeta, pwm *pwm1, alphabeta *FBRef);
void PI_Controller (PI_cont *PI_arg);
float j;


float x;
int32 Speed = 0;
Uint16 k=0;
int VtgPointer;
int i = 0;
pwm pwm1;
alphabeta alphabeta1, FBref1;
dqo Iref, PIop, cont1fb, Ifb, cont3op;
abc I;
```

```
angle thetaqep = {0,1};
float s=0;

float Speed_Ref = 0;

float Offset[3] = {1.5161,1.5205,1.5173};



PI_cont pi_id, pi_iq,pi_spd;

PI_cont pi_id = {0,1,3,10000,-10000,0,0};
PI_cont pi_iq = {0,50,120,10000,-10000,0,0};
PI_cont pi_spd = {0,0.1,1,10000,-10000,0,0};

alphabeta FBref1 = alphabeta_defaults;
alphabeta alphabeta1 = alphabeta_defaults;
dqo Iref = dqo_defaults;
dqo PIop = dqo_defaults;
dqo cont1fb = dqo_defaults;
dqo Ifb = dqo_defaults;
dqo cont3op = dqo_defaults;
QEP qep1 = QEP_DEFAULTS;
pwm pwm1 =pwm_defaults;
float Iqref = 0;

abc I = {0,0,0};

// Prototype statements for functions found within this file.
void Gpio_select(void);
void Setup_ePWM(void);
interrupt void TINT0_ISR1(void);
void InitAdc(void);
POSSPEED poss1 = POSSPEED_DEFAULTS;

void main(void)
{
      InitSysCtrl();      // Basic Core Init from DSP2833x_SysCtrl.c

//    EALLOW;
//    SysCtrlRegs.WDCR= 0x00AF;  // Re-enable the watchdog
//    EDIS;               // 0x00AF  to NOT disable the Watchdog, Prescaler = 64

      DINT;                     // Disable all interrupts

      Gpio_select();            // GPIO9, GPIO11, GPIO34 and GPIO49 as output
                                    // to 4 LEDs at Peripheral Explorer Board
                                       // GPIO for ePWM


      InitPieCtrl();            // basic setup of PIE table; from
DSP2833x_PieCtrl.c
```

```
       InitPieVectTable(); // default ISR's in PIE

       InitAdc();                // 1

       InitCpuTimers();

       ConfigCpuTimer(&CpuTimer0,90,70);      // Cpu timer 0 (TINT0), 90 MHz, 75
us interval ? Check!
       StartCpuTimer0();

//     qep1.LineEncoder = QEP_PULSE_PER_CHNL;
//     x = 0.25/qep1.LineEncoder;
//     qep1.MechScaler = _IQ30(x);
//     qep1.PolePairs = P/2;
//     qep1.CalibratedAngle = 0;
//     QEP_init(&qep1);


       EALLOW;

       AdcRegs.ADCSOC0CTL.bit.ACQPS = 7;
       AdcRegs.ADCSOC0CTL.bit.CHSEL = 4;
       AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 1;

       AdcRegs.ADCSOC1CTL.bit.ACQPS = 7;
       AdcRegs.ADCSOC1CTL.bit.CHSEL = 5;
       AdcRegs.ADCSOC1CTL.bit.TRIGSEL = 1;

       AdcRegs.ADCSOC2CTL.bit.ACQPS = 7;
       AdcRegs.ADCSOC2CTL.bit.CHSEL = 6;
       AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 1;

       AdcRegs.ADCSOC3CTL.bit.ACQPS = 7;
       AdcRegs.ADCSOC3CTL.bit.CHSEL = 7;
       AdcRegs.ADCSOC3CTL.bit.TRIGSEL = 1;
       EDIS;

       poss1.init(&poss1);

       EALLOW;
       PieVectTable.TINT0 = &TINT0_ISR1;
       EDIS;


       // Enable TINT0 in the PIE: Group 1 interrupt 7
       PieCtrlRegs.PIEIER1.all = M_INT7;             //ADC

       IER |= M_INT1;                  // enable INT1 for TINT0

       EINT;
```

```
       ERTM;

}

void Gpio_select(void)
{
       EALLOW;
       GpioCtrlRegs.GPAMUX1.all = 0;          // GPIO15 ... GPIO0 = General
Puropse I/O
       GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;    // ePWM1A active
       GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1; // ePWM1B active
       GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1; // ePWM2A active
       GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1; // ePWM2B active
       GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1; // ePWM3A active
       GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1; // ePWM3B active

       GpioCtrlRegs.GPAMUX2.all = 0;          // GPIO31 ... GPIO16 = General
Purpose I/O
       GpioCtrlRegs.GPBMUX1.all = 0;          // GPIO47 ... GPIO32 = General
Purpose I/O
       GpioCtrlRegs.GPBMUX2.all = 0;          // GPIO63 ... GPIO48 = General
Purpose I/O

    GpioCtrlRegs.GPAPUD.bit.GPIO24 = 0;   // Enable pull-up on GPIO20 (EQEP1A)
    GpioCtrlRegs.GPAPUD.bit.GPIO25 = 0;   // Enable pull-up on GPIO21 (EQEP1B)
    GpioCtrlRegs.GPAPUD.bit.GPIO27 = 0;   // Enable pull-up on GPIO22 (EQEP1S)
    GpioCtrlRegs.GPAPUD.bit.GPIO26 = 0;   // Enable pull-up on GPIO23 (EQEP1I)

    GpioCtrlRegs.GPAQSEL2.bit.GPIO24 = 0;   // Sync to SYSCLKOUT GPIO20 (EQEP1A)
    GpioCtrlRegs.GPAQSEL2.bit.GPIO25 = 0;   // Sync to SYSCLKOUT GPIO21 (EQEP1B)
    GpioCtrlRegs.GPAQSEL2.bit.GPIO27 = 0;   // Sync to SYSCLKOUT GPIO22 (EQEP1S)
    GpioCtrlRegs.GPAQSEL2.bit.GPIO26 = 0;   // Sync to SYSCLKOUT GPIO23 (EQEP1I)

    GpioCtrlRegs.GPAMUX2.bit.GPIO24 = 2;   // Configure GPIO20 as EQEP1A
    GpioCtrlRegs.GPAMUX2.bit.GPIO25 = 2;   // Configure GPIO21 as EQEP1B
    GpioCtrlRegs.GPAMUX2.bit.GPIO27 = 1;   // Configure GPIO22 as EQEP1S
    GpioCtrlRegs.GPAMUX2.bit.GPIO26 = 2;   // Configure GPIO23 as EQEP1I

       GpioCtrlRegs.GPADIR.all = 0;
       GpioCtrlRegs.GPADIR.bit.GPIO14 = 1;           // test

       GpioCtrlRegs.GPBDIR.all = 0;           // GPIO63-32 as inputs
       EDIS;
}

void Setup_ePWM(void)
{
       EPwm1Regs.TBCTL.bit.CLKDIV =  0; // CLKDIV = 1
       EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;      // HSPCLKDIV = 1
       EPwm1Regs.TBCTL.bit.CTRMODE = 2; // up - down mode
//
```

```
//     // ePWM2 module
//
       EPwm2Regs.TBCTL.bit.CLKDIV =  0; // CLKDIV = 1
       EPwm2Regs.TBCTL.bit.HSPCLKDIV = 0;      // HSPCLKDIV = 1
       EPwm2Regs.TBCTL.bit.CTRMODE = 2; // up - down mode
//
//     // ePWM3 module
//
       EPwm3Regs.TBCTL.bit.CLKDIV =  0; // CLKDIV = 1
       EPwm3Regs.TBCTL.bit.HSPCLKDIV = 0;      // HSPCLKDIV = 1
       EPwm3Regs.TBCTL.bit.CTRMODE = 2; // up - down mode
//
}

interrupt void TINT0_ISR1(void)
{

       I.a = VA_GAIN*((AdcResult.ADCRESULT1)*ADC_GAIN - Offset[0]);
       I.b = VB_GAIN*((AdcResult.ADCRESULT2)*ADC_GAIN - Offset[1]);
       I.c = VC_GAIN*((AdcResult.ADCRESULT3)*ADC_GAIN - Offset[2]);

       poss1.calc(&poss1);

       Speed = poss1.SpeedRpm_pr;

       thetaqep.sin = sin(poss1.theta_elec*3.052e-5*2*PI);
       thetaqep.cos = cos(poss1.theta_elec*3.052e-5*2*PI);

       Iref.q = Iqref;
       Iref.d = 0;

       abc_to_dq(&Ifb, &thetaqep, &I, 1, 0);

       pi_id.Error = Iref.d - Ifb.d;
       PI_Controller(&pi_id);
       PIop.d = pi_id.out;

       pi_iq.Error = Iref.q - Ifb.q;
       PI_Controller(&pi_iq);
       PIop.q = pi_iq.out;

       alphabeta_to_dq(&FBref1,&thetaqep,&cont1fb);

       Controller3(&PIop, &cont3op, &cont1fb,Speed);       // cont3op is the
output of Controller 3 in dq-ref frame
                                                                        //
cont1fb is feedback from cont 1 (needs to be in dq).

       dq_to_alphabeta(&alphabeta1, &thetaqep, &cont3op);
```

```
	Controller1(&alphabeta1, &pwm1, &FBref1);	// alphabeta1 is the
converted cont3op
                                                                        //
FBref1 is the feedback to controller 3 in alphabeta frame

	PWM_GEN(&pwm1);

	// Acknowledge this interrupt to receive more interrupts from group 1
	PieCtrlRegs.PIEACK.all = 1;

}
```

Supplemental code file:

```
/*
 * Controllers.c
 *
 *  Created on: Nov 18, 2014
 *      Author: Sandesh
 */

#include "Controllers.h"
#include "F2806x_Device.h"
#include "math.h"

float angle_rot = 0;
float d_temp=0, q_temp=0,d_temp2=0,q_temp2 = 0;
void PI_Controller (PI_cont *PI_arg);
//

void abc_to_dq(dqo *p, angle *w, abc *v, int flag, int zero_term){

//	float twoa = 2*v->a;
	// flag =1 implies abc to dq, flag = 0 is abc to alphabeta
	if (flag == 1){
		p->d = TWO_THRD*(w->cos*v->a + 0.5*(-w->cos+SQRT_3*w->sin)*v->b +
0.5*(-w->cos-SQRT_3*w->sin)*v->c);
		p->q = -TWO_THRD*(w->sin*v->a + 0.5*(-SQRT_3*w->cos-w->sin)*v->b +
0.5*(SQRT_3*w->cos-w->sin)*v->c);

	}else {

		p->d = ONE_THRD * (2*v->a - v->b - v->c);
		p->q = SQRT_3_3 * (v->b - v->c);

		}
	if(zero_term == 1){
		p->o = ONE_THRD*(v->a + v->b + v->c);
	}else{
		p->o = 0;
	}
```

```
}

void dq_to_abc(abc *ph, angle *w, dqo *p, int flag, int zero_term){
       // flag = 1 is dq - abc, flag = 0 is alphabeta - abc
       if(zero_term == 0){
              p->o = 0;
       }
       if (flag == 1){
       ph->a = w->cos*p->d + w->sin*p->q;
       ph->b = 0.5*((-w->cos + SQRT_3*w->sin)*p->d + (-w->sin - SQRT_3*w->cos)*p-
>q) + p->o;
    ph->c = -(ph->a + ph->b);
       }
       else {
       ph->a = p->d + p->o;
       ph->b = 0.5*(-p->d + SQRT_3*p->q) + p->o;
    ph->c = -(ph->a + ph->b);
       }
}

void dq_to_alphabeta(alphabeta *albeta, angle *w, dqo *p){
       albeta->alpha =  w->cos * p->d - w->sin * p->q;
       albeta->beta = w->sin * p->d + w->cos * p->q;
}

void alphabeta_to_dq(alphabeta* albeta, angle *w, dqo *p)
{
       // Park transform
p->d = albeta->alpha*w->cos + albeta->beta*w->sin;
p->q = albeta->beta*w->cos - albeta->alpha*w->sin;
}
// This implements the Controller I from the paper. The output (returned value)
// shall be the passed value to the PWM_GEN function.

 void Controller1(alphabeta *albeta, pwm *pwm1, alphabeta *FBRef)
  {

        angle_rot = atan2(albeta->beta,albeta->alpha);
       //     angle_rot = _IQatan2PU(Vbeta,Valpha);
              if ((angle_rot > -1*(PI_6)) && (angle_rot <= (PI_6)))
       // Vertex 1
                     {
                     pwm1->Section = 1;
                     FBRef->alpha = TWO_THRD*Vdc*cos(0);
                     FBRef->beta = TWO_THRD*Vdc*sin(0);
                     }
              if ((angle_rot > (PI_6)) && (angle_rot <= (THREE_PI_6)))
       // Vertex 2
                     {
                     pwm1->Section = 2;
                     FBRef->alpha = TWO_THRD*Vdc*cos(PI_3);
```

```
                FBRef->beta = TWO_THRD*Vdc*sin(PI_3);
                   }
            if ((angle_rot > (THREE_PI_6)) && (angle_rot <= (FIVE_PI_6)))
      // Vertex 3
                   {
                   pwm1->Section = 3;
                   FBRef->alpha = TWO_THRD*Vdc*cos(TWO_PI_3);
                FBRef->beta = TWO_THRD*Vdc*sin(TWO_PI_3);
                   }
            if ((angle_rot > (FIVE_PI_6)) || (angle_rot <= -1*(FIVE_PI_6)))
      // Vertex 4
                   {
                   pwm1->Section = 4;
                   FBRef->alpha = TWO_THRD*Vdc*cos(PI);
                FBRef->beta = TWO_THRD*Vdc*sin(PI);
                   }
            if ((angle_rot > -1*(FIVE_PI_6)) && (angle_rot <= -1*(THREE_PI_6)))
            // Verter 5
                   {
                   pwm1->Section = 5;
                   FBRef->alpha = TWO_THRD*Vdc*cos(FOUR_PI_3);
                FBRef->beta = TWO_THRD*Vdc*sin(FOUR_PI_3);
                   }
            if ((angle_rot > -1*(THREE_PI_6)) && (angle_rot <= -1*(PI_6)))
      // Vertex 6
                   {
                   pwm1->Section = 6;
                   FBRef->alpha = TWO_THRD*Vdc*cos(FIVE_PI_3);
                FBRef->beta = TWO_THRD*Vdc*sin(FIVE_PI_3);
                   }
      }

      // This function serves as the PWM generator and gives appropriate signals
to the switches.

       void PWM_GEN (pwm *pwm1)
      {

            if (pwm1->Section == 1)
            {

                   EPwm1Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing
disabled on output A of EPWM1          */\
                   EPwm1Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM1    */\

                         /*
                               */\
                   EPwm2Regs.AQCSFRC.bit.CSFA = 1;              /* Forcing a
continuous Low on output A of EPWM2    */\
```

```
                    EPwm2Regs.AQCSFRC.bit.CSFB = 2;              /* Forcing a
continuous High on output B of EPWM2   */\

                          /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing a
continuous High on output A of EPWM3   */\
                    EPwm3Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM3    */\

                    pwm1->pointer = 0;
             }

             if (pwm1->Section == 2)
             {
                    EPwm1Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing
disabled on output A of EPWM1          */\
                    EPwm1Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM1    */\

                          /*
                                */\
                    EPwm2Regs.AQCSFRC.bit.CSFA = 1;              /* Forcing a
continuous Low on output A of EPWM2    */\
                    EPwm2Regs.AQCSFRC.bit.CSFB = 2;              /* Forcing a
continuous High on output B of EPWM2   */\

                          /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 1;              /* Forcing a
continuous High on output A of EPWM3   */\
                    EPwm3Regs.AQCSFRC.bit.CSFB = 2;              /* Forcing a
continuous Low on output B of EPWM3    */\
                    pwm1->pointer = 1;
             }

             if (pwm1->Section == 3)
             {
                    EPwm1Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing
disabled on output A of EPWM1          */\
                    EPwm1Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM1    */\

                          /*
                                */\
                    EPwm2Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing a
continuous Low on output A of EPWM2    */\
                    EPwm2Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous High on output B of EPWM2   */\
```

```
                        /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 1;             /* Forcing a
continuous High on output A of EPWM3    */\
                    EPwm3Regs.AQCSFRC.bit.CSFB = 2;             /* Forcing a
continuous Low on output B of EPWM3     */\
                    pwm1->pointer = 2;
             }

             if (pwm1->Section == 4)
             {
                    EPwm1Regs.AQCSFRC.bit.CSFA = 1;             /* Forcing
disabled on output A of EPWM1           */\
                    EPwm1Regs.AQCSFRC.bit.CSFB = 2;             /* Forcing a
continuous Low on output B of EPWM1     */\

                        /*
                                */\
                    EPwm2Regs.AQCSFRC.bit.CSFA = 2;             /* Forcing a
continuous Low on output A of EPWM2     */\
                    EPwm2Regs.AQCSFRC.bit.CSFB = 1;             /* Forcing a
continuous High on output B of EPWM2    */\

                        /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 1;             /* Forcing a
continuous High on output A of EPWM3    */\
                    EPwm3Regs.AQCSFRC.bit.CSFB = 2;             /* Forcing a
continuous Low on output B of EPWM3     */\
                    pwm1->pointer = 3;
             }

             if (pwm1->Section == 5)
             {
                    EPwm1Regs.AQCSFRC.bit.CSFA = 1;             /* Forcing
disabled on output A of EPWM1           */\
                    EPwm1Regs.AQCSFRC.bit.CSFB = 2;             /* Forcing a
continuous Low on output B of EPWM1     */\

                        /*
                                */\
                    EPwm2Regs.AQCSFRC.bit.CSFA = 2;             /* Forcing a
continuous Low on output A of EPWM2     */\
                    EPwm2Regs.AQCSFRC.bit.CSFB = 1;             /* Forcing a
continuous High on output B of EPWM2    */\

                        /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 2;             /* Forcing a
continuous High on output A of EPWM3    */\
```

```
                    EPwm3Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM3     */\
                    pwm1->pointer = 4;
             }

             if (pwm1->Section == 6)
             {
                    EPwm1Regs.AQCSFRC.bit.CSFA = 1;              /* Forcing
disabled on output A of EPWM1           */\
                    EPwm1Regs.AQCSFRC.bit.CSFB = 2;              /* Forcing a
continuous Low on output B of EPWM1     */\

                          /*
                                */\
                    EPwm2Regs.AQCSFRC.bit.CSFA = 1;              /* Forcing a
continuous Low on output A of EPWM2     */\
                    EPwm2Regs.AQCSFRC.bit.CSFB = 2;              /* Forcing a
continuous High on output B of EPWM2    */\

                          /*
                                */\
                    EPwm3Regs.AQCSFRC.bit.CSFA = 2;              /* Forcing a
continuous High on output A of EPWM3    */\
                    EPwm3Regs.AQCSFRC.bit.CSFB = 1;              /* Forcing a
continuous Low on output B of EPWM3     */\
                    pwm1->pointer = 5;
             }

      }

      // Controller 3 : This carries out the reduction in back-emf values for
faster control dynamics.

       void Controller3 (dqo *PIop, dqo *cont3op, dqo *cont1fb, int32 speed)
       {

             q_temp2 = PIop->q;
             d_temp2 = PIop->d;
             if (speed >= 0)
             {
                    cont3op->d = d_temp2 - q_temp;
                    cont3op->q = q_temp2 + d_temp; // sign (Wr) check?
             }
             else {
                    cont3op->d = d_temp2 + q_temp;
                    cont3op->q = q_temp2 - d_temp; // sign (Wr) check?
             }
             d_temp = PIop->d - cont1fb->d;
             q_temp = PIop->q - cont1fb->q;
       }
```

```
// Controller 2: This is the flux-weakening mode controller.

void Controller2 (dqo *Iref, dqo *PIfb)
{
	float temp, Id, Iq, temp1;
	PI_cont PI1;
	temp = sqrt(pow(PIfb->d,2) + pow(PIfb->q,2));
	temp = temp - Vlim;
	// pass temp through an integrator with gain
	PI1.Error = temp;
	PI1.Kp = 0;
	PI1.Ki = 0.1;
	PI_Controller(&PI1);
	Id = Idset;
	Iref->d = Id - PI1.out;
	if (Iref->d <= NegIdmax)
	{
		Iref->d = NegIdmax;
	}
	temp1 = pow(Ilim,2) - pow(Iref->d,2);

	if (temp1 < 0)
	{
		Iq = -1*sqrt(-1*temp1);
		if (Iref->q<=Iq)
		{
			Iref->q = Iq;
		}
	}
	else
	{
		Iq = sqrt(temp1);
		if (Iref->q>=Iq)
		{
			Iref->q = Iq;
		}
	}
}

void PI_Controller (PI_cont *PI_arg)
{
	PI_arg->temp1 = PI_arg->temp1 + PI_arg->Error*TS;
	PI_arg->out  = PI_arg->Kp*PI_arg->Error + PI_arg->Ki*PI_arg->temp1;
	if (PI_arg->out >= PI_arg->MaxPos)
	{
		PI_arg->out = PI_arg->MaxPos;
	}
	if (PI_arg->out <= PI_arg->MinPos)
	{
		PI_arg->out = PI_arg->MinPos;
```

```
            }
        }
```

Initialization code file:

```
/*
 * Controllers.h
 *
 *  Created on: Nov 18, 2014
 *      Author: Sandesh
 */

#ifndef CONTROLLERS_H_
#define CONTROLLERS_H_

#include "F2806x_Device.h"
#include "IQmathLib.h"
#include "Motor_Control_Math.h"
#include <math.h>

#define TS 1e-6
#define TWO_THRD .6666667
#define SQRT_3 1.73205080
#define SQRT_3_2 0.86602540
#define SQRT_3_3 0.57735027
#define ONE_THRD .3333333
#define PI 3.14159265
#define PI_6 0.52359877
#define THREE_PI_6 1.57079632
#define FIVE_PI_6 2.61799387
#define PI_3 1.04719755
#define TWO_PI_3 2.09439510
#define FOUR_PI_3 4.18879020
#define FIVE_PI_3 5.23598775
#define Vdc 200           // DC voltage
#define Vlim 133.33       // Voltage limit
#define Ilim 50                 // Current limit
#define Flux 0.3          // FLux level
#define P 12              // number of poles
#define NegIdmax 7.5
#define ADC_GAIN 0.0008058
#define VA_GAIN 332.79
#define VB_GAIN 331.40
#define VC_GAIN 328.61
#define Idset 0

typedef struct {
       float a;     // phase a variable
       float b;     // phase b variable
       float c;     // phase c variable
} abc;
```

```
typedef struct {
	float d;	// d-axis variable
	float q;	// q-axis variable
	float o;	// o-axis variable
} dqo;

typedef struct {
	float alpha; // alpha-axis variable
	float beta;		// beta-axis variable
} alphabeta;

typedef struct {
	float sin;
	float cos;
} angle;

typedef struct {
	int Section;
	int pointer;
} pwm;

#define pwm_defaults {	\
	0,	\
	0	\
}


typedef struct {
		float Error; // error signal input
		float Kp;		// Proportional gain
		float Ki;		// Integral gain
		float MaxPos;// Maximum value of output (saturation)
		float MinPos;// Minimum value of output (saturation)
		float out;		// Output
		float temp1;
} PI_cont;

#define dqo_defaults {	\
			0,	\
			0,	\
			0	\
			}

#define alphabeta_defaults {	\
			0,	\
			0	\
			}

#endif /* CONTROLLERS_H_ */
```